

Recent Developments of World-Line Monte Carlo Methods

Naoki KAWASHIMA* and Kenji HARADA[†]

Department of Physics, Tokyo Metropolitan University, Tokyo 192-0397

[†]*Department of Applied Analysis and Complex Dynamical Systems, Kyoto University, Kyoto 606-8501*

(Received December 22, 2003)

World-line quantum Monte Carlo methods are reviewed with an emphasis on breakthroughs made in recent years. In particular, three algorithms — the loop algorithm, the worm algorithm, and the directed-loop algorithm — for updating world-line configurations are presented in a unified perspective. Detailed descriptions of the algorithms in specific cases are also given.

KEYWORDS: quantum spin system, Heisenberg model, quantum Monte Carlo, XY model, XXZ model, cluster algorithm, loop algorithm, worm algorithm, directed-loop algorithm, world-line method

DOI: 10.1143/JPSJ.73.1379

1. Introduction

The Monte Carlo method based on a Markov process has been quite a powerful tool of the model analysis in many-body physics such as condensed matter physics, statistical physics and field theory. In the present review, we focus on a branch of Markov-chain Monte Carlo methods that have been developed remarkably during the past decade, i.e., the quantum Monte Carlo¹⁾ that samples from an ensemble of *world-line* configurations in the path-integral representation of the partition function. The methodological advancement is largely due to the global update of the world-line configurations. The breakthrough was made by Evertz *et al.*,²⁾ who proposed a new algorithm, called a *loop algorithm*, for the $s = 1/2$ XXZ model, and later also by Prokof'ev *et al.*³⁾ whose approach, called the worm algorithm, seemed quite different at first sight. In a loop algorithm, the world-line configuration is updated in the unit of loops in the space-time formed by a stochastic procedure. It turned out that the loop update does not only reduce the critical slowing-down, but it also removes several other drawbacks of the conventional quantum Monte Carlo. In the worm algorithm,³⁾ on the other hand, the world-line configuration is updated by the movements of a *worm*, i.e., a pair of artificial singular points at which world-lines are discontinuous. A framework was proposed⁴⁾ recently that unifies these two ways of updating and enjoys the virtues of both. In the present article, therefore, we focus on three important algorithms (or, to be more precise, three frameworks for algorithms); the loop algorithm,²⁾ the worm algorithm,³⁾ and the directed-loop algorithm.⁴⁾ In some special cases two of them are identical, *e.g.*, the directed-loop algorithm applied to the $s = 1/2$ antiferromagnetic Heisenberg model is nothing but a single-cluster version of the loop algorithm.

Before the proposal of these new algorithms, simulations had been done with local updating rule on the discretized imaginary time. The local updating rule is analogous to the single-spin-flip Metropolis algorithm of the Ising model. While it provided the first systematic means of numerical study of systems at finite temperatures, it had a number of

drawbacks; (i) the critical slowing-down, (ii) the fine-mesh slowing-down (i.e., the slowing-down when the discretization step of the imaginary time is decreased), (iii) non-ergodicity (the temporal and the spatial winding numbers are conserved), (iv) the discretization error, and (v) difficulty in measuring the off-diagonal quantities, (vi) the negative-sign problem. These drawbacks have been removed (or at least reduced) by the recent development of the quantum Monte Carlo method mentioned above. The critical slowing-down and the fine-mesh slowing-down have been reduced to the negligible level in most applications.^{2–6)} The non-ergodicity and the discretization error have been completely removed.⁷⁾ In addition, most of the off-diagonal quantities of interest can be measured.^{3,8,9)} The negative-sign problem is the toughest and only very limited solution is available. However, there is at least a few cases where this difficulty can be overcome by the loop algorithm.^{10,11)}

There are a number of articles already published on the quantum Monte Carlo. We here only refer the reader to a review article¹²⁾ for the achievement made before the loop algorithm was proposed. For the loop algorithm and related algorithms, an excellent overview¹³⁾ has been written on the loop algorithm by one of the founders of the algorithm. Still there remains a lot of technical difficulties for an unfamiliar reader to start simulations from scratch. Therefore, we feel it useful to put various technical details together with the background mathematics. The purpose of the present article is, therefore, to present various ingredients in a single article in a form comprehensible to non-specialists and ready to use for practitioners. In what follows we describe how we perform the simulation in detail and take a particular care in making the article practical. On the other hand, we do not intend to make the present article to be comprehensive; we mention applications only when it is necessary for illustrating a new idea and show how effective it is. As a result, only a few applications are discussed in the rest of the article. We refer the readers who are interested in various applications, as well as other things that we omit in the present article, to the review articles mentioned above.

To make this article usable, we separate *how* from *why*, i.e., the description of the algorithms from their mathematical derivations. Therefore, those who need a quick start, rather than knowing why an algorithm gives correct results, may skip the theoretical part (§2) and immediately go to §3

*E-mail: nao@phys.metro-u.ac.jp

[†]E-mail: harada@acs.i.kyoto-u.ac.jp

entitled “Numerical Recipe”. This section is almost self-contained and only the minimal references are made to the other parts of the article.

2. Theory

In this section, we present a few algorithms, roughly in chronological order. Descriptions will constitute a mathematical justification of the algorithms’ validity, though they do not follow the conventional theory-and-proof format. Some examples are also presented for illustrating relevant ideas and the efficiency of the resulting algorithms. While this style would make it easier to follow the logic that establishes the validity of the algorithms, it may make it hard to find the precise and detailed definitions of the procedures. Therefore, we add another section following the present one, in which we concentrate on describing the procedures precisely.

2.1 Cluster update

The improvements accomplished on the quantum Monte Carlo simulation during the last decade was largely due to the global update, in which configurations are updated in units of some non-local clusters. Such a method of updating is inspired by the Swendsen–Wang (SW) algorithm¹⁴⁾ for the Ising model. In fact, it is not merely inspired but has the same mathematical back-ground as the SW algorithm. This is manifested by the fact that the loop algorithm proposed by Evertz *et al.* for the $s = 1/2$ XXZ quantum spin model depends continuously on the anisotropy and in the limit of a large uni-axial anisotropy (i.e., the Ising limit), the algorithm converges to something equivalent to the SW algorithm.¹⁵⁾ In this sense, the loop algorithm for the quantum spin systems can be considered as a generalization of the SW algorithm. The same is true for the single-cluster variant of the cluster algorithm by Wolff;¹⁶⁾ the single-cluster variant of the loop algorithm for the quantum Monte Carlo can be derived from the Wolff algorithm in exactly the same way as we can derive its multiple-cluster variant from the SW algorithm. In what follows, we consider the multiple-cluster variant, when we have to choose one, while the generalization to the single-cluster variant is straightforward in many cases.

We start with describing the SW algorithm to clarify the mathematical basis underlying almost all the algorithms discussed in the present article. Simply stated, the SW algorithm and other algorithms presented below are special cases of the dual Monte Carlo algorithm.^{17,18)} In a dual Monte Carlo algorithm, the Markov process alternates between two configuration spaces; the space of the original configurations that naturally arise from the model (such as the spin configurations in the Ising model and the world-line configurations in the quantum lattice models) and the space of the configurations of auxiliary variables. It is up to us to define the auxiliary variables and the resulting algorithm depends on the definition. In what follows, we denote the original configuration by S and the auxiliary one by G . (We denote the size of spins by s instead of S , to avoid confusion.) Once the auxiliary variables are defined, a stochastic process is characterized by the transition probabilities $T(G|S)$ and $T(S|G)$, the former being of generating G with S given and the latter of generating S with G given. The

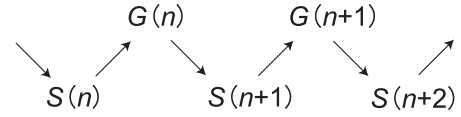


Fig. 1. A schematic process of a dual Monte Carlo. The arrow indicates dependencies. It should be noted that the new state $S(n+1)$ depends on the previous one $S(n)$ only through $G(n)$. The same is true for $G(n+1)$.

stochastic process as depicted in Fig. 1 yields the limiting distribution

$$\lim_{n \rightarrow \infty} P_n(S) \equiv \lim_{n \rightarrow \infty} \text{Prob } S(n) = S \propto W(S)$$

provided that we define the transition probabilities so that the ergodicity and the extended detailed balance

$$T(G|S)W(S) = T(S|G)W(G) \quad (1)$$

may hold. Here $W(S)$ ($W(G)$) is an arbitrary positive function of S (G). It is specified for each individual case as we see below.

Swendsen and Wang chose the auxiliary variable G_u to be a one-bit (i.e., 0-or-1) variable defined on each pair u of interacting spins. The auxiliary configuration G is defined as the set of all such variables: $G \equiv (G_1, G_2, \dots, G_{N_B})$, where N_B is the total number of the nearest-neighbor pairs. It is very cumbersome to describe the procedure in terms only of variables, with no picture, though in the end such a description is needed for coding. We, therefore, resort to visual means whenever a suitable visualization is available. The local unit u on which we define an auxiliary variable G_u is not necessarily a pair of sites. In addition, G_u is not necessarily a 0/1 variable either. Therefore, the visualization varies depending on the problem. In the case of the Ising model, however, the visualization is done most naturally by representing an up-spin and a down-spin by an open and a solid circle, respectively, and an auxiliary variable by the presence (corresponding to $G_u = 1$) or the absence (corresponding to $G_u = 0$) of a solid line connecting two neighboring circles. (See Fig. 2.)

Swendsen and Wang¹⁴⁾ proposed the following procedure of updating the spin configuration. For a given configuration, (i) assign $G_u = 0$ or 1 probabilistically to each u , (ii) identify connected spins to form clusters, and for each cluster (iii) assign a common value ± 1 to all the spin variables on it. In the graphical terms, this yields the following (i) connect nearest-neighbor circles with some probability, (ii) recognize the connected sets of circles, and for each connected set, (iii) change the color of all circles simultaneously with a

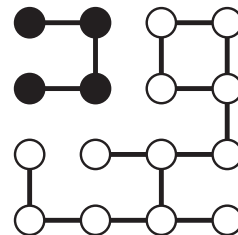


Fig. 2. A spin configuration S and a matching graph G of the Ising model. The spin configuration is represented by the open and solid circles whereas the graph consists of the lines connecting circles.

certain probability. The step (iii) is often called a ‘cluster-flip’.

In the following, we show that this stochastic process produces the distribution of spin configuration proportional to the Boltzmann weight

$$W(S) = \exp\left(K \sum_{\langle ij \rangle} S_i S_j\right) = \prod_u w(S_u), \quad (2)$$

where $u \equiv (i, j)$, $S_u = (S_i, S_j)$, and $w(S_u) \equiv \exp(KS_i S_j)$. Following Fortuin and Kasteleyn,^{19,20)} we decompose the local Boltzmann weight as

$$w(S_u) = \sum_{G_u} w(S_u, G_u). \quad (3)$$

The function $w(S_u, G_u)$ is defined as

$$w(S_u, G_u) = \begin{cases} e^{-K} & (G_u = 0) \\ e^K - e^{-K} & (S_i = S_j \text{ and } G_u = 1) \\ 0 & (S_i \neq S_j \text{ and } G_u = 1) \end{cases} \quad (4)$$

It is easy to verify that this definition satisfies (3). Using (3), eq. (2) can be formally rewritten as

$$W(S) = \sum_G W(S, G), \quad (5)$$

where

$$W(S, G) \equiv \prod_u w(S_u, G_u). \quad (6)$$

Once the target weight $W(S)$ is written in the form of (5) with (6) and (3), we can in general satisfy the detailed balance (1) by defining the transition probabilities as

$$T(G|S) \equiv \frac{W(S, G)}{W(S)}, \quad T(S|G) \equiv \frac{W(S, G)}{W(G)}, \quad (7)$$

where $W(G) \equiv \sum_S W(S, G)$. As stated above, a Markov process with these transition probabilities yields the target distribution $W(S)$.

For the graph-assignment probability $T(G|S)$, we can rewrite (7) using (2) and (3) as $T(G|S) = \prod_u t(G_u|S_u)$ with $t(G_u|S_u)$ being

$$t(G_u|S_u) \equiv w(S_u, G_u)/w(S_u). \quad (8)$$

Since $T(G|S)$ is factorized into the local factors $t(G_u|S_u)$, the graph assignment can be done locally; we can assign a graph element to each local unit independently with the probability $t(G_u|S_u)$. For the Ising model, in particular, this transition probability is realized by the well-known Swendsen–Wang procedure, i.e., connecting each nearest-neighbor pair of parallel spins with the probability $1 - e^{-2K}$ and leaving them unconnected otherwise.

For the spin-updating probability $T(S|G)$, we similarly obtain

$$T(S|G) = 2^{-N_C(G)} \Delta(S, G), \quad (9)$$

where $N_C(G)$ is the number of connected clusters in G and $\Delta(S, G)$ is the function that takes a value 1 if and only if all spins in each cluster are aligned in the same direction in S . Therefore, this transition probability is realized by the step (iii) in Swendsen and Wang’s procedure. Thus the validity of the procedure has been proved.

2.2 Path integral and quantum Monte Carlo

The description of the SW algorithm given in the previous subsection is quite general; the only model-specific part is the first equality in (2) and (4). In fact, the loop algorithm for the quantum Monte Carlo can be regarded as a special case of this framework. As long as the target weight $W(S)$ can be expressed as (5) with some $W(S, G)$, the transition probabilities (7) constitute a valid algorithm (provided, of course, that the ergodicity holds), regardless of the model we consider. Therefore, the only that we have to do is to specify ingredients such as S , G , and $W(S, G)$, which we do in this subsection.

There are two ways of introducing S ; one by the path integral¹⁾ and the other by the high-temperature series expansion.²¹⁾ While the latter leads to a discrete-time algorithm with an exponentially small systematic error, the former is simpler to describe. In addition, both the representations reduce to the same algorithm in the continuous-time limit. Therefore, we describe the framework starting from the path-integral representation. The formulation based on the high-temperature series expansion is discussed in §2.4.

For the derivation of the algorithm presented below, it is often useful to consider the path integral in the discretized imaginary time (though the discretization is not needed in the final algorithm). Such an expression can be obtained as follows. First we consider the identity,

$$Z = \sum_{\psi} \left\langle \psi \left| \lim_{L \rightarrow \infty} \prod_{k=1}^L \left(1 - \frac{\beta}{L} \mathcal{H} \right) \right| \psi \right\rangle. \quad (10)$$

In particular, when the Hamiltonian is a sum of M terms,

$$\mathcal{H} = \sum_{b=1}^M \mathcal{H}_b, \quad (11)$$

then, (10) leads to

$$Z = \sum_{\psi} \left\langle \psi \left| \lim_{L \rightarrow \infty} \prod_{k=1}^L \prod_{b=1}^M \left(1 - \frac{\beta}{L} \mathcal{H}_b \right) \right| \psi \right\rangle.$$

Here, the summation is taken over some complete orthonormal basis $\{\psi\}$ of the Hilbert space. Inserting $1 = \sum_{\psi} |\psi\rangle\langle\psi|$ between two adjacent factors, we obtain

$$Z = \lim_{L \rightarrow \infty} \sum_{\{\psi_b(k)\}} \prod_{k=1}^L \prod_{b=1}^M \langle \psi_{b+1}(k) | (1 - (\Delta\tau) \mathcal{H}_b) | \psi_b(k) \rangle, \quad (12)$$

where $\Delta\tau \equiv \beta/L$. For simplifying the notation, we denote (b, k) as u , $\psi_b(k)$ as ψ_u , $\psi_{b+1}(k)$ as ψ'_u , \mathcal{H}_b as \mathcal{H}_u , and $\{\psi_b(k)\}$ as S . It follows that

$$Z = \lim_{L \rightarrow \infty} \sum_S W_L(S), \quad (13)$$

$$W_L(S) \equiv \prod_u w(S_u),$$

$$w(S_u) \equiv \langle \psi'_u | (1 - (\Delta\tau) \mathcal{H}_u) | \psi_u \rangle. \quad (14)$$

Thus the partition function is expressed as the sum of a weight that is a function of a space-time configuration. In the case of the particle systems, with the basis that diagonalizes the local number operators, the space-time configuration is called a world-line configuration, since the configuration is visualized by trajectories of particles in the space-time. The configuration for spin models is also called the world-line

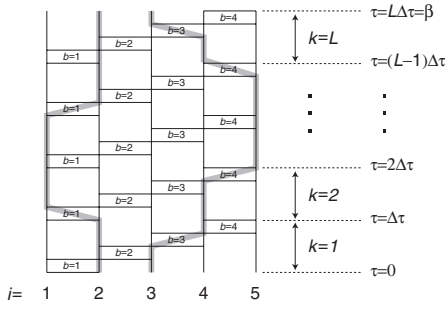


Fig. 3. A visualization of the path integral of a five-site system with discrete imaginary time. The world-lines are represented by two thick gray lines.

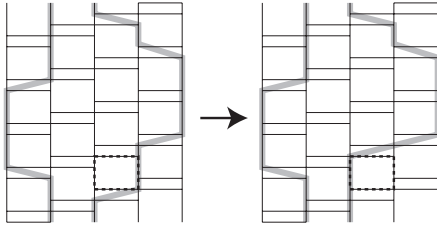


Fig. 4. One step in the local update algorithm. The squares such as the one drawn with dashed lines are the units of the update. At every step, one of the squares is chosen at random. The flip of the chosen square is accepted with a probability that depends on the weights of the configurations before and after the flip.

configuration by regarding up-spins and down-spins as particles and holes respectively. An example of the world-line configuration is shown in Fig. 3. The whole system consists of L layers whereas each layer contains M sub-layers. (The number L is called the Trotter number.) The “height” of each layer is $\Delta\tau$ and the height of the whole system is always β regardless of L . Every \mathcal{H}_b has its representative in each layer, i.e., a unit u called a *plaquette*. Each of M sub-layers contains a plaquette.

In early world-line Monte Carlo algorithms, updates of a configuration were done in many steps,^{1,22,23)} each being a local update that modifies only a small part of the system. Before the loop algorithm, the unit of the local update was a square whose spatial dimension equals the lattice spacing and the temporal dimension the discretization unit of time. The square is shown in Fig. 4 together with the world-line configurations before and after the update. Because of the local nature of the updating unit, the algorithm exhibits a severe slowing-down. It happens when we approach a critical point or zero temperature. This can be intuitively understood as the discrepancy of the physical correlation length and the spatial scale of the updating unit. Another slowing-down, pointed out by Wiesler,²⁴⁾ when the temporal scale of the system (i.e., the inverse energy gap) largely differs from the temporal scale of the updating unit. The situation occurs when one decreases the discretization unit of the imaginary time in order to reduce the systematic error due to the discretization. It was proposed²⁵⁾ that this slowing-down can be removed by applying the loop method only to the temporal direction. The algorithms discussed below solve both types of slowing-down in many cases of interest.

2.3 Loop update

A loop algorithm for a quantum system can be constructed in a similar fashion as the Swendsen–Wang algorithm mentioned in §2.1. That is, by introducing additional variables $G_u = G(b, k) = 0, 1$, we can rewrite $W_L(S)$ in (14) as

$$W_L(S) \equiv \prod_u \sum_{G_u=0,1} \langle \psi'_u | (-\Delta\tau) \mathcal{H}_u^{G_u} | \psi_u \rangle$$

$$= \sum_G (\Delta\tau)^{n(G)} \prod_u \langle \psi'_u | (-\mathcal{H}_u)^{G_u} | \psi_u \rangle \quad (15)$$

$$= \sum_G W_L(S, G), \quad (16)$$

where $G \equiv \{G_u\}$, $n(G) \equiv \sum_u G_u$, and

$$W_L(S, G) \equiv \prod_u w(S_u, G_u), \quad (17)$$

$$w(S_u, G_u) \equiv \langle \psi'_u | (-\Delta\tau) \mathcal{H}_u^{G_u} | \psi_u \rangle. \quad (18)$$

Since these expressions (16) and (17) have the same form as (5) and (6), we can apply the prescription presented in §2.1 for defining the transition probabilities $T(G|S)$ and $T(S|G)$ through (7), thereby constructing an algorithm of simulating a target distribution $W_L(S)$. Since $W_L(S)$ is an approximation to $W(S)$, such an algorithm can be used as an ‘approximate’ algorithm of simulating the distribution $W(S)$. (In §2.5, we see that this ‘approximation’ can be made exact.)

In order to complete the definition of the algorithm, we have to specify the Hamiltonian, what orthonormal set we use, and how we decompose it in (11). In what follows, we do these and examine what procedure corresponds to the resulting transition probabilities.

First of all, we specify the meaning of the decomposition (11) of the Hamiltonian. We start with the graphical decomposition^{5,8,17,26–29)} of the pair Hamiltonian \mathcal{H}_{ij} :

$$\mathcal{H}_{ij} = \sum_g \mathcal{H}_{ij}(g), \quad \mathcal{H}_{ij}(g) = -a(g) \hat{\Delta}_{ij}(g), \quad (19)$$

where g specifies a type of a graph element, $a(g)$ is some positive constant, and $\hat{\Delta}_{ij}(g)$ is an operator whose matrix elements are 0 or 1. As shown in Table I, a $\hat{\Delta}$ -operator corresponds to a graph element with two types of lines, each representing a condition for making the matrix element 1. A solid line connecting two spins represents the condition that the two must be parallel whereas a dashed line requires that the two be anti-parallel.

In the case of the $s = 1/2$ anti-ferromagnetic Heisenberg model,

$$\mathcal{H}_{ij} = -J(S_i^x S_j^x + S_i^y S_j^y - (S_i^z S_j^z - 1/4)),$$

which we use in the following as an example, the summation in eq. (19) contains only one term:

$$\mathcal{H}_{ij} = -\frac{J}{2} \hat{\Delta}_{ij}(g_H), \quad (20)$$

where g_H is the graph element shown in the third graph from the top in Table I. Here, for the orthonormal complete set $\{|\psi\rangle$ in (10)], we have chosen the set of the simultaneous eigenstates of the z -components of all spin operators, as we do in most of the present article. The operator $\hat{\Delta}_{ij}(g_H)$ is explicitly defined in terms of the matrix elements as

Table I. The graph elements, and the matrix elements of the delta operators $\hat{\Delta}_{ij}(g)$ corresponding to each element. The base vectors of the two-spin Hilbert space are $|\frac{1}{2}, \frac{1}{2}\rangle$, $|\frac{1}{2}, -\frac{1}{2}\rangle$, $|\frac{1}{2}, \frac{1}{2}\rangle$, and $|\frac{1}{2}, -\frac{1}{2}\rangle$. To emphasize the difference in the constraints indicated by the lines, dashed lines are used when the connected spins must be anti-parallel to each other whereas solid lines connect parallel spins.

Symbol	Graph	$\langle \sigma'_i \sigma'_j \hat{\Delta}_{ij}(g) \sigma_i \sigma_j \rangle$
g_I		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
g_C		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
g_H		$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$
g_{CB}		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
g_{HB}		$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

$$\langle \sigma'_i \sigma'_j | \hat{\Delta}_{ij}(g_H) | \sigma_i \sigma_j \rangle = \delta_{\sigma_i, -\sigma_j} \delta_{\sigma'_i, -\sigma'_j}. \quad (21)$$

Equation (19) results in the decomposition of the total Hamiltonian as $\mathcal{H} = \sum_{(ij)} \sum_g \mathcal{H}_{ij}(g)$. Thus the Hamiltonian is decomposed in the form of (11) by identifying b and $((i, j), g)$, i.e., u and $((i, j), g, k)$. Then, the algorithm follows from the prescription given in §2.1.

For example, the graph assignment probability $t(G|S)$ in (8) becomes

$$t(1|S_u) = (\Delta\tau) \times a(g) \langle \psi'_u | \hat{\Delta}_{ij}(g) | \psi_u \rangle \quad (22)$$

for S_u being a non-kink, i.e., $\psi'_u = \psi_u$. On the other hand, if S_u is a kink, or $\psi'_u \neq \psi_u$, then $t(1|S_u) = 1$. Thus, for the case of $s = 1/2$ antiferromagnetic Heisenberg model, the probability is

$$t(1|S_u) = \begin{cases} 1 & (S_u \text{ is a kink.}) \\ \frac{J}{2}(\Delta\tau) \left(S_u = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \text{ or } \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \right) & \\ 0 & (\text{otherwise}) \end{cases} \quad (23)$$

Choosing the value 1 for G_u means that we place a graph of the type g on the plaquette u . For all the plaquettes with the value 0, we assign the 'identity', or 'trivial' graph (the top row in Table I) representing the identity operator. In what follows, we call a plaquette on which a non-trivial graph-element is assigned a *vertex*.

The procedure that realizes the probability $T(S|G)$ is the same as that in the SW algorithm; first identify the points connected by the lines and then flip each cluster with probability $1/2$. (When applied to the isotropic Heisenberg

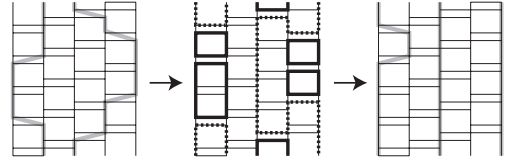


Fig. 5. A loop update. The decomposition into loops (from the left to the middle) and the flipping of the loops (from the middle to the right) are shown for the $s = 1/2$ antiferromagnetic Heisenberg model in one dimension. The only non-trivial graph elements are the 'horizontal' ones, g_H , in Table I. The loops flipped in the transition from the middle diagram to the right are indicated by dashed lines in the middle diagram whereas other loops are drawn with solid lines.

model, the graph elements g_{CB} or g_{HB} in Table I do not appear, and the resulting clusters are simple loops. The name of the algorithm follows from this fact. In the present paper, we use the name even for the cases where clusters are not simple loops.) A 'space-time' point (i, k) plays the same role as a site i in the SW algorithm. An example of one step in the loop algorithm is depicted in Fig. 5 for the $s = 1/2$ antiferromagnetic Heisenberg model in one dimension.

It is useful to see what kind of loops and clusters are formed^{2,26,28} in various other cases. We consider the XYZ model described by the Hamiltonian

$$\mathcal{H} = \sum_{(ij)} \mathcal{H}_{ij}, \quad \mathcal{H}_{ij} = c - J_x S_i^x S_j^x - J_y S_i^y S_j^y - J_z S_i^z S_j^z, \quad (24)$$

where c is a constant. As for the orthonormal complete set, we take the set of the simultaneous eigenstates of the z -components of all spin operators as above. Therefore, a basis vector ψ can be uniquely specified by the eigenvalues of the N operators, $S_1^z, S_2^z, \dots, S_N^z$. In this representation, when J_x and/or J_y are negative, the off-diagonal matrix elements of $-\mathcal{H}$ may be negative. For the bipartite lattices, however, the number of the negative matrix elements in the whole configuration is even, which makes the weight $W(S)$ always positive. Another way of seeing this³⁰ is to divide the whole lattice into two sub-lattices, A and B, so that a site on the sub-lattice A is surrounded by sites on the sub-lattice B, and rotate the spins on the sub-lattice B. For example, when $J_x = J_y < 0$, we rotate spins on the sub-lattice B around the z -axis, so that $S_i^x \rightarrow -S_i^x$ and $S_i^y \rightarrow -S_i^y$. This rotation makes all the off-diagonal elements positive. In what follows, therefore, we consider the cases with no negative-sign problem and assume that all the off-diagonal matrix elements of $-\mathcal{H}_{ij}$ are non-negative.

Then, the pair Hamiltonian \mathcal{H}_{ij} , eq. (24), with $J_x = J_y > 0$ can be expressed with two graph elements. We can see this in the following graphical decomposition analogous to (20),

$$-\mathcal{H}_{ij} = \begin{cases} \frac{J_x}{2} \hat{\Delta}_{ij}(g_C) + \frac{J_z - J_x}{2} \hat{\Delta}_{ij}(g_{CB}) & \text{(I)} \\ \frac{J_x + J_z}{4} \hat{\Delta}_{ij}(g_C) + \frac{J_x - J_z}{4} \hat{\Delta}_{ij}(g_H) & \text{(II)} \\ \frac{J_x}{2} \hat{\Delta}_{ij}(g_H) + \frac{-J_z - J_x}{2} \hat{\Delta}_{ij}(g_{HB}) & \text{(III)} \end{cases}$$

where the constant c in (24) has been chosen so that the matrix elements are positive in each form. Since we need an

expression of the form (19) with positive $a(g)$, only one of these three expressions can be used for a particular set of the values of J_x and J_z . The form (I) can be used for the easy-axis ferromagnetic model ($0 < J_x = J_y \leq J_z$), the form (II) for the easy-plane model ($0 \leq |J_z| < J_x = J_y$), and the form (III) for the easy-axis antiferromagnetic model ($0 < J_x = J_y \leq -J_z$). The five types of graph elements shown in Table I are sufficient for expressing the pair Hamiltonian in all the three cases.

The second and the fourth elements in Table I are required for expressing the pair Hamiltonian of the easy-axis ferromagnetic model [the case (I)]. The fourth graph-element binds all the four spins $\sigma_i, \sigma_j, \sigma'_i$, and σ'_j . Therefore, in this case, a resulting cluster of spins that is to be flipped simultaneously is not generally a single loop but a number of loops bound together. On the other hand, the second and the third graph elements are sufficient for expressing the pair Hamiltonian of the easy-plane model [the case (II)], such as the XY model. In either graph element, the four spins are bound only pair-wise. Therefore, a graph G consists of loops in this case. For the easy-axis antiferromagnetic model [the case (III)], the graph elements required are the third and the last. Therefore, in this case, a cluster is not a single loop, in general, similar to the case (I). For more details of the algorithm and the case with a lower symmetry (i.e., the XYZ model), see §3. (A sample program may be found at a web-site.³¹⁾)

Many applications of the loop updating method have been done. Here, we only show the result for the quantum $s = 1/2$ XY model in two dimensions,^{32,33)} which clearly demonstrates the utility of the loop algorithm.

It is well-known that the helicity modulus exhibits the universal jump at the Kosterlitz–Thouless type phase transition.³⁴⁾ The system-size dependence of the quantity near the critical point is also predicted theoretically. In the quantum spin model, such as the $s = 1/2$ XY model, the helicity modulus is related to the fluctuation in the total winding number of the world-lines by $\Upsilon = (T/2)\langle \mathbf{W}^2 \rangle$,³⁵⁾ where $\mathbf{W} \equiv (W_x, W_y)$ with W_x (W_y) being the total winding number in the x (y) direction. Therefore, we can estimate the critical temperature accurately by measuring the winding number. In Fig. 6, the raw data of the helicity modulus is shown. We can see the universal jump even in the raw data

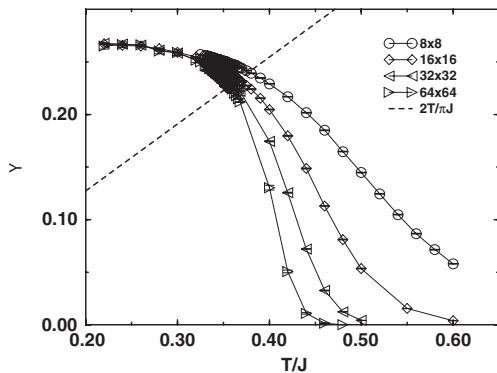


Fig. 6. The helicity modulus (or the superfluid density) $\Upsilon = (T/2)\langle \mathbf{W}^2 \rangle$ as a function of the temperature. The universal jump is expected at the point where $\Upsilon = 2T/\pi J$. The error bars are drawn, but most of them are too small to be recognized. (Adopted from Harada and Kawashima.³²⁾)

and obtain a rough estimate of the transition temperature $T_{KT} \sim 0.35$. We can obtain a much more precise estimate for the critical temperature by fitting the data to the theoretically predicted form of the size dependence. The best estimate of the transition temperature has been obtained in this way. Note that it is difficult to estimate the transition temperature by means of a conventional world-line quantum Monte Carlo method, such as the one shown in Fig. 4. This is because the auto-correlation time becomes too long as we approach the critical temperature from above. Unlike the ordinary phase transition, it is increasingly more difficult to equilibrate the system even after passing the transition temperature since the system remains critical in the whole low-temperature region. Therefore, with conventional methods, we can obtain reliable estimates of various quantities only in the high-temperature region. Another reason that makes difficult the estimation by the local update is that it does not yield an ergodic algorithm; the winding number of world-lines is not allowed to vary. Therefore, one must observe other quantities, for which the size dependence is known less precisely, or introduce some additional global updates for making the winding number vary. The latter was done in an early simulations,³⁶⁾ and later in a simulation of a bosonic system.³⁷⁾ However, these additional global flips tend to form bottlenecks in the configuration space, slowing down the whole simulation.

2.4 Formulation based on the series expansion

So far we have been using the approximation of the imaginary-time discretization. While we can use the finite- L expressions for constructing an approximate algorithm in order to obtain all the results that we need, the results would come with a systematic error due to the imaginary-time discretization. Therefore, we would have to do an extrapolation to get the final result free from this systematic error. However, there are two ways to get rid of the discretization error. In the first method, which we discuss in §2.5, we perform the extrapolation to the continuous imaginary time in the algorithm, not in the numerical results. In other words, there exists a computational procedure that operates directly on continuous degrees of freedom (on the floating-point variables, to be precise). In this method there is no discretization error. In the present section, on the other hand, we present a method with discretized degrees of freedom that yields an algorithm with a much smaller error (negligible for most purposes) than the naive discretized-time method.

The formulation is based on the high-temperature series expansion, and is originated in Handscomb's method.²¹⁾ It was later elaborated by Sandvik and coworkers.^{6,38,39)} It starts from the expansion of the partition function,

$$Z = \lim_{L \rightarrow \infty} Z_L,$$

where

$$Z_L \equiv \sum_{n=0}^L \frac{\beta^n}{n!} \text{Tr}(-\mathcal{H})^n.$$

Then, we visualize each term by considering L “boxes” and put n “marbles”, each corresponds to $-\mathcal{H}$, into these boxes. Each box can contain one marble at most. Therefore, there are $\binom{L}{n}$ distinct pictures corresponding to the same term.

Thus we have

$$Z_L = \sum_{\{\gamma_k\}} \beta^{\sum_k \gamma_k} \frac{(L-n)!}{L!} \text{Tr} \prod_{k=1}^L (-\mathcal{H})^{\gamma_k},$$

where $\gamma_k = 0, 1$ represents a filled or an empty box, respectively. When the Hamiltonian is decomposed into a product of local factors as in (11), we can rewrite the above as

$$Z_L = \sum_G \beta^{n(G)} \frac{(L-n(G))!}{L!} \text{Tr} \prod_u (-\mathcal{H}_u)^{G_u},$$

where $u \equiv (b, k)$, $\mathcal{H}_u \equiv \mathcal{H}_b$, $G_u = 0, 1$, $G \equiv \{G_u\}$ and $n(G) \equiv \sum_u G_u$. The summation is restricted to the graphs G such that $\sum_b G_{(b,k)} = \gamma_k \leq 1$. As we did in the path-integral formulation to obtain (12), we can insert the identity operators expanded in the orthonormal complete set to obtain,

$$Z_L = \sum_S W_L(S),$$

$$W_L(S) = \sum_G \beta^{n(G)} \frac{(L-n(G))!}{L!} \prod_u \langle \psi'_u | (-\mathcal{H}_u)^{G_u} | \psi_u \rangle. \quad (25)$$

Apart from the factor $(L-n(G))!/L!$ and the restriction on the summation, eq. (25) looks similar to (15). In fact, when $L \gg 1$, the difference in the factor is small because $\beta^n (L-n)!/L!$ is approximately equal to $(\beta/L)^n$. In addition, for large L , the difference due to the restriction produces only a small difference, because the typical value of $n(G)$ in an actual simulation should not depend on L (as long as L is large enough) and therefore having more than one vertices in the same layer is an increasingly rare event for large L even if such an event is allowed. Therefore, eq. (25) derived from the series expansion is approximately equal to (15) derived from the path-integral formulation for the same L , and they become identical in the large- L limit. This means that the algorithms that follow are similar for a finite L and identical for the infinite L . The important difference, however, is that the discretization error for a finite L is exponentially small for (25) whereas that for (15) vanishes only algebraically. Therefore, in the path-integral formulation, we need to take the infinite- L limit whereas in the series-expansion formulation, it is not necessary as long as L is large enough.

All the algorithms, such as the loop algorithm and the directed-loop algorithm discussed below, can be derived from the series-expansion formulation as well as the path-integral formulation, and in most cases the mapping from an algorithm derived from the latter to the one derived from the former is straightforward. While a “discretized-time” algorithm based on the series-expansion representation can be advantageous for efficient implementation (because only integral variables are needed there), we discuss in what follows a number of algorithms using the path-integral formulation to avoid the factor $(L-n)!/L!$ appearing in the expressions.

2.5 Continuous-imaginary-time limit

The loop algorithm is useful not only in speeding up the simulation but also in taking the $L \rightarrow \infty$ limit in the algorithm,⁷⁾ which makes the algorithm free from the systematic error due to the discretization of the imaginary time.

For a large L , the target distribution $W_L(S)$ of the spin configuration mimics the distribution $W(S) \equiv \lim_{L \rightarrow \infty} W_L(S)$. It means that if we look at a configuration with a poor resolution in the imaginary time, we cannot tell whether the configuration is generated with the weight $W(S)$ or $W_L(S)$. Therefore, when we have a finite correlation time ξ_τ in the target distribution $W(S)$, we do not have a kink at which a state changes, i.e., $\psi_{b+1}(k) \neq \psi_b(k)$ in (typically) $\xi_\tau/\Delta\tau$ consecutive layers. Let us consider an imaginary-time interval of the length I that includes many layers with no kink. As can be seen in (22), we assign a graph element of type g with probability $(\Delta\tau)a(g)$ to a unit u when S_u makes the matrix element of $\hat{\Delta}(g)$ unity. Since there are $I/\Delta\tau$ layers in this interval, the probability of assigning n graph elements of type g to this interval is given by

$$\binom{I/(\Delta\tau)}{n} ((\Delta\tau)a(g))^n (1 - (\Delta\tau)a(g))^{I/(\Delta\tau)-n}.$$

In the continuous-time limit $L \rightarrow \infty$ this reduces to

$$\frac{1}{n!} (Ia(g))^n e^{-Ia(g)}.$$

This is nothing but the Poisson distribution with mean $Ia(g)$. Therefore, instead of repeating the graph assignment procedure for all the plaquettes, we can generate a number n with the Poisson distribution with mean $Ia(g)$, and choose n points from I uniform-randomly. The result would be statistically the same as what we would obtain from the discrete-time procedure described in §2.3 (with extremely large L).

Another advantage of the continuous-imaginary-time algorithms is that we do not have to deal with the fine structure of the ‘space-time’. For example, the time ordering of the plaquettes with different b in each layer (such as the one shown in Fig. 3) can be arbitrary, because in the continuous-time limit, individual plaquettes do not appear and therefore the order of them does not matter at all.

Since we consider the $\Delta\tau \rightarrow 0$ limit in what follows, the height of a plaquette is zero, i.e., a plaquette in the discrete time corresponds to a horizontal line. We call the horizontal line (plaquette) on which a non-trivial graph is placed a *vertex*. The four corners of a plaquette are called *legs*. (See Fig. 21 for the names of various objects.)

It may be helpful to summarize here the procedure of one Monte Carlo step with the continuous-imaginary-time loop algorithm. Starting from an arbitrary pair of S and G that match each other, first we remove all the vertices (i.e., graph elements) at which there is no kink. Next, for each pair of the nearest neighbor sites (ij) , we decompose the interval $(0, \beta)$ into uniform intervals (UI). [Here, a UI for a pair of sites (i, j) , is an imaginary-time interval delimited by two kinks that involves one or both of i and j . (See Fig. 7.)] For each UI, which we denote as I , and for each kind of graph elements, which we denoted as g , we generate an integer n with the Poisson distribution of mean $Ia(g)$, and place n graph elements of the type g uniform-randomly in I . When this is done for all types of graphs, all the uniform intervals, and all the nearest neighbor pairs, we identify loops, or clusters. Finally we flip each loop (cluster) with probability $1/2$.

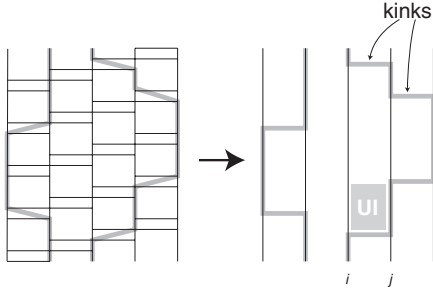


Fig. 7. The correspondence between the world-line configuration in the discrete time and that in the continuous time. When a loop algorithm is used, a real-time “animation” on the computer screen would look the same for both the representations as long as the imaginary time step $\Delta\tau$ in the discrete representation is small enough. A uniform interval (UI) is indicated as a lightly shaded region.

2.6 Large spins

The generalization of the loop algorithm to larger (i.e., higher) spins can be done by replacing each spin operator by the sum of $2s$ $s = 1/2$ spins.⁵⁾ That is, we replace the spin operators in (24) as

$$S_i^\alpha \Rightarrow \tilde{S}_i^\alpha \equiv \sum_{\mu=1}^{2s} \sigma_{i,\mu}^\alpha \quad (\alpha = x, y, z), \quad (26)$$

where each spin operator σ_i carries $s = 1/2$. Accordingly, a basis vector is specified by eigenvalues of the $2sN$ operators $\{\sigma_{i,\mu}^z\}$ ($i = 1, 2, \dots, N$ and $\mu = 1, 2, \dots, 2s$). In what follows, we identify the label $\psi_b(k)$ with a set of $2sN$ variables $\{\sigma_{i,\mu}\}$, where $\sigma_{i,\mu} = \pm \frac{1}{2}$ denotes an eigenvalue of $\sigma_{i,\mu}^z$. The new Hilbert space spanned by these vectors has the dimension 2^{2sN} , somewhat larger than the original one which is spanned by only $(2s+1)^N$ basis vectors. Therefore, we have to eliminate many states in order to obtain the correct partition function of the original model. This can be achieved by introducing the projection operator \hat{P} ,^{40,41)} i.e.,

$$Z = \text{Tr} e^{-\beta \mathcal{H}(\{S_i\})} = \text{Tr} \hat{P} e^{-\beta \mathcal{H}(\{\tilde{S}_i\})}. \quad (27)$$

The projection operator \hat{P} eliminates all the states that do not have corresponding states in the original problem, such as the singlet states in the $s = 1$ problem.

When the original spins are split into $s = 1/2$ spins, the pair Hamiltonian \mathcal{H}_{ij} can be written as

$$\mathcal{H}_{ij} = \sum_{\mu=1}^{2s} \sum_{\nu=1}^{2s} \mathcal{H}_{i\mu,j\nu},$$

where $\mathcal{H}_{i\mu,j\nu}$ is the pair Hamiltonian that can be obtained by replacing S_i and S_j by $\sigma_{i\mu}$ and $\sigma_{j\nu}$, respectively, in the definition (24). The pair Hamiltonian $\mathcal{H}_{i\mu,j\nu}$ is nothing but the pair Hamiltonian of the $s = 1/2$ model discussed above. It is thus obvious that we can apply the general prescription described in §2.2, §2.3 and §2.5 simply by re-interpreting b as $((i, j, \mu, \nu), g)$ instead of $((i, j), g)$. As a result we have $2s$ vertical lines for each site i as illustrated in Fig. 8. The graph-assignment procedure must be repeated $(2s)^2$ times corresponding to $(2s)^2$ pairs of the indices μ and ν . The procedure is otherwise identical to the one for the $s = 1/2$ model. The types of the graphs and the graph-assignment density are exactly the same as the corresponding $s = 1/2$ model.

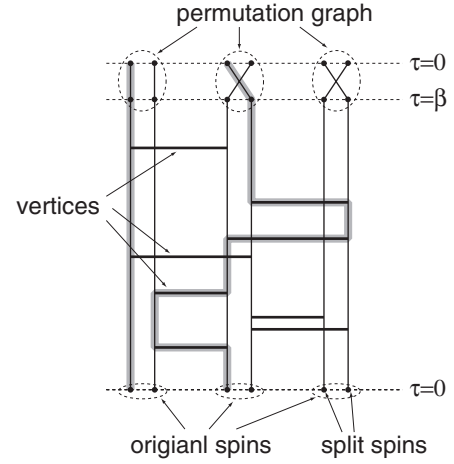


Fig. 8. The split-spin representation of a world-line configuration for the $s = 1$ quantum spin system.

We can handle the projection operator \hat{P} through a graphical decomposition as we do for the Hamiltonian. It should be noted that the operator \hat{P} projects the extended Hilbert space onto the sub-space that is isomorphic to the original Hilbert space. This sub-space consists of the simultaneous eigenstates of the Casimir operators $(S_i)^2$. The states must be symmetric in the μ space for each site. In other words, the state is invariant under any permutation of the split spins $(\sigma_{i,1}, \sigma_{i,2}, \dots, \sigma_{i,2s})$ for each i . Therefore, the projection operator is a product of local projection operators, and each local projection operator can be expressed as the sum of permutation operators;

$$\hat{P} = \prod_i \hat{P}_i, \quad \hat{P}_i = \frac{1}{(2s)!} \sum_{\pi} \hat{\Delta}_{\pi}.$$

Here, the summation is taken over the set of permutations among the split spin indices $\mu = 1, 2, \dots, 2s$, and $\hat{\Delta}_{\pi}$ is an operator that generates the permutation π . Specifically,

$$\langle \tilde{\psi}' | \hat{\Delta}_{\pi} | \tilde{\psi} \rangle \equiv \prod_{\mu=1}^{2s} \delta_{\sigma'_{i,\pi(\mu)}, \sigma_{i,\mu}},$$

where $\tilde{\psi} = (\sigma_{i,1}, \sigma_{i,2}, \dots, \sigma_{i,2s})$. This operator corresponds to a graph element that connects a point on the vertical line specified by (i, μ) to a point on the other line specified by $(i, \pi(\mu))$. This correspondence is similar to that of the operator $\hat{\Delta}_{ij}(g)$ and the graph element g as we see above. Therefore, in order to take the projection operator into account, we have only to include the following step in the updating procedure. That is, after assigning graph elements to the vertices, we assign a special graph element to the end points of the $2s$ vertical lines for each site i . Each graph element represents a particular permutation of $2s$ spins and connects the end points of the $2s$ vertical lines at $\tau = \beta$ to those at $\tau = 0$. The graph element is chosen with equal probability from the ones that are compatible to the current spin configuration (at $\tau = \beta$ and $\tau = 0$).

Among a number of applications of the split-spin algorithm described in this section, we briefly mention the calculation done by Todo and Kato,⁴¹⁾ since it is illustrative of the high efficiency of the algorithm. They computed the energy gap Δ between the ground state and the first excited state, and the correlation length ξ of the antiferromagnetic

Heisenberg model in one dimension at $T = 0$ for $s = 1, 2$ and 3. This system is known to exhibit the Haldane gap and is disordered even at zero temperature. The correlation length for $s = 1$ is about 6. Therefore, one can use the exact diagonalization for obtaining a rather accurate estimates of various quantities in this case. However, since the inverse gap and the correlation length diverge exponentially as the spin length increases, it is increasingly difficult to obtain accurate estimates for larger spins. They obtained the following estimates:

$$\begin{aligned} \xi &= 6.0153(3), \quad \Delta = 0.41048(6) \quad (s = 1), \\ \xi &= 49.49(1), \quad \Delta = 0.08917(4) \quad (s = 2), \\ \xi &= 637(1), \quad \Delta = 0.01002(3) \quad (s = 3). \end{aligned} \quad (28)$$

It is obvious from this result that we cannot compute these quantities with the exact diagonalization method for $s = 2$ and 3. To our knowledge, these numbers are very difficult to compute by any other methods than the ones described in this article. The estimates for $s = 2$, for example, are the best estimates known so far. For the estimates for $s = 3$, we are not aware of any other methods that can compute them.

2.7 Loop algorithms with non-binary loops

In some applications, it is advantageous to use non-binary loop variables. For example, let us consider the bilinear-biquadratic interaction model with $s = 1$,⁴²⁾

$$\mathcal{H} = J \sum_{\langle ij \rangle} ((\cos \theta) \mathbf{S}_i \cdot \mathbf{S}_j + (\sin \theta) (\mathbf{S}_i \cdot \mathbf{S}_j)^2). \quad (29)$$

Simulation of this model can be done with the split-spin method described in §2.6 with or without the coarse-graining in §2.11 and the details can be found in §3. (For an application, see Harada and Kawashima.⁴³⁾) In what follows, however, we consider an alternative algorithm which is particularly useful in dealing with special cases with higher symmetry.

The model (29) obviously has the SU(2) symmetry. At $\theta = \pm\pi/2$ and $\theta = \pm\pi/4$, however, it possesses a higher symmetry than is obvious from the definition. Here we consider the case $\theta = -\pi/2$ for which

$$\mathcal{H} = -J \sum_{\langle ij \rangle} ((\mathbf{S}_i \cdot \mathbf{S}_j)^2 - 1), \quad (30)$$

where the constant -1 is added for convenience. The Hamiltonian (30), as well as the Hamiltonian at other special values of θ , has the SU(3) symmetry. Using the ordinary S^z representation basis $|\sigma_i\rangle$ ($\sigma_i = -1, 0, +1$)

$$S_i^z |\sigma_i\rangle = \sigma_i |\sigma_i\rangle,$$

the pair Hamiltonian can be re-written as

$$- \mathcal{H}_{ij} = J \times \hat{\Sigma} \times \hat{\Delta}(g_H). \quad (31)$$

Here, $\hat{\Sigma}$ is the operator that carries the sign, whose matrix element is $+1$ or -1 , and is -1 if and only if one of the initial state (σ_i, σ_j) and the final state (σ'_i, σ'_j) is $(0, 0)$ and the other is $(1, -1)$ or $(-1, 1)$. It is easy to see that this sign is irrelevant since the negative signs always occur in pairs leaving the sign of the whole system positive. Therefore, we can simply neglect the operator $\hat{\Sigma}$, as we do in what follows. The operator $\hat{\Delta}(g_H)$ is defined by its matrix elements, which we denote by $\Delta(S_u, g_H)$ and are defined as

$$\begin{aligned} \Delta(S_u, g_H) &\equiv \langle \sigma'_i, \sigma'_j | \hat{\Delta}(g_H) | \sigma_i, \sigma_j \rangle \\ &= \begin{cases} 1 & (\text{if } \sigma_i + \sigma_j = \sigma'_i + \sigma'_j = 0) \\ 0 & (\text{otherwise}) \end{cases}. \end{aligned}$$

This is almost identical to (21). The only difference is that the present operator is defined on a larger Hilbert space ($\sigma_i = -1, 0, 1$) than the previous one ($\sigma_i = -1/2, 1/2$). It is therefore obvious that the present problem is a generalization of the ordinary SU(2) antiferromagnetic Heisenberg model. The constraint imposed by $\hat{\Delta}(g_H)$ upon the world-line configuration can be expressed by the same graph element as the one in the SU(2) case, i.e., the horizontal graph in Table I. The local spins bound by the graph must take the values complementary to each other (such as $+1$ and -1 , or 0 and 0). Therefore, once a loop has been formed, the local spin value must be 0 everywhere along the loop or it must alternate between $+1$ and -1 . As in the SU(2) case, choosing a local spin state at one point of the loop determines the state of the whole loop. The difference is simply that every loop can take three possible states rather than two. The loop flipping process must be altered accordingly when we consider the loop algorithm for the present model; we must choose one state among three possible ones with equal probability for each loop. All the rest of the procedure remains the same. For example, the graph assignment is done in the same way as the SU(2) case; the horizontal graph elements are assigned with the density J between two nearest neighbor sites if the local spin values at the two sites are complementary to each other.

A similar algorithm can be constructed in other cases with lower symmetry, i.e., the SU(2) symmetry, for the parameter region $-3\pi/4 \leq \theta \leq -\pi/2$. We start from the expression²⁹⁾

$$- \mathcal{H}_{ij} = J((- \sin \theta + \cos \theta) \hat{\Delta}(g_H) - (\cos \theta) \hat{\Delta}(g_C)), \quad (32)$$

where an irrelevant sign and an additive constant have been omitted. The symbol $\hat{\Delta}(g_C)$ corresponds to the cross graph in Table I. To be more specific, its matrix elements are

$$\Delta(S_u, g_C) \equiv \begin{cases} 1 & (\text{if } \sigma_i = \sigma'_j \text{ and } \sigma_j = \sigma'_i) \\ 0 & (\text{otherwise}) \end{cases}.$$

The loop construction and the loop flipping can be done in much the same way as described above.

These algorithms can be easily generalized to the case where each local spin variables takes N possible values, i.e., $\sigma_i = (-N + 1)/2, (-N + 3)/2, \dots, (N - 1)/2$. Of particular interest is the Hamiltonian that consists of $\hat{\Delta}(g_H)$ only, which possesses the SU(N) symmetry. See ref. 44 for results of a numerical simulation. It should be also pointed out that the algorithm presented here is similar to the Swendsen-Wang algorithm¹⁴⁾ for the classical antiferromagnetic N -state Potts model, in which a cluster is constructed in much the same way as the SW algorithm for the Ising model, and each cluster can take N different states.

2.8 Magnetic field

For a number of models, the loop algorithm described above is the most efficient algorithm among the ones described in the present article. For instance, the easy-axis XXZ model with general spin size s can be best handled with the loop algorithm. The easy-plane XXZ models can also be simulated most efficiently with the loop algorithm if there is

no external magnetic field parallel to the diagonalization axis, namely, the z axis in the present case. However, if we have such an external magnetic field and it is competing with the spin–spin couplings, the loop algorithm does not work.⁹⁾

To see this, we first describe a simple loop algorithm for a case with magnetic field in the z -direction, and see what makes it inefficient. In the simple algorithm, we deal with the magnetic field separately; we simply neglect the external field while assigning graph elements. Then, in flipping clusters or loops, we take it into account. This can be formally justified as follows. First decompose the Hamiltonian into the field-free part $\mathcal{H}^{(0)}$ and the field part $\mathcal{H}^{(1)}$.

$$\mathcal{H} = \sum_b \mathcal{H}_b^{(0)} + \sum_i \mathcal{H}_i^{(1)}.$$

Then, we have

$$\begin{aligned} W_L(S) &= \prod_{b,k} \langle \psi_{b+1}(k) | e^{-\Delta\tau \mathcal{H}_b^{(0)}} | \psi_b(k) \rangle \\ &\quad \times \prod_{i,k} \langle \psi_1(k) | e^{-\Delta\tau \mathcal{H}_i^{(1)}} | \psi_1(k) \rangle \\ &= \sum_G W_L^{(0)}(S, G) V(S), \end{aligned}$$

where we have assumed that the field part is diagonal. The factor $V(S)$ is the contribution from the magnetic field term defined as

$$V(S) \equiv e^{-\sum_{k,i} \Delta\tau \langle \psi_1(k) | \mathcal{H}_i^{(1)} | \psi_1(k) \rangle} = e^{\int_0^\beta d\tau \sum_i F_i(\tau)},$$

where

$$F_i(k\Delta\tau) \equiv \langle \psi_1(k) | (-\mathcal{H}_i^{(1)}) | \psi_1(k) \rangle.$$

The factor $V(S)$ can be rewritten in terms of clusters as

$$V(S) = \prod_c V_c(S),$$

where $V_c(S)$ is defined for each cluster c in G as

$$V_c(S) \equiv e^{\int_c dX F(X)}.$$

Here $\int_c dX$ is the $(d+1)$ -dimensional integral in the space-time over the cluster c .

It is obvious from this form that the magnetic-field term does not affect the graph-assignment probability (7) whereas it modifies the cluster-flipping probability. Substituting $W(S, G) = W_L^{(0)}(S, G) V(S)$ for (7), we obtain

$$T(S|G) = \Delta(S, G) \prod_c \frac{V(S_c)}{\sum_{S'_c} V(S'_c)},$$

where S_c is the specifier of the state of the cluster c . (For the $s = 1/2$ spin models S_c is a binary variable.) This indicates that we have to choose the cluster state S_c with the probability $V(S_c) / \sum_{S'_c} V(S'_c)$ for each c . When the external field is zero, this reduces the random unbiased choice between two possible cluster states as already explained above.

This procedure works well when the magnetic field is cooperative with the spin–spin couplings as is the case with the easy-axis ferromagnetic XXZ model with a uniform magnetic field parallel to the axis. However, if the field is competitive against the spin–spin couplings, as is the case with the antiferromagnet with a uniform field, the procedure becomes increasingly inefficient as the temperature is lowered. To see this, we consider a small system that

consists of only two spins coupled with each other by an antiferromagnetic interaction. Let us first suppose that spins are totally aligned in the direction of the magnetic field. Because of the graph-assignment rule presented above, the density with which we assign the non-trivial graph elements is zero. The resulting graph is therefore a trivial one that consists of two loops, i.e., two vertical lines going from the bottom of the system to the top. In order to visit a different spin configuration, we have to flip at least one of these two loops. However, the flipping must be done against the magnetic field, and the flipping probability is roughly proportional to $e^{-\beta H}$ according to the simple procedure discussed above. Here H is the magnetic-field strength. Therefore, flipping seldom takes place at a low temperature regardless of the magnitude of J relative to H . However, we need to visit other states frequently, particularly when J is much larger than H . When H is much larger than J , on the other hand, we need to visit the completely aligned state frequently. However, it hardly happens if we start from the anti-parallel state in which one of the two spins is up and the other is down because the transition probability to the completely aligned state is exponentially small at a low temperature. This is because we cannot change the total magnetization unless we flip a loop whose temporal winding number is not zero; but such a loop can be formed only when no non-trivial graph elements are assigned to the system. Such an event takes place with an exponentially small probability proportional to $e^{-\beta J/2}$, regardless of H . In short, when the magnetic field competes with the other couplings, the transition probability from one value of magnetization to another becomes very small at low temperatures, making the simulation extremely slow.

2.9 Worm algorithm

There are cases where one can avoid the freezing problem due to the magnetic field by using the worm algorithm.^{3,9)} Updates of the world-line configuration in the worm algorithm is done through stochastic movements of two discontinuity points at which the conservation rule is violated. In the case of particle–hole problems or $s = 1/2$ quantum spin problems, a world-line terminates at these points. Only one of the two points moves around in our implementation, and we call the mobile one the *head* of the worm and the other stationary one the *tail*. A *worm* is the pair of these two points. (A worm in the present paper does not have a ‘body’, in contrast to real ones.⁴⁵⁾) The spin configuration is modified as the head moves around. There can be several types of heads depending on the change in the local state caused by them. In many applications, however, we only consider two types; the one for which the local state above the head is one higher than that below (positive head), and the one for which the opposite is true (negative head). The types of the tail are defined likewise.

One step in the worm algorithm consists of three elementary movements and their anti-movements: the creation/annihilation of a worm, the vertical movement, and the jump/anti-jump, as illustrated in Fig. 9. Each movement is a stochastic transition that satisfies the detailed balance condition with respect to the weight in (14) with an additional contribution from the source term. That is, we consider the Hamiltonian $\mathcal{H} - \eta Q$, where the operator Q

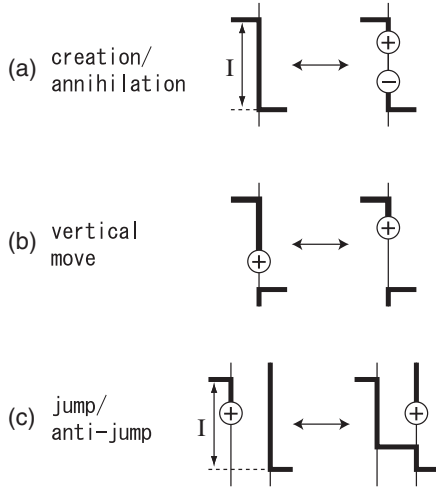


Fig. 9. Three elementary movements and their anti-movements of the head. At the positive head labeled '+', the local spin variable increases by one whereas at the negative one labeled '-', it decreases by one. The thick line, therefore, is the part where the spin value is greater relative to the thin part.

represents the source and is the sum of local operators, $Q = \sum_i Q_i$. The partition function is expressed in the discrete imaginary time as

$$Z = \lim_{L \rightarrow \infty} \sum_S \tilde{W}_L(S), \quad (33)$$

$$\begin{aligned} \tilde{W}_L(S) &\equiv \prod_u \langle \psi'_u | (1 - (\Delta\tau) \mathcal{H}_u) | \psi_u \rangle \\ &\times \prod_v \langle \psi'_v | (1 + (\Delta\tau) \eta Q_v) | \psi_v \rangle, \end{aligned} \quad (34)$$

where v specifies a local unit defined on a vertical line i so that every layer contains exactly one unit for each vertical lines. The symbol Q_v stands for Q_i as \mathcal{H}_u does for \mathcal{H}_b in (14). The right-hand side of (34) consists of three parts; the contribution from the diagonal matrix elements of the Hamiltonian, the contribution from the off-diagonal matrix elements, and the contribution from the source term. Specifically, denoting the number of kinks as N_{kink} and the number of discontinuity points as N_{dc} , we can rewrite the weight as

$$\begin{aligned} \tilde{W}_L(S) &= W_D \times (\Delta\tau)^{N_{\text{kink}}} W_K \times (\Delta\tau)^{N_{\text{dc}}} W_W, \\ W_D &\equiv \prod_{u: \text{non-kink}} \langle \psi'_u | (1 - (\Delta\tau) \mathcal{H}_u) | \psi_u \rangle \\ &= \exp \left(- \int_0^\beta d\tau \sum_b \langle \psi(\tau) | \mathcal{H}_b | \psi(\tau) \rangle \right), \\ W_K &\equiv \prod_{u: \text{kink}} \langle \psi'_u | (-\mathcal{H}_u) | \psi_u \rangle, \\ W_W &\equiv \prod_{v: \text{dc}} \langle \psi'_v | \eta Q_v | \psi_v \rangle. \end{aligned}$$

Since we only need up to the second order in η , we truncate the last product at the second order, which is indicated by the prime in $\prod'_{v: \text{dc}}$. In what follows, we consider only configurations with no worm or those with exactly one worm.

The detailed balance condition

$$T(S'|S) \tilde{W}_L(S) = T(S|S') \tilde{W}_L(S') \quad (35)$$

has to be satisfied by the transition matrices expressing three elementary movements of the head.

In the creation process [Fig. 9(a)], we first choose a site i , a uniform interval of it, say I , and two temporal positions in I , τ and τ' , for placing the worm. Then we decide if the proposed placement is accepted. In this process, W_W and W_D are altered while W_K remains the same. Let the probability of choosing I be $P(I)$, the probability of choosing x and y in the intervals dx and dy respectively, $P(x, y) dx dy$, and the probability of acceptance, P_{create} . For the inverse process, namely, the annihilation, we do not have to choose I , τ or τ' . We simply decide whether we erase the worm or not with probability $P_{\text{annihilate}}$. Thus the detailed balance (35) in this case can be rewritten as

$$\begin{aligned} \frac{1}{2} P_{\text{create}} \times P(I) (d\tau d\tau' P(\tau, \tau')) W_D(S) \\ = P_{\text{annihilate}} \times W_D(S') (d\tau d\tau' W_W(S')), \end{aligned} \quad (36)$$

where S is the state with no worm and S' is the state with a worm whose head is at x and the tail y . The factor $1/2$ on the left-hand side is due to the two possibilities concerning the initial type of the worm. Here we obviously have many degrees of freedom. One of many possible choices, though it may not be the optimal, is given by setting $P(I) = |I|/(N\beta)$, which corresponds to choosing the interval I by “throwing a dart”. Then, we obtain

$$\frac{P_{\text{create}}}{P_{\text{annihilate}}} = R \equiv \frac{2N\beta \int_I dx dy W_D(S') W_W(S')}{|I| W_D(S)}$$

and

$$dx dy P(x, y) = \frac{dx dy W_D(S') W_W(S')}{\int_I dx dy W_D(S') W_W(S')}. \quad (37)$$

To be more specific, the acceptance probabilities can be chosen as

$$P_{\text{create}} = \min(1, R), \quad P_{\text{annihilate}} = \min(1, R^{-1}), \quad (38)$$

and the free parameter η is adjusted so that neither of P_{create} nor $P_{\text{annihilate}}$ is too small.

In practice, it is often too cumbersome to compute (37) every time a creation or an annihilation is proposed. Therefore, an alternative may be used. That is,

$$dx dy P(x, y) = \frac{dx dy \exp(-|x - y| \times \overline{\Delta V})}{\int_I dx dy \exp(-|x - y| \times \overline{\Delta V})}, \quad (39)$$

where $\overline{\Delta V}$ is the average excess action (per unit time) caused by the creation of the worm,

$$\overline{\Delta V} \equiv - \frac{1}{|I|} \ln \left(\frac{W_D(S(I))}{W_D(S)} \right),$$

where $S(I)$ is the world-line configuration that results from creating the worm with the tail at the bottom and the head at the top of the interval I . When this alternative is used, R in (38) must be modified accordingly, so that the detailed balance condition (36) is satisfied. (As a result, the new R depends on the times τ and τ' .)

The vertical movement [Fig. 9(b)] is much simpler. The head moves to another point of the vertical line on which it is currently located. The new position is chosen from the interval I that contains no kink in it and is delimited by two kinks. The choice is made with an appropriate density so that the detailed balance is satisfied. Since the kink contribution W_K and the worm contribution W_W are the same for the initial and the final state of the move, we have only to consider the diagonal part W_D for the detailed balance. Namely, the detailed balance (35) is satisfied if the probability $d\tau P_{\text{vertical}}(\tau)$ of choosing the new position of the head in the interval $d\tau$ is $d\tau P_{\text{vertical}}(\tau) \propto W_D(S')$ where S' is the state after the head position is moved to τ . Therefore, $P_{\text{vertical}}(\tau)$ should be

$$d\tau P_{\text{vertical}}(\tau) = \frac{d\tau W_D(S')}{\int_I d\tau W_D(S')}.$$

Finally we consider the jump and the anti-jump [Fig. 9(c)]. A jump is a movement in which the head changes its spatial position while the temporal position is kept. At the same time, a kink is created in a jump process between the two vertical lines. There are two kinds of jumps according to the temporal location of the kink to be created; whether it is above the head or below. In the original article,⁹⁾ one of the two is called a reconnection. We do not distinguish the two, since both the movement can be done in exactly the same way. The anti-jump, too, has two kinds according to the position of the kink relative to the head. The detailed balance in the jump process can be worked out in a fashion similar to the two cases discussed above. This time, all three of W_D , W_K and W_W change. The detailed balance condition is

$$P_{\text{jump}} \times (P(x) dx) W_W(S) W_D(S) W_K(S) \quad (40)$$

$$= P_{\text{anti-jump}} \times W_W(S') W_D(S') (W_K(S') dx), \quad (41)$$

where S' is the state after the jump. P_{jump} and $P_{\text{anti-jump}}$ are the probabilities of accepting a proposed jump and anti-jump, respectively, and $P(x)dx$ is the probability of choosing the position of the new kink in the infinitesimal interval dx around x . We choose

$$dx P(x) = \frac{dx W_D(S') W_K(S')}{\int_I dx W_D(S') W_K(S')}.$$

Then, the acceptance probabilities must be chosen so that

$$\frac{P_{\text{jump}}}{P_{\text{anti-jump}}} = R \equiv \frac{W_W(S') \int_I dx W_D(S') W_K(S')}{W_W(S) W_D(S) W_K(S)}$$

is satisfied. Then, one possible choice for the acceptance probability is

$$P_{\text{jump}} = \min(1, R), \quad P_{\text{anti-jump}} = \min(1, R^{-1}).$$

A few comments on the worm weight may be appropriate here. In general, we can assign a non-trivial weights to the head and the tail. A frequent choice is

$$\langle \psi'_v | Q_v | \psi_v \rangle, \quad (42)$$

where ψ_v and ψ'_v are the local spin states just below the head (or the tail) and the above, respectively, and Q_v is an

operator that represents the order parameter relevant for the model. For example, for the XY model $Q_v = S_v^x$ is used. In the $s = 1/2$ case, in particular, the weight is a constant. The reason for the choice (42) is obvious, considering the relationship between the head's trajectory and Green's function $\Gamma_Q(X' - X) \equiv \langle Q(X') Q(X) \rangle$, with X and X' specifying space-time points. (See §2.16 for estimators of various quantities.) When the worm is assigned the above-mentioned weight, it can be shown that $\Gamma_Q(X)$ is proportional to the frequency with which the head visits a location specified by X relative to the head's original location. Therefore, if the range where $\Gamma_Q(X)$ of $O(1)$ is determined by the system's correlation length, the head's trajectory extends a region whose linear size is roughly equal to the correlation length.

This is desirable since this guarantees that the scale of the update coincides with the correlation length. This is also the case with the loop algorithm with no external field. However, the worm algorithm works better than the loop algorithm when a competing external field exists. This is because the effect of the field is reflected in choosing each local movement of the head. Therefore, a typical trajectory of the head strongly depends on the strength of the field. In the loop algorithm, on the other hand, the loop construction is done with no reference to the external field, making the typical loop, which corresponds to the trajectory of the head, depends on the external field only indirectly. As a result, the acceptance ratio of the loop flipping can be extremely small in the loop algorithm whereas the acceptance is always unity in the worm algorithm (the local spin state along the trajectory is already changed when the head finishes its journey).

2.10 Directed-loop algorithm

The directed-loop algorithm^{4,6)} can be thought of as a hybrid of the loop algorithm and the worm algorithm. While it has an advantage of the worm algorithm, we do not need to do integrations for obtaining the transition probabilities. In addition, although the directed-loop algorithm becomes identical to the loop algorithm when the external magnetic field is zero, it does not have the freezing problem even when the field is turned on.

The directed-loop algorithm can be formulated in much the same way as the formulation of the loop algorithm. Therefore, we start with (12) [or (14)]. In the loop algorithm, we have decomposed the local Hamiltonian into several terms, each corresponding to a particular graph element. In addition, we split each original spin into $2s$ Pauli spins in the case of $s > 1/2$. Therefore, b in (11) is equivalent to $((ij), (\mu\nu), g)$. In the directed-loop algorithm, we do not decompose the local Hamiltonian at all. Accordingly, b in (11) must be regarded simply as (ij) . Then, the procedure of updating G follows from the general prescription in §2.2. For example, we set $G_u = 1$ for a given u with probability $(\Delta\tau)w(S_u) \equiv (\Delta\tau)\langle \psi'_u | (-\mathcal{H}_{ij}) | \psi_u \rangle$ when $\psi'_u = \psi_u$. This means, in the continuous-time formulation, that vertices (which correspond to $-\mathcal{H}_{ij}$ here, rather than $\hat{\Delta}(g)$ in the loop algorithm) are placed with the density $\langle \psi_u | (-\mathcal{H}_{ij}) | \psi_u \rangle$ in uniform intervals. In addition, a vertex is placed on every kink.

The updating procedure for S , on the other hand, is quite different from that in the multi-cluster variant of the loop

algorithm discussed in the previous subsections. There, clusters are formed naturally as a result of the graph assignment because the Hamiltonian has been decomposed into graph elements. Since we do not have graph elements in the directed-loop algorithm, loop (cluster) must be formed in the S -updating process rather than in the G -updating process.

While the S -updating is done with a worm in the directed-loop algorithm similarly to the worm algorithm, the head of the worm in the directed-loop algorithm cannot choose the positions at which it creates kinks unlike the worm algorithm. This is because G has been fixed (i.e., all the vertices are fixed) before the worm is created, and we cannot have a kink at a plaquette on which there is no vertex, i.e., $G_u = 0$. Therefore, new kinks can be made only at the vertices which are fixed during the worm's life-time. However, this is not an essential difference because one can easily generalize⁴⁾ the algorithm so that the vertices are generated dynamically during the head's motion. Another (probably more important) difference between the directed-loop algorithm and the worm algorithm arises from the direction of the head's motion. In the worm algorithm, the direction of the head's motion is biased only by the weight $W(S, G)$ and there is no algorithmically preferred direction. In the directed-loop algorithm, on the other hand, the head has a "moment of inertia" and can go only in the direction that is the same as in the previous step. The head can change its direction of motion only when it is scattered by a scatterer, i.e., a vertex. Therefore, $G_u = 1$ can be interpreted as having a scattering object at u . This is a clear advantage of this method compared to the worm algorithm, because in the worm algorithm, a head in general goes back and forth along a vertical line, sometimes unnecessarily. When applied to the $s = 1/2$ antiferromagnetic Heisenberg model, for example, the trajectory of the head is roughly the same as the loop in the loop algorithm when the field is absent. Therefore, the head's motion in the worm algorithm is a random walk along a loop. While it takes a time proportional to the squared length of the loop for a head to finish its travel in the worm algorithm, it takes only a time proportional to the length in the directed-loop algorithm.

When the head arrives at a vertex, it may or may not change its location as well as the direction of motion. It has four possibilities as to the location after the scattering, namely, the four legs of the vertex (Fig. 10). The choice among the four is made probabilistically. However, unlike all the cases discussed above, we cannot use the detailed balance condition for determining the probability due to the direction of the head's motion. It is obvious that the probability of having the left-most state in Fig. 10 as the final state is zero when the initial state is one of the four states on the right, because of the direction of motion. (The head is moving away from the vertex, not coming in.)

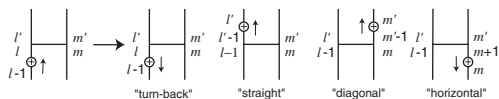


Fig. 10. The four events that can happen when a head hits a vertex. Arrows indicate the directions of the heads' motion. The numbers indicate the local spin states. Namely, $l = 0, 1, \dots, 2s$ correspond to the local spin states $S_l^z = -s, -s+1, \dots, s$, respectively.

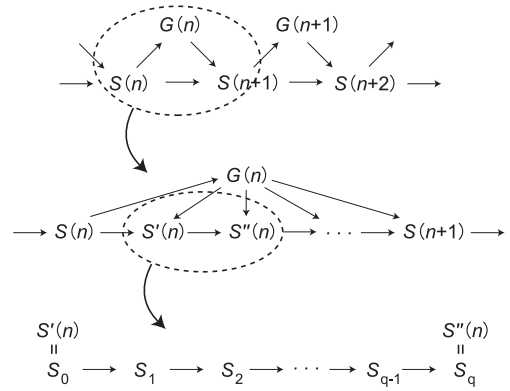


Fig. 11. A schematic illustration of the directed-loop Monte Carlo. The top, middle, and bottom parts show an overview, one step, and one cycle, respectively.

Instead of the detailed balance, we use the time-reversal symmetry condition as we discuss below.

The stochastic process of the directed-loop algorithm can be formally viewed as the stochastic process in the extended state space. The extension of the state is done in two ways. As mentioned above, the first extension is due to the introduction of the auxiliary variable G , and the other is due to the introduction of a worm. Since the directed-loop algorithm is a kind of single-cluster algorithm similar to the Wolff algorithm, the whole stochastic process is not a simple alternating Markov chain as in the loop algorithm (Fig. 1). As illustrated in the top part of Fig. 11, the probability of generating a new state $S(n+1)$ depends not only on the current graph $G(n)$, but also on the current state $S(n)$. This is in contrast to the multiple-cluster variant of the loop algorithm that corresponds to Fig. 1. This updating process of the spin configuration is achieved by a number of worm creation/annihilation cycles. Each cycle starts with a state that contains no worm and ends with another worm-free state. Let us denote the initial state S_0 and the final state S_q where q stands for the number of elementary motions of the head during the life-time. Each state between the two, S_1, S_2, \dots , or S_{q-1} , contains a worm.

Let us denote the transition probability that governs the elementary head motion as $T_w(S'|S)$. (Here we have dropped the dependence on G of the transition probability because it is fixed throughout the cycle.) Instead of the detailed balance condition, this transition probability is chosen so that it satisfies the time-reversal symmetry condition

$$T_w(S_{k+1}|S_k)W_L(S_k, G) = T_w(\bar{S}_k|\bar{S}_{k+1})W_L(S_{k+1}, G), \quad (43)$$

where \bar{S} is the state identical to S except the direction of the head's motion. In other words, \bar{S} is the time-inversion of S . Note that the weight of a state does not depend on the direction of a head. Once (43) is satisfied, the ordinary detailed balance condition is recovered in the process from S_0 to S_q , i.e.,

$$T(S_q|S_0)W_L(S_0, G) = T(S_0|S_q)W_L(S_q, G).$$

This can be seen easily as follows. First we note that

$$T(S_q|S_0)W(S_0, G) = \sum_q \left(\sum_{S_1} \sum_{S_2} \cdots \sum_{S_{q-1}} \right)$$

$$T_w(S_q|S_{q-1})T_w(S_{q-1}|S_{q-2})\cdots T_w(S_1|S_0)W_L(S_0, G).$$

But because of the direction independence of the weight, by using the time-reversal invariance of the transition matrix (43) repeatedly, we obtain

$$T_w(S_q|S_{q-1})T_w(S_{q-1}|S_{q-2})\cdots T_w(S_1|S_0)W_L(S_0, G) = T_w(\bar{S}_0|\bar{S}_1)T_w(\bar{S}_1|\bar{S}_2)\cdots T_w(\bar{S}_{q-1}|\bar{S}_q)W_L(S_q, G).$$

Thus, the detailed balance is recovered for every individual path that leads from S_0 to S_q .

Here we consider the weight of the states with worms. Since the worm is an artifact for the algorithm, in principle we can assign any weight to the states with a worm.⁴⁶⁾ The most natural definition, however, is to use the same expression as (14) with an additional factor for the worm,⁴⁶⁾

$$w_w(S_x) = \eta \langle \psi'_x | S'_x | \psi_x \rangle, \quad (44)$$

where x is h or t corresponding to the head or the tail, respectively. The local state S_x is defined as (ψ'_x, ψ_x) , where ψ_x and ψ'_x are the local spin states just below and above the discontinuity point, respectively. The constant η is included for adjusting the worm creation and annihilation probabilities. A similar factor for the tail is also included. The weight of a state with a worm altogether becomes

$$\tilde{W}_L(S) \equiv w_w(S_h)w_w(S_t) \times \prod_u \langle \psi'_u | (1 - (\Delta\tau)\mathcal{H}_u) | \psi_u \rangle,$$

where S_h and S_t are the local states around the head and the tail, respectively. The product is taken over all the vertices (plaquettes). Note that the weight does not depend on the direction of the head's motion.

Next, we consider how to define $T_w(S'|S)$ so that it satisfies eq. (43). Three cases must be considered; (i) the scattering of the head at the vertex, (ii) the pair creation, and (iii) the pair annihilation. We first look into the case (i). For the scattering process, (43) can be written as

$$T_w(S'|S)\tilde{w}(S_{u+w}) = T_w(\bar{S}|\bar{S}')\tilde{w}(S'_{u+w}), \quad (45)$$

where

$$\tilde{w}(S_{u+w}) \equiv w(S_u)w_w(S_w).$$

Here, S_w is the local state around the head and S_{u+w} stands for (S_u, S_w) . Remember that there are only four possible final states for each initial state. Suppose that $S_u^{(1)}$ is the initial local state of the vertex. The state $\bar{S}_u^{(1)}$ is obviously one of the four possible final states because if the head turns back at the vertex, the state $\bar{S}_u^{(1)}$ is the final state. Let us denote the inverse of the other three possible final states as $S_u^{(2)}$, $S_u^{(3)}$, and $S_u^{(4)}$. Then, the four states $S_u^{(k)}$ ($k = 1, 2, 3, 4$) form a closed set, i.e., if the initial state is among the four the final state is always the inverse of one of the four. Therefore, eq. (45) can be generally decomposed into several closed sets of four equations.

In order to find a solution to one of these quartets, let us suppose that a matrix w_{ij} exists and satisfies the properties

$$w_i(\equiv \tilde{w}(S_u^{(i)})) = \sum_{j=1}^4 w_{ji} \quad (46)$$

and

$$w_{ij} = w_{ji}. \quad (47)$$

Then, it is easy to verify that

$$t_{ij}(\equiv T_w(S_u^{(i)}|S_u^{(j)})) \equiv \frac{w_{ij}}{w_j}$$

satisfies the property (45). Therefore, the problem of solving (45) has been reduced to finding a symmetric matrix that satisfies (46) with given w_i .

The following solution is always available for any model:

$$w_{ij} \equiv \frac{w_i w_j}{\sum_k w_k}. \quad (48)$$

The final state is chosen simply proportional to the weight of the final state if we use this solution; hence the name ‘‘heat-bath’’ type solution. However, it has been known that this solution yields an inefficient algorithm in many cases.

In (46), we have ten free parameters and only four equations. However, the bounce-free condition $w_{ii} = 0$ is often imposed for obtaining better efficiency. In the case of the quantum $s = 1/2$ XXZ model, in particular, the solution becomes unique with this additional constraints. Still, we have six free parameters left. While little is known about the general principle for obtaining solutions that lead to efficient algorithms, good solutions are known for many important cases. In the next section, we show such a solution for the XXZ quantum spin model with an arbitrary s .

As for the pair creation/annihilation process, we have to consider the detailed balance between a state with a worm and a state without. Specifically, the relation

$$T(S'|S) = T(S|S')w_w(S'_h)w_w(S'_t)$$

must hold for the transition probability T where S' and S are the states with and without a worm, respectively. Note that S and S' are identical except that S' contains a worm. The symbols S'_h represents the local state around the head, just before the collision of the head and the tail, whereas S'_t is the state around the tail. It should be noted here that the creation of the worm consists of two steps; the selection of the position of the creation and the rejection/acceptance of the proposed creation. In the discrete-time representation there are NL positions at which we can place a worm. Therefore, denoting the acceptance probability for the creation by P_{create} , we can write $T(S'|S)$ as

$$T(S'|S) = \frac{1}{NL} P_{\text{create}}(S_v),$$

where the S_v is the local state around the proposed point of creation before the creation. On the other hand, there is no position selection in the annihilation process. Therefore,

$$T(S|S') = P_{\text{annihilate}}(S_v).$$

(Note $S'_v = S_v$.) The detailed balance condition becomes

$$\frac{1}{NL} P_{\text{create}}(S_v) = P_{\text{annihilate}}(S_v)w_w(S'_h)w_w(S'_t).$$

This yields the choice of the acceptance probabilities

$$P_{\text{create}} = \min(1, R), \quad P_{\text{annihilate}} = \min(1, R^{-1})$$

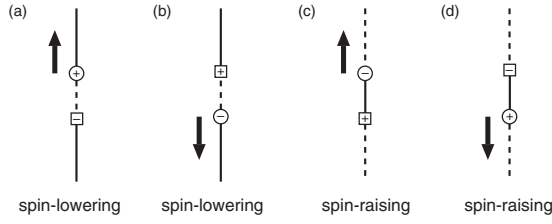


Fig. 12. The four initial states of the worm. Circles denote the heads and squares the tails. A spin state on a dashed line is lower than that on a solid one. In the case $s = 1/2$, in particular, a solid line and a dashed line correspond up and down spins, respectively. An arrow indicates the direction of the head's initial motion.

with

$$R \equiv NLw_w(S'_h)w_w(S'_t).$$

In particular, when the worm weight is the matrix element of S_i^x , we obtain

$$R = NL\eta^2 \langle \psi'_h | S_i^x | \psi_h \rangle \langle \psi'_t | S_i^x | \psi_t \rangle.$$

As we did in §2.9, we can use η for adjusting the transition probabilities. In general, we should choose η so that none of the transition probabilities is too small. If the worm weight does not depend on the local state, as is the case with the $s = 1/2$ and $s = 1$ spin systems, we can choose the free parameter η so that $R = 1$, which is obviously the optimal choice. In general, however, no such choice exists and the creation probability and/or the annihilation probability is smaller than 1 at least in some cases. In §2.11, we present an example of the choice for the XXZ model.

It may be useful to consider here the case of the $s = 1/2$ XXZ spin model to make the description concrete. In this case, the pair creation/annihilation is simple as discussed above. The pair creation is always accepted at any proposed position and the pair annihilation takes place whenever the head meets the tail. When the worm is created at a point where the local spin is up, the upper discontinuity point is positive where the lower one is negative (see Fig. 12). For a point with a down spin, the types of the created worm should be the opposite. The vertex density, as stated at the beginning of the present subsection, is the negated diagonal matrix element of the pair Hamiltonian. For example, it is $J/2$ for the antiferromagnetic Heisenberg model. The probabilities that governs the scattering of the head at vertices can be derived from solving the quartets of the equations discussed above. The result depends on the anisotropy. The solution is presented in §3 for various cases. The resulting algorithm is rather simple for the antiferromagnetic Heisenberg model; whenever the head hits a vertex we let it make the horizontal scattering.

In the original paper by Syljuåsen and Sandvik,⁴⁾ we can find a good example that shows the utility of the directed-loop algorithm. In Fig. 14(b), the integrated auto-correlation time defined for the magnetization is shown (the middle panel) as a function of the magnetic field. The magnetization itself (a) and the average loop size (c) are also shown in the same figure. As has been discussed above, the presence of the magnetic field competing against the exchange couplings makes the configuration freeze in simulations with the loop algorithm. As a result, it is impossible to observe a

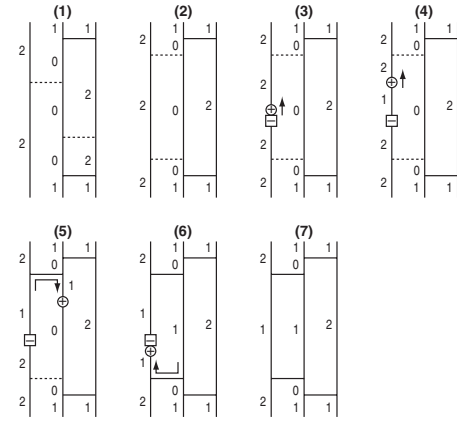


Fig. 13. Assignment of vertices and a cycle of the worm update. Kinks are indicated by solid horizontal lines whereas other vertices are indicated by dotted lines. First, all the existing non-kink vertices are removed while new vertices are assigned (1 to 2). Then, a worm is created (3). The head starts moving and it changes the local spin state (4). Every time the head hits a vertex, one of four possible events in Fig. 10 is chosen stochastically. In this figure, both the scatterings happen to be the horizontal ones (5 and 6). When the head comes back to the original position, it annihilates with some probability leaving a worm-less configuration (7). The cycle [such as the one from (3) through (7)] is repeated a number of times before the vertices are updated.

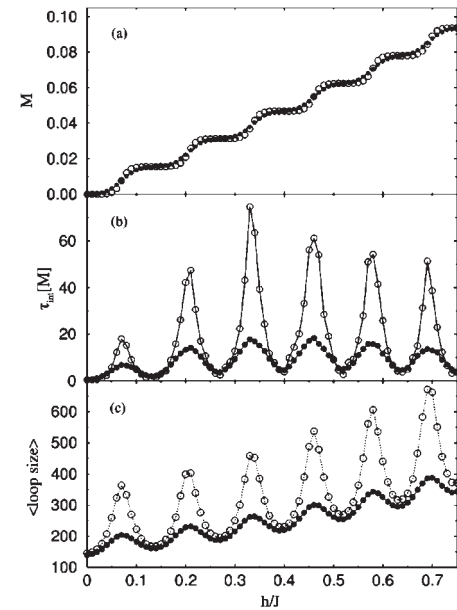


Fig. 14. The magnetization (a), the integrated auto-correlation time (b), and the average length of the loop (i.e., the number of the visited vertices) (c), plotted against the magnetic field h/J for the $s = 1/2$ antiferromagnetic Heisenberg model in one dimension. The solid and open circles are for $\beta = 64$ and $\beta = 128$, respectively. The linear dimension of the system is 64 lattice spacings. (Adopted from Syljuåsen and Sandvik.⁴⁾)

magnetization curve such as Fig. 14(a). By using the directed-loop algorithm, one can obtain the curve within a reasonable amount of computational time. However, the difficulty has not been completely removed as can be seen in Fig. 14(b). The figure shows that the auto-correlation time diverges between two successive plateaus in the magnetization curve. So far, a solution to this problem is not known.

2.11 Coarse-grained algorithm

In general, the solution of the time-reversal-symmetry equation (43) is not unique. In addition, the choice of the worm weight is arbitrary. However, the efficiency of the resulting algorithm largely depends on these choices. While one can obtain the solution by solving eq. (43) numerically in general, there is no automatic way to choose a good one. It is up to the practitioner's physical insight, experience, and, to a certain extent, luck to find a solution and worm weights that lead to an efficient algorithm. Therefore, it is worthwhile to present some efficient solutions for models of particular importance. We here consider the XXZ model with general s . For this model, a set of simple formulas for such solutions are known.⁴³⁾ It includes the single-cluster variant of the loop algorithm for the $s = 1/2$ case. Therefore, the algorithm can be viewed as a natural generalization of the loop algorithm to the cases with larger spins and with a uniform magnetic field. While the solution was found in a way quite different from solving the time-reversal symmetry condition (43), we can show that the resulting solution satisfies (43).⁴⁶⁾ Below, we briefly describe the procedure for obtaining the solution. The explicit formulas for the head-scattering probability and the vertex density of the XXZ model are presented in §3.

The idea is based on the split-spin representation. As discussed in §2.6, it is in general possible to reformulate the model with $s > 1/2$ in terms of the $2s$ Pauli spins: $S_i \Rightarrow \sum_{\mu} \sigma_{i,\mu}$. We would obtain the algorithm in which a head moves around in the space-time that consists of $2s$ vertical lines for each site. What, then, would happen if we look at the real-time animation of the simulation on a low-resolution monitor? The $2s$ lines are blurred and they appear to be a single thick line. In the blurred image on the monitor, we cannot tell on which one of $2s$ lines, namely μ , the head is. The only that we can tell is on which site, i , and at what time, τ , it is. Similarly, we cannot tell on which one of $2s$ lines a particular vertex is footed while we can tell the site and the time. Suppose also that the single line in the blurred image look brighter when we have more up-spins in the $2s$ lines in the original image. Then, there are $2s + 1$ levels of brightness distinguishable in the blurred image. As the head moves, it changes the brightness level of the line by one.

It was pointed out⁴³⁾ that such a blurred animation can be generated with a set of transition matrix defined directly in terms of the brightness, without constructing the original sharp image. We should note that we only need the blurred animation for our original purpose to compute various physical quantities. In short, the split-spin representation is not necessary for describing the algorithm or writing computer codes while it is useful in deriving them, as we see below.

To see how the head-scattering probability can be derived, let us consider the general $s = 1$ antiferromagnetic Heisenberg model for example. Suppose that the head has just hit a vertex that is in the state S_u (the first diagram in Fig. 15). The probability of obtaining the last diagram as the final state of the scattering can be given as

$$T(S'_u|S_u) = \sum_{\Sigma_u} \sum_{\Sigma'_u} T(S'_u|\Sigma'_u) T(\Sigma'_u|\Sigma_u) T(\Sigma_u|S_u). \quad (49)$$

The symbol $T(\Sigma_u|S_u)$ is the probability that the original

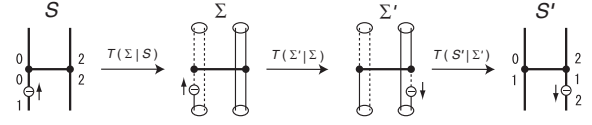


Fig. 15. The derivation of the scattering probability of the head in the “blurred” algorithm. S_u and S'_u are the initial and the final states in the blurred image whereas Σ_u and Σ'_u are the corresponding states in the original sharp image. (The suffix u is dropped in the figure for clarity.) The numbers represent the “brightness” of the line.

(sharp) image of the blurred image S_u is Σ_u . It is proportional to the weight of the original image, i.e.,

$$T(\Sigma_u|S_u) = \frac{w(\Sigma_u, 1) \Delta(\Sigma_u, S_u)}{\sum_{\Sigma_u} w(\Sigma_u, 1) \Delta(\Sigma_u, S_u)},$$

where $\Delta(\Sigma_u, S_u) = 1$ if and only if S_u is the blurred image of Σ_u . The weight $w(\Sigma_u, 1)$ is the one in the split-spin representation,

$$w(\Sigma_u, 1) = \langle \sigma'_{i\mu} \sigma'_{j\nu} | (-\mathcal{H}_{i\mu, j\nu}) | \sigma_{i\mu} \sigma_{j\nu} \rangle.$$

The second factor $T(\Sigma'_u|\Sigma_u)$ in (49) is the scattering probability in the split-spin algorithm, i.e., the scattering probability in the case of $s = 1/2$. The third factor $T(S'_u|\Sigma'_u)$ only represents the compatibility of the final state S'_u with its original image Σ'_u , i.e.,

$$T(S'_u|\Sigma'_u) = \Delta(\Sigma'_u, S'_u).$$

It should be noted here that we do not explicitly introduce the worm weight. In fact, it was pointed out⁴⁶⁾ that the present algorithm agrees with the directed-loop algorithm that follows from a special solution to (46) with the choice of the worm weight: $w(S_x) \propto \langle \psi'_x | S_x^z | \psi_x \rangle$ ($x = h, t$).

The worm creation/annihilation probabilities can also be obtained from the blurring (or coarse-graining). In what follows, we express the local spin state by an integer $l = 0, 1, 2, \dots, 2s$, which corresponds to the $2s + 1$ eigenvalues of S_z^i , $-s, -s + 1, -s + 2, \dots, +s$, respectively. In the split-spin representation, we choose a point uniform-randomly from the space-time, and if the local spin state at the chosen point is up, we place a positive discontinuity point above the negative one. We do the opposite if the local spin state is down. When coarse-grained, this yields the following; when the local spin state at the chosen point is l , the probability of creating a positive discontinuity point above the negative one is $l/2s$. For the worm annihilation, if a positive discontinuity point is above a negative one before the “rendezvous” and the spin state between the two is l , the probability that the two are on the same line in the split-spin representation is $(2s - l)^{-1}$. Therefore, the annihilation takes place with the probability $(2s - l)^{-1}$ in the coarse-grained algorithm. If the relative location of the head and the tail is the opposite, the probability is l^{-1} .

Finally, the vertex assignment density can be derived as follows. Let us consider an interval in which a local spin state is l on one of the two sites and m on the other. In the original (split-spin) image, we assign vertices with the density $\langle \sigma'_{i\mu} \sigma'_{j\nu} | (-\mathcal{H}_{i\mu, j\nu}) | \sigma_{i\mu} \sigma_{j\nu} \rangle$ between the two vertical lines specified by $(i\mu)$ and $(j\nu)$. Therefore, in the blurred image, we assign vertices with the density

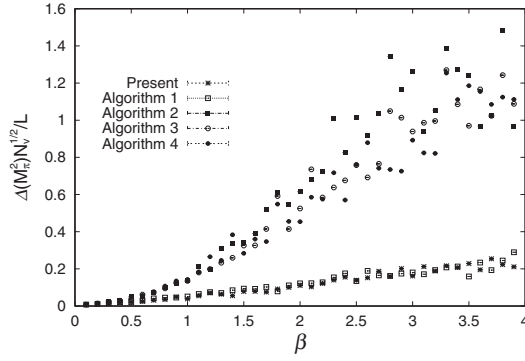


Fig. 16. The statistical error in the estimate of the squared staggered magnetization multiplied by the square root of the average number of scattering events during the lifetime of a worm. The “present” algorithm is the coarse-grained algorithm described in the text. The system is the $s = 1$ antiferromagnetic Heisenberg chain of the length of 64 lattice spacings in a uniform magnetic field $H = 0.1$. Each point is a result of 50 sets of simulations, where each set consists of 20000 creations and annihilations of worms. (Adopted from Harada and Kawashima.⁴³⁾)

$$\begin{aligned} \rho &= \sum_{\mu, \nu} \langle \sigma_{i\mu}, \sigma_{j\nu} | (-\mathcal{H}_{i\mu, j\nu}) | \sigma_{i\mu}, \sigma_{j\nu} \rangle \\ &= l m \rho_{++} + \bar{l} \bar{m} \rho_{+-} + \bar{l} m \rho_{-+} + \bar{l} \bar{m} \rho_{--}, \end{aligned}$$

where $\rho_{\pm\pm}$ is the vertex density for the $s = 1/2$ model with the local spin state $(\pm \frac{1}{2}, \pm \frac{1}{2})$.

Below we see an example that shows the efficiency of the coarse-grained algorithm. Although the algorithm can be applied to an arbitrary s , we only show the case for the $s = 1$ antiferromagnetic Heisenberg chain in a uniform magnetic field. This model has the freezing problem when simulated with the loop algorithm, and it was one of the primary motivations for developing the coarse-grained algorithm. In Fig. 16, we can see the performance of the coarse-grained algorithm. For comparison, we exploited the degrees of freedom in the time-reversal symmetry equation and obtained many solutions. Algorithms 1–4 in Fig. 16 are the ones chosen (in an *ad-hoc* manner) from them. (See the paper⁴³⁾ for how these were chosen.) Plotted in Fig. 16 is $\Delta(M_\pi^2)N_v^{1/2}/L$, where $\Delta(M_\pi^2)$ is the estimated statistical error of the squared staggered magnetization and N_v is the average number of the vertices visited by the head during its lifetime. Here (only in this paragraph and in Fig. 16) L is the system size, not the Trotter number or the order of the expansion. Since the scattering process is the most time-consuming part of the code, the total CPU time is roughly proportional to the total number of scattering events of heads, including the “straight” scatterings. Therefore, the CPU time is proportional to N_v . This is why the statistical error should be multiplied by $N_v^{1/2}$ in order to make the comparison fair. In Fig. 16, we can clearly see that the coarse-grained algorithm performs as well as the best algorithm among the other four (i.e., Algorithm 1). Obviously, there is no exponential slowing-down in the coarse-grained algorithm and Algorithm 1, as was the case with Syljuåsen and Sandvik’s solution for $s = 1/2$.

2.12 Algorithms for bosons

In this section, we present an algorithm for simulating bosonic systems. The algorithm⁴⁷⁾ is based on mapping of

bosonic models to spin models and the coarse-graining discussed in §2.11. The result is similar to the worm algorithm, as we see below. While the ordinary directed-loop algorithm can also be used for the boson models directly, a problem arises from the fact that the boson occupation number is unbounded in general. An artificial bound must be set to make the resulting solution to the detailed balance equation (46) meaningful. The limitation is, however, undesirable since the range of values that the occupation number takes on in the equilibrium is not known *a priori*. While this is not a serious problem in a uniform model where a typical value as well as a typical fluctuation in the occupation number is known, it can be serious in some cases, such as the soft-core boson model with random chemical potential; the typical occupation number may largely vary from site to site in the inhomogeneous potential. The algorithm presented below is free from this problem.

In order to explain the idea, we consider a simple model of non-interacting soft-core bosons on a d -dimensional hyper-cubic lattice. The Hamiltonian is

$$H = -\frac{t}{2} \sum_{(ij)} (b_i^\dagger b_j + b_j^\dagger b_i) - \mu \sum_i b_i^\dagger b_i, \quad (50)$$

where t is the (positive) hopping amplitude, μ is the chemical potential, and b_i^\dagger and b_i are the boson creation and annihilation operators, respectively. In addition, the chemical potential must satisfy $\mu \leq -dt$. In order to map the boson model to the spin model, we use the Holstein–Primakoff transformation,⁴⁸⁾

$$\begin{aligned} S_i^+ &= b_i^\dagger (2s - b_i^\dagger b_i)^{1/2}, \\ S_i^- &= (2s - b_i^\dagger b_i)^{1/2} b_i, \\ S_i^z &= b_i^\dagger b_i - S, \end{aligned}$$

where S_i^+ , S_i^- and S_i^z are spin operators on the i th site. With this transformation, the model of (50) is approximately transformed to an XY spin model,

$$H = -\frac{t}{4S} \sum_{(ij)} (S_i^+ S_j^- + S_j^+ S_i^-) - \mu \sum_i S_i^z. \quad (51)$$

In the limit of infinite s , this mapping is exact. Therefore, if the infinite s limit of the coarse-grained algorithm of the spin system exists, it serves as an exact algorithm for the boson system. In the following, therefore, we consider the infinite s limit of the coarse-grained algorithm discussed in §2.11.

We first consider the beginning and the ending of a cycle; the creation and the annihilation of a worm. In the coarse-grained algorithm, a spin-lowering worm [i.e., the positive head (tail) above the negative tail (head)] is created with the probability $l/2s$. Here, $l = 0, 1, 2, \dots, 2s$ specifies the local spin state, which corresponds to the number of bosons in the bosonic algorithm presented below. Since the number of particles is finite, the probability is zero in the limit of infinite s . Therefore, we always start a cycle with a spin-raising or boson-creating worm [i.e., the negative head (tail) above the positive tail (head)]. On the other hand, when the head meets the tail, it may annihilate with its partner or simply pass it. The probability of the annihilation depends on the type of the head. If the head is of such a type that its passage increases the occupation number by one, i.e., if it is positive and moving downward or if it is negative and

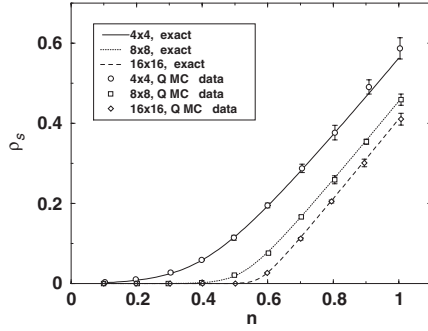


Fig. 17. The superfluid density plotted against the average occupation number for the three-dimensional free lattice-boson system. The lines are the exact analytical values. (Adopted from Šmakov *et al.*⁴⁷⁾)

moving upward, then the annihilation probability is $1/(2s - l)$ where l is the local spin state between the the head and the tail just before they come to the same location. The probability is zero in the infinite s limit. Therefore, the annihilation takes place only for a head whose passage decreases the occupation number by one, and it happens with the probability $1/l$.

Next we consider the vertex assignment and the scatterings of the head at the vertices. Since the density of vertices are proportional to s , at first glance, assigning the vertices in the coarse-grained algorithm is impossible in the limit of infinite s . However, the head goes straight through most of the vertices. The probability that the head changes the direction of motion at a vertex is proportional to $1/s$. Therefore, the density of real scattering events, which is the product of the density of vertices and the scattering probability at a vertex, remains finite. With this density of the scattering event, the imaginary time at which the next scattering happens can be generated by a Poisson process, similar to what we do in taking the continuous-imaginary-time limit in the loop algorithm (see §2.5). In this way, we can make the head move and scatter with a finite number of procedures. (See §3.3 for the details of the procedure.)

In Fig. 17, the result of the numerical simulation using the present algorithm is shown together with the exact result. Plotted is the superfluid density ρ_s at $t\beta = 2$ as a function of the average occupation number n . The transition point is around $n_c \sim 0.6$. With the present method, there is no major difficulty in performing simulations near the critical point as can be seen in Fig. 17. Although not shown in the figure, we also tried a direct application of the directed-loop algorithm with the heat-bath-type solution discussed in §2.10 to the present problem. For the reason mentioned at the beginning of the present section, we have to set the upper bound for the occupation number. When we set it to be 20, which is close to the minimal to perform a non-biased simulation, we observed that the simulation becomes very slow at $n \sim n_c$. It was practically impossible to do a simulation when the occupation number exceeds the critical value.

2.13 Negative-sign problem and meron algorithm

The negative-sign problem is unarguably the worst obstacle in numerical simulations of quantum models. It is originated in the negative matrix elements of the Hamiltonian. When some of the off-diagonal matrix elements of

the local Hamiltonian \mathcal{H}_u are negative, in general the weight (14) can be negative for some of the states S . In such a case, we perform a Monte Carlo simulation of which the target distribution is $|W(S)|$, not $W(S)$. Then, we estimate an arbitrary quantity Q using the identity

$$\begin{aligned} \langle Q \rangle_{\text{thermal}} &= \frac{\sum_S |W(S)| \operatorname{sgn} S Q(S)}{\sum_S |W(S)| \operatorname{sgn} S} \\ &= \frac{\sum_S |W(S)| \operatorname{sgn} S Q(S)}{\sum_S |W(S)|} \bigg/ \frac{\sum_S |W(S)|}{\sum_S |W(S)|} \\ &= \frac{\langle \operatorname{sgn} S Q(S) \rangle_{\text{MC}}}{\langle \operatorname{sgn} S \rangle_{\text{MC}}}, \end{aligned} \quad (52)$$

where $\operatorname{sgn} S = \pm 1$ is an abbreviation of $\operatorname{sgn} W(S)$ and $\langle \cdots \rangle_{\text{MC}}$ is the Monte Carlo average with the weight $|W(S)|$.

The negative contribution to the partition function, Z_- , cancels out with a part of the positive contribution, Z_+ , and the total $Z = Z_+ - Z_-$ must be always positive. In fact, in many cases, the negative contribution *almost completely* cancels out the positive one. This can be seen⁴⁹⁾ by considering a fictitious Hamiltonian \mathcal{H}' , whose matrix elements are the absolute value of the corresponding matrix elements of the original Hamiltonian, \mathcal{H} ;

$$\langle \psi' | \mathcal{H}' | \psi \rangle \equiv \begin{cases} \langle \psi' | \mathcal{H} | \psi \rangle & (\psi' = \psi) \\ |\langle \psi' | \mathcal{H} | \psi \rangle| & (\psi' \neq \psi) \end{cases}.$$

The difference between the free energy per site $\Delta f \equiv (F' - F)/N$ is of $O(N^0)$, when the difference between the two Hamiltonians is extensive. It follows that

$$\frac{Z_+ - Z_-}{Z_+ + Z_-} = \frac{Z}{Z'} = e^{-\beta N \Delta f},$$

where Z' is the partition function of the fictitious system. It is clear from this expression that the positive and the negative part cancel out almost completely at a low temperature and/or for a large size, and it becomes practically impossible to estimate the denominator (as well as the numerator) in (52) because of the statistical error.

In some special cases, the loop algorithm is useful for solving the negative-sign problem in fermionic systems. Considering that there are not many cases where the negative-sign problem is overcome with or without the loop algorithm, it may be worthwhile to mention here the meron algorithm^{10,11)} by which we can overcome the negative-sign problem in some cases. Let us consider the simplest non-trivial model for the fermion,

$$\begin{aligned} \mathcal{H} &= -t \sum_{\langle ij \rangle} (c_i^\dagger c_j + c_j^\dagger c_i) \\ &\quad + U \sum_{\langle ij \rangle} (n_i - 1/2)(n_j - 1/2), \end{aligned}$$

where $t > 0$ and $U \geq 2t$. The spin degrees of freedom are absent. We first consider the case $U = 2t$ and then describe how to generalize the algorithm to the case $U > 2t$.

Apart from the fermion sign due to the exchange of particles, when $U = 2t$, this problem is identical to the $s = 1/2$ antiferromagnetic Heisenberg model [eq. (24) with $J_x =$

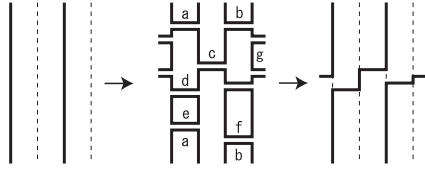


Fig. 18. A loop decomposition (from the left to the middle) and a loop flip (from the middle to the right) in a four-spin chain of the $s = 1/2$ antiferromagnetic Heisenberg model with the periodic boundary condition. The initial spin configuration is the perfect Néel state. The only non-trivial graph elements are the horizontal graphs in Table I. (All the graphs are drawn with solid lines for clarity whereas dashed lines are used in Table I for emphasizing the relative spin orientations.) The loops, c and d , in the middle diagram are merons. From the middle to the right, the loop d is flipped. As a result, the two particles are exchanged in the left diagram, which makes the sign of the whole diagram negative.

$J_y = J_z$. Therefore, the graph decomposition of the Boltzmann weight can be done only with the horizontal graph g_H . (See Table I.) Using the world-line representation discussed above, we can express the partition function as

$$Z = \sum_S \text{sgn } S W_L(S).$$

[Here, the absolute value of the weight in (14) is simply written as $W_L(S)$.] The sign is positive if and only if the world-lines of the fermions corresponds to an even permutation of particles. The weight $W_L(S)$ is the same as the one for the $s = 1/2$ antiferromagnetic Heisenberg model. Applying the graphical decomposition of the weight (19), we can rewrite it as

$$Z = \sum_{S,G} \text{sgn } S W_L(S, G).$$

Let us choose an arbitrary G and fix it, then consider a loop variable $\sigma_l = 0, 1$ assigned to every loop l in G . (We can specify a state S compatible to G by specifying the values of these variables.) The following properties can be shown for the sign of the compatible states to G : (i) The sign $\text{sgn } S$ can be expressed as a product of factors each of which is a function of only one of the loop variables, and (ii) if the sign $\text{sgn } S$ does not depend on the choice of the loop variables, $\text{sgn } S = 1$.

The example of the factorization [the property (i)] is shown in Fig. 18. There are two loops in the middle diagram, c and d , whose flipping changes the sign of the whole configuration. Note that these loops always cause a change in the sign no matter what the initial state may be. On the other hand, all the other loops do not cause any sign change. Generally, whether a flip of a loop causes a sign change or not depends only on the geometric feature of the loop, not on the initial spin configuration. Therefore, one can determine if a given loop may cause the sign-change or not without knowing the state S . The loops that cause sign change are called *merons*. In Fig. 18, there are two merons, c and d .

It is easy to show the positivity of the state with no merons [the property (ii)]. It suffices to show that there is at least one positive configuration among the ones compatible to G . But the sign of the Néel state is positive and the Néel state is compatible with any G . To see the latter, let us notice that the constraint imposed by any graph is simply that when we

trace a loop the local spin state must alternate everytime we make the horizontal (spatial) move. The Néel state obviously satisfies this condition and is compatible to any G .

Because of the two properties, the sign can be expressed as

$$\text{sgn } S = \prod_l (\epsilon_l)^{\sigma_l},$$

where $\epsilon_l = -1$ for merons whereas $\epsilon_l = 1$ for the other loops. The variables denoted as σ_l are the loop variables; each of them takes 0 or 1. They are defined so that $\sigma_l = 0$ for all the loops l if the whole system is in one of the two Néel states. (The choice of the Néel state does not matter because there are always an even number of merons.) The whole partition function now becomes

$$Z = \sum_G \sum_{\{\sigma_l\}} W(S(\{\sigma_l\}), G) \prod_l \epsilon_l^{\sigma_l},$$

where $S(\{\sigma_l\})$ is the state specified by the loop variable σ_l . But the summation over $\{\sigma_l\}$ is zero if there is a meron in G . Therefore, the summation over all G can be replaced by the summation over all meron-free G ;

$$Z = \sum_{\substack{G \\ (\text{no meron})}} V(G) 2^{N_c(G)}$$

where

$$V(G) \equiv \sum_{\{\sigma_l\}} W(S(\{\sigma_l\}), G).$$

Thus, it becomes apparent that we can avoid the negative sign problem if it is possible to perform a simulation in the restricted phase space of the meron-free graphs.

This can be achieved in the following way. Every time insertion or removal of a graph element is attempted, we check whether the attempt would create merons. If it would, such an attempt is rejected. The actual insertion or removal is executed only when it does not create merons. This checking procedure requires a relatively large amount of computational time and raises the complexity of the algorithm. This additional complication, however, is of the algebraic type at most and pays off considering the exponential complexity of the simulation with negative signs. In an actual simulation, because of the necessity of computing the susceptibility, we soften the condition of no merons. Namely, we need to sample the graphs with two merons in order to estimate the susceptibility. Therefore, sampling of graphs is usually done for two-meron graphs as well as meron-free graphs. In this case, the insertion and the removal are rejected only when an attempt is made to create the third meron. For the same purpose, instead of placing a strict upper limit in the meron number, we can also use an extended-ensemble method with respect to the meron number, in which we consider a fictitious weight $W(n_{\text{meron}})$ for controlling the meron number. (For the extended-ensemble method, see §2.15.)

The algorithm can be easily generalized to the case where the easy-axis anisotropy exists, i.e., the case where $U > 2t$ ($J_z > J_x$). In the easy-axis case, we have to introduce the binding graph g_{HB} that bind two loops into one cluster. (See Table I.) For instance, if two merons are bound they form a non-meron. Insertion or removal of these binding graphs is performed in the same way as above; first count the number

of new merons that would be created or annihilated by the attempt and then accept or reject the attempt according to the restriction or the fictitious weight mentioned above.

2.14 $T = 0$ Simulation

The ground state properties are of particular interest in low-dimensional models. While the ground state properties can be deduced, in principle, by extrapolating the finite temperature results to zero temperature, it is prone to the systematic error due to the extrapolation especially when we do not have a solid knowledge about how low the temperature must be to obtain a reasonable extrapolation. Evertz and von der Linden⁵⁰⁾ proposed a method for directly obtaining the zero-temperature expectation values for a given model without extrapolation. The method is a special application of a more general method (proposed also by them) for obtaining results for an infinite lattice. (Note that a system at zero temperature is regarded as a system with an infinite length in the imaginary time direction.) Since this idea can be most clearly described using an example of the ferromagnetic Ising model, let us consider this case first.

The method is closely related to Wolff's single-cluster algorithm.¹⁶⁾ In Wolff's single-cluster algorithm, we start the construction of a cluster from a null cluster that contains no spin. Then, we choose a spin at random from the whole system and include it in the cluster. Next, we assign bonds between the spins surrounding the cluster and the ones already in the cluster in the same way as the SW algorithm. If a new spin is bound to a spin that is already in the cluster, the new spin is also included in the cluster. This procedure is continued until there is no surrounding spins to be checked. When the cluster construction is finished, the spins in the cluster are flipped. The Wolff's procedure is statistically the same as constructing clusters by the SW algorithm for the whole system, choosing a point at random, and then flipping the cluster containing the chosen point. However, the Wolff's procedure is obviously better in efficiency for constructing a single cluster because bond assignments to the spins outside of the formed cluster are a waste of time.

Roughly speaking the method we discuss here is the Wolff's procedure applied to an infinite system. The difference arises from the fact that choosing a point at random from an infinite system does not make much practical sense. Therefore, we simply stick to the same point, which we call the origin, throughout the simulation in the new method. In addition, we have to make sure that the cluster does not percolate, since if it does we cannot finish the cluster construction within a finite computational time. To this end, we start from the Néel state as the initial spin configuration. Due to this choice of the initial spin configuration, the first round of the cluster construction is destined to be very short since we can assign no bond that connects the origin to its nearest neighbor. After flipping this first cluster, i.e., the one that consists of the origin only, we start the second round. We can go a little further this time. Since the spin at the origin is now aligned with its neighbors, we assign bonds between them with a non-zero probability. As a result, the cluster that we obtain from the second round tends to be larger than the first one. In this way, we can go on. It should be noted that we only have to store the spin configuration of the part that is modified in the simulation at

least once and do not have to keep the spin configuration beyond the frontier. It must be also noted that the frequency of visiting a site whose distance from the origin is R is proportional to the correlation function $\Gamma(R) \equiv \langle S(R)S(0) \rangle$. Therefore, the 'already visited' region does not grow too fast after its size reaches the correlation length. This means that the whole process is manageable as long as the system is in the disordered state. Since the distribution of the spin configuration within the region approaches the equilibrium one, one can in principle compute any correlation function that can be defined within this region.

The procedure can be easily generalized for studying the disordered state of the quantum system. For example, in the case of the $s = 1$ antiferromagnetic Heisenberg model, which is disordered even at $T = 0$ due to the Haldane gap, let us consider the split-spin loop algorithm. The initial spin configuration is the complete Néel state where the up-spin straight world-lines and the down-spin ones alternate. Initially, only the local spin configuration at the origin is stored in the computer memory. The head is placed on the origin and the direction of the initial motion is set to be upward or downward (with probability $1/2$). We then generate the scattering object in this direction at some stochastically chosen distance from the origin. The distance is generated following the Poissonian process with the density $J/2$ for each nearest neighbor site, as we have seen in §2.10. At the collision, the head changes its direction of motion as well as the location, as it does in the directed-loop algorithm. The same procedure is repeated until the head comes back to the origin. The spin configuration is stored only for the part that has been visited by the head.

When the system has zero energy gap, a naive application of this method would fail, since the 'already visited' region would glow endlessly. This can be avoided, however, by using a system whose spatial size is finite while its temporal length is infinite ($\beta = \infty$). This is because there is usually a finite gap associated with the system's spatial dimension. Therefore, we can at least perform a zero-temperature simulation for finite systems. This is still somewhat advantageous compared to ordinary simulations for which one needs two kinds of extrapolations, the one with respect to the size and the other with respect to the temperature.

2.15 Extended-ensemble methods

While extending the state space by introducing the auxiliary variables G is a powerful way of speeding up the simulation as we have seen, there is another general strategy for efficient simulations. That is, we can overcome a slow relaxation by using a fictitious target weight, $\tilde{W}(S)$ instead of the given weight $W(S)$. Simulation methods based on this idea are called extended-ensemble methods. In most of the extended-ensemble methods, one does not fix the target weight $\tilde{W}(S)$ but use it as an adjustable function. One of the successful examples is the multi-canonical Monte Carlo (MCMC) method.^{51–53)} In the MCMC method, the weight depends on the state S only through the energy $E(S)$. Therefore, we have

$$\tilde{W}(S) = \tilde{w}(E(S)),$$

and the function $\tilde{w}(E)$ is adjusted adaptively to make the frequency of having the event $E = E(S)$ independent of E .

To this end, several sets of simulation may be performed. For the first set, the initial guess of the appropriate weight that is often used is $\tilde{w}(E) = \text{const}$. In every set, the histogram $h(E)$, i.e., the number of times $E(S)$ takes the value E , is recorded. In other words, every time the configuration is changed we do

$$h(E(S)) := h(E(S)) + 1. \quad (53)$$

At the end of each set, the fictitious weight $\tilde{w}(E)$ is updated as

$$\tilde{w}(E) := \tilde{w}(E)/h(E),$$

and the new weight is used for the next set. In each set, the Metropolis single-spin-flip Monte Carlo method is used with the target weight $\tilde{W}(S) = \tilde{w}(E(S))$ for updating the spin configuration.

At the end of the last set, one can obtain the density of states $g(E)$, as

$$g(E) \equiv \sum_S \delta_{E,E(S)} \approx \text{const} \times (\tilde{w}(E))^{-1}.$$

With this $g(E)$, we can compute the canonical average of an arbitrary quantity Q as

$$Q(T) \equiv \sum_E g(E) w(E) Q(E) / \sum_E g(E) w(E),$$

where $Q(E)$ is the micro-canonical average of Q at the energy E , which we can compute in the last set of simulation.

While this procedure turned out to be quite useful in studying various systems with slow dynamics, one drawback was noticed. That is, the visited range of E do not widen much in each set of simulation, partially due to the poor initial guess of the weight $\tilde{w}(E)$ and also to the slow diffusive nature of the random walk in the energy space. This drawback was removed in Wang and Landau's variant of the MCMC method.⁵⁴⁾ In their method, \tilde{w} is updated every time an attempt is made at changing the spin configuration. The update is done as

$$\tilde{w}(E) := r \times \tilde{w}(E) \quad (54)$$

where $0 < r < 1$ is the reduction factor fixed throughout each set of simulation. At the beginning of each set, the histogram is reset, i.e., $h(E) := 0$ for all E , and the reduction factor is updated as $r := r^\alpha$ where $0 < \alpha < 1$, while the weight $\tilde{w}(E)$ is kept unchanged. Each set does not have a prefixed duration, but is terminated when the histogram becomes approximately flat. Since the dynamic update (54) strongly penalizes the random walker's staying at the same value of the energy, the already visited region widens much faster than the ordinary MCMC.

Since these extended-ensemble methods are complementary to the algorithms described in previous subsections, it is natural to consider the combination of the two. However, since the weight $\tilde{W}(S)$ is not the function of the energy observable in a typical quantum Monte Carlo method, there is no good reason to assign a special role to the energy observable in the quantum case. In addition, the value of the energy observable is not a multiple of a single constant, which is inconvenient for the present purpose. In the loop algorithm (and in the directed-loop algorithm), therefore, the

Wang–Landau method was used^{55–57)} with the histogram of the number of vertices (or graph elements), $n(G) = \sum_u G_u$, rather than the energy. (The idea was originally proposed by Janke and Kappler⁵⁸⁾ and applied to the Ising model.)

Two types of the implementation are possible; in one, the pair Hamiltonian is decomposed graphically,⁵⁵⁾ as in the loop algorithm, and in the other, it is used undivided⁵⁶⁾ as in the directed-loop algorithm. We present below a continuous-time variant of an algorithm closer to the latter.

We start from (15) with u being (i, j) . We control the histogram by introducing an adjustable weight $f(n)$ and replacing $(\Delta\tau)^n \equiv \beta^n/L^n$ by $f(n)/L^n$ in (15), which yields

$$Z = \lim_{L \rightarrow \infty} \sum_S W_L(S),$$

$$W_L(S) = \sum_G W_L(S, G),$$

$$W_L(S, G) = \frac{f(n(G))}{L^n} \prod_u \langle \psi'_u | (-\mathcal{H}_u 0)^{G_u} | \psi_u \rangle.$$

We then perform a Monte Carlo simulation in the (S, G) space so that the detailed balance condition is satisfied with respect to the weight $W_L(S, G)$.

One sweep of the simulation consists of two steps, as in the ordinary directed-loop algorithm. In the first step we update G . To do this, we decompose the whole system into uniform intervals (UIs) with no kink. For each possible local state, S_u , we consider the class of the UIs on which the local state is S_u . Let the sum of the lengths of all the UIs in this class be $I(S_u)$ or simply I . We have IL local units in this class. (Here we use the convention in which the total length of the lattice along the imaginary time direction is 1.) Our task, then, is to set variables G_u for all of these local units. To do this, we first remove all the existing vertices from the UIs. There are $IL C_m$ ways of assigning m vertices to IL units. Therefore, the probability of placing m vertices becomes

$$P(m) \propto \binom{IL}{m} \frac{f(n_0 + m)}{L^m} w^m,$$

where w is the local weight $w \equiv \langle \psi_u | (-\mathcal{H}_u) | \psi_u \rangle$ for the currently chosen type of UIs, and n_0 is the total number of vertices before placing m vertices. In the limit $L \rightarrow \infty$ this leads to

$$P(m) = \frac{1}{A} \frac{(Iw)^m f(n_0 + m)}{m!},$$

where A is the normalization constant. The procedure of assigning graphs to the UIs follows directly from this expression. That is, we first generate a uniform random number r ($0 < r < 1$) and find the integer m that satisfies

$$X(m-1) < r < X(m),$$

where $X(m) \equiv \sum_{m'=0}^m P(m')$. We then choose m points uniform-randomly from the intervals belonging to the current class and place non-trivial graph elements there. We repeat this procedure for all the classes.

In the next step we update the spin configuration. However, this step can be done in exactly the same way as described in §2.10 using worms.

It should be noted here that in the procedure given above, the histogram updating (53) [with $E(S)$ replaced by $n(G)$]

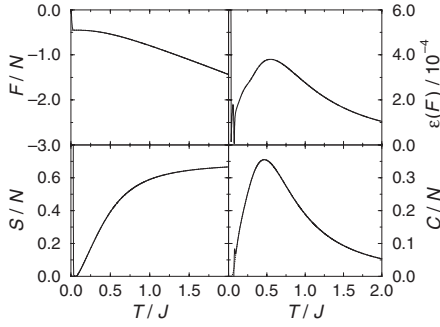


Fig. 19. The free energy F , the entropy S , and the specific heat C of the antiferromagnetic Heisenberg model in one dimension. The system's linear dimension is 10 lattice spacings. The relative error in the free energy $\epsilon(F) \equiv |(F - F_{\text{exact}})/F_{\text{exact}}|$ is also shown. (Adopted from Troyer *et al.*⁵⁶⁾)

can be done only once in a sweep because there is no intermediate state in the procedure. On the other hand, in other methods such as the original algorithm proposed by Janke and Kappler,⁵⁸⁾ the histogram updating can be done after every local assignment of a vertex. This difference would make the statistical error greater than it should be for the present method. This apparent disadvantage can be easily avoided by adding $P(n)$ to the histogram, rather than 0 or 1. Specifically, we should replace (53) by⁵⁵⁾

$$h(n) := h(n) + P(n).$$

The utility of the extended-ensemble methods can be demonstrated best in the computation of the quantities directly related to the density of states, such as free energy, entropy and specific heat. These quantities can be computed easily with the aid of the extended-ensemble methods. In Fig. 19, some results⁵⁶⁾ of the computation with the extended-ensemble method is shown. The method used to obtain the results is based on the discrete-time formulation and the details are different from the one discussed here. However, the basic idea is the same and the efficiency is believed to be similar. Because of the cut-off (the maximum order of the expansion), the results deviate from the exact results at low temperatures ($T/J < 0.1$) whereas they are indistinguishable from the exact results at high temperatures ($T/J > 0.1$) in the scale shown in Fig. 19. (Note, however, that the accessible temperature range can be easily widened by modifying the cut-off.)

It was shown recently⁵⁷⁾ that a further improvement can be made by employing the broad-histogram method.^{59,60)} Compared to the other extended-ensemble methods, this method is unique in that it is based on the exact relation between the expectation value of the transition matrix and the density of states. While one estimates the density of states directly from the histogram itself in the ordinary extended-ensemble methods, the micro-canonical average of the transition matrix is used in the broad-histogram method. This estimator generally gives a better result than the histogram itself because of its macroscopic nature. An improvement of more than one order of magnitude in the relative error was reported⁶¹⁾ in the case shown in Fig. 19.

2.16 Estimators and efficiency of algorithms

As discussed in §2.2, the local updating algorithm is slow

to approach the equilibrium because the size of the region modified at a time is fixed and can be much smaller than the correlation length. On the other hand, the size of the modified region is roughly equal to the correlation length (both in space and in time) in the three algorithms discussed in this article; the loop, the worm and the directed-loop algorithm. This can be seen by considering the estimators of Green's functions. In this section, we therefore discuss estimators of various quantities including Green's functions.

Let us consider a quantity \hat{Q} that can be decomposed in the same way as the Hamiltonian is in (11);

$$\hat{Q} = \sum_b \hat{Q}_b.$$

Then, its canonical average can be expressed with the source term as

$$\langle \hat{Q} \rangle = \frac{1}{\beta} \left[\frac{d}{dH} Z(H) \right]_{H \rightarrow 0} / Z(H) \quad (55)$$

where

$$Z(H) \equiv \text{Tr} e^{-\beta(\mathcal{H} - H\hat{Q})}. \quad (56)$$

Accordingly, the weight (14) is modified as

$$W'_L(S) \equiv \prod_u \langle \psi'_u | [1 - \Delta \tau (\mathcal{H}_u - H\hat{Q}_u)] | \psi_u \rangle. \quad (57)$$

Substituting (56) for (55), we can express the thermal average of \hat{Q} as the Monte Carlo average of the estimator $Q(S)$ as

$$\langle \hat{Q} \rangle_{\text{thermal}} = \langle Q(S) \rangle_{\text{MC}} \quad (58)$$

with

$$Q(S) \equiv \frac{1}{\beta} \sum_u q(S_u), \quad q(S_u) \equiv \frac{\langle \psi'_u | \Delta \tau \hat{Q}_u | \psi_u \rangle}{\langle \psi'_u | 1 - \Delta \tau \mathcal{H}_u | \psi_u \rangle}. \quad (59)$$

In the limit of $L \rightarrow \infty$, eq. (59) becomes

$$q(S_u) = \begin{cases} \Delta \tau \langle \psi_u | \hat{Q}_u | \psi_u \rangle & (\psi'_u = \psi_u) \\ \frac{\langle \psi'_u | \hat{Q}_u | \psi_u \rangle}{\langle \psi'_u | (-\mathcal{H}_u) | \psi_u \rangle} & (\psi'_u \neq \psi_u) \end{cases}. \quad (60)$$

Therefore, if the Q is expressed as a diagonal operator, the above equation is reduced to

$$Q(S) = \frac{1}{\beta} \int_0^\beta d\tau Q(\tau) \quad (L \rightarrow \infty), \quad (61)$$

where $Q(\tau) \equiv \langle \psi(\tau) | Q | \psi(\tau) \rangle$. For example, the magnetization $Q \equiv \sum_i S_i^z$ can be easily evaluated with this formula. In general, however, we cannot ignore the contribution of kinks ($\psi'_u \neq \psi_u$) in $Q(S)$. For example, the contribution of a kink to the total energy $\langle \mathcal{H} \rangle$ is $O(-1/\beta)$.

Next, suppose that the operator Q_u has a non-zero off-diagonal matrix element for some local state whereas the corresponding matrix element of the Hamiltonian is zero. In such a case, the estimator $q(S_u)$ diverges, and we cannot define $Q(S)$ to start with. An example is the measurement of the squared magnetization in the transverse direction, $Q \equiv (\sum_i S_i^x)^2$, with the Hamiltonian being that of the XXZ

Hamiltonian.

By using the loop algorithm, such non-diagonal quantities may be measured.⁸⁾ The idea is based on the improved estimator. An improved estimator is an estimator defined in terms of the graph degrees of freedom rather than the original spin (or occupation number) degrees of freedom. A classical example is Wolff's estimator¹⁶⁾ of the susceptibility of the ferromagnetic Ising model. An improved estimator can be generally derived as follows. Consider first the case where an ordinary estimator $Q(S)$ can be defined for a quantity \hat{Q} . The thermal average can be expressed as

$$\begin{aligned} \langle Q \rangle_{\text{thermal}} &= \frac{\sum_S W(S) Q(S)}{\sum_S W(S)} \\ &= \frac{\sum_{S,G} W(S,G) Q(S)}{\sum_{S,G} W(S,G)} = \frac{\sum_G W(G) Q(G)}{\sum_G W(G)}, \end{aligned}$$

where $Q(G)$ is the fixed-graph average of $Q(S)$:

$$Q(G) \equiv \sum_S W(S,G) Q(S) / W(G). \quad (62)$$

In the case of the magnetic susceptibility of the Ising model, we take $Q(S) \equiv N^{-1}(\sum_i S_i)^2$. Then, eq. (62), with $W(S,G)$ defined by (6) and (4), is reduced to

$$Q(G) = 2^{-N_c(G)} \sum_S \Delta(S,G) Q(S),$$

where $\Delta(S,G)$ is defined in (9). Let us define cluster variables σ_c so that $S_i = \sigma_{c(i)}$ with $c(i)$ being the specifier of the cluster to which i belongs. Then, $Q(S) = N^{-1} \sum_{i,j} S_i S_j$ can be rewritten as $Q(S) = N^{-1} \sum_{c,c'} V_c V_{c'} \sigma_c \sigma_{c'}$ where V_c is the total number of sites in the cluster c . Therefore, we obtain

$$Q(G) = N^{-1} \sum_c (V_c)^2 \quad (63)$$

as the graphical estimator of the squared magnetization.

A similar argument can be used for deriving an improved estimator of the uniform magnetic susceptibility of a quantum spin models,

$$Q \equiv \chi_{zz} = N^{-1} \int_0^\beta d\tau \langle M_z(\tau) M_z(0) \rangle.$$

The ordinary estimator of this quantity is

$$Q(S) = (N\beta)^{-1} \left(\int dX S^z(X) \right)^2,$$

where $X \equiv (i, \tau)$ and the integral $\int dX$ stands for $\sum_i \int d\tau$. Then, similar to the case of the Ising model, the improved estimator can be expressed by (63) with V_c replaced by the cluster magnetization M_c ,

$$M_c \equiv \int_c dX S^z(X).$$

Now, let us consider the case where $Q(S)$ cannot be defined, as is the case with off-diagonal quantities such as the transverse susceptibility

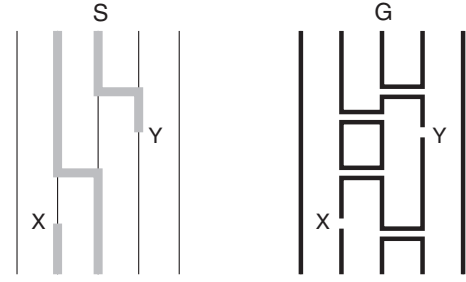


Fig. 20. The spin configuration S and the graph G that appear in the computation of an off-diagonal Green's function. Discontinuities in the world-lines and in the graph at two points X and Y are tolerated.

$$Q = (N\beta)^{-1} \mathcal{T} \int dX dY S^x(X) S^x(Y),$$

where $X = (i, \tau)$ and $Y = (j, \tau')$ specify two points in the space-time. The symbol \mathcal{T} indicates the time-ordered product. We can express its thermal average as

$$\begin{aligned} \chi_{xx} &\equiv \langle Q \rangle_{\text{thermal}} \\ &= (N\beta)^{-1} \int dX dY Z^{-1} \sum'_S W'(S) \\ &\quad \times \langle \psi'(X) | S_i^x | \psi(X) \rangle \langle \psi'(Y) | S_j^x | \psi(Y) \rangle, \end{aligned}$$

where $\psi'(X)$ and $\psi(X)$ are the states just above and below the point X , respectively. The prime in \sum'_S indicates that the summation is taken over all the states that have discontinuities at X and Y . Such a state is shown in Fig. 20. The prime in $W'(S)$ indicates that it allows the discontinuities at the two points. (Note $W(S) = 0$ when the state S has any discontinuities.)

Then, we introduce graphs in the above expression to yield

$$\begin{aligned} \chi_{xx} &= (N\beta)^{-1} \int dX dY Z^{-1} \sum'_{S,G} V(G) \Delta(S,G) \frac{1}{4} \\ &\quad (1 - \delta_{\psi(i,\tau+0),\psi(i,\tau-0)})(1 - \delta_{\psi(j,\tau'+0),\psi(j,\tau'-0)}). \quad (64) \end{aligned}$$

We can take the summation over S . Note here that there is no way to assign local spin values along a loop so that the value is discontinuous at one and only one point in the loop. It means that the summation is zero unless the two points X and Y are connected by the graph G . Therefore, the result of the summation becomes

$$\chi_{xx} = (N\beta)^{-1} \int dX dY \frac{\sum_{G'} 2^{n_c(G')} V(G') \frac{1}{4} \delta_{l(X),l(Y)}}{\sum_G 2^{n_c(G)} V(G)},$$

where $l(X)$ and $l(Y)$ are the specifiers of the loops to which the points X and Y belong respectively. The summation in the numerator is over the set of graphs that yields a non-zero term in the summation (64), i.e., graphs that have at least one matching spin configuration with two discontinuity points. For most quantities and models, this coincides with the set of graphs over which the summation in the denominator is taken, i.e., the graphs that have at least one matching spin configuration with no discontinuity points. (In some pathological cases, however, this is not the case, and the present estimator does not work in such cases.) If such is the case,

the above expression can be simply rewritten as

$$\chi = \frac{1}{4} (N\beta)^{-1} \left\langle \sum_l V_l^2 \right\rangle_{\text{MC}}. \quad (65)$$

Thus we have obtained estimators of the z -component susceptibility (63) for the Ising model and the x -component susceptibility (65) for the quantum model. In both cases, the estimator is expressed as the average cluster (or loop) size, $\bar{V}_c \equiv \sum_c (V_c)^2 / \sum_c V_c$. This means that the typical size of clusters correctly reflects the correlation length in the loop algorithm. This is the reason why the loop algorithm works very efficiently in reducing the critical slowing-down.

The worm algorithm and the directed-loop algorithm are also efficient near the critical point for a similar reason. In these cases, the estimator of Green's functions is the frequency of the head's visiting a certain location. For example, in order to compute the correlation function $\Gamma(X, Y) \equiv \langle S^x(X) S^x(Y) \rangle$, we have only to count the number of times the head passes the position $X-Y$ (relative to the original point). This is quite natural, considering that the trajectory of the head in the directed-loop algorithm is statistically identical to a loop in the loop algorithm in the cases where the two algorithms coincide. In what follows, we see that the estimator is valid in general even when the directed-loop algorithm does not coincide with the loop algorithm.

We start with eq. (34). When the worm weight is chosen as (44), the correlation function can be expressed as

$$(\Delta\tau\eta)^2 \Gamma(X, Y) = \frac{\sum_{S': X, Y} \tilde{W}_L(S')}{\sum_{S': \text{no worm}} \tilde{W}_L(S')},$$

where the summation in the numerator is over the states with the head at X and the tail at Y or the other way around. The number of times that we encounter a state S in the Monte Carlo simulation is proportional to $\tilde{W}_L(S)$. Therefore, the above expression can be rewritten as

$$(\eta)^2 \Gamma(X, Y) dX dY = \frac{\langle \Delta_{X,Y}(S) dX dY \rangle_{\text{MC}}}{\langle \Delta_\emptyset(S) \rangle_{\text{MC}}},$$

where $\Delta_{X,Y}(S) dX dY = 1$ if and only if one discontinuity point is in the interval dX centered at X and the other in the interval dY centered at Y . The other function $\Delta_\emptyset(S)$ is 1 if there is no worm in S . Now, we obtain

$$\begin{aligned} \Gamma(R) &\equiv \frac{1}{N\beta} \int dX dY \Gamma(X, Y) \delta(R - (X - Y)) \\ &= \frac{1}{N\beta\eta^2} \frac{\int dX \langle \Delta_{X, X+R}(S) \rangle_{\text{MC}}}{\langle \Delta_\emptyset(S) \rangle_{\text{MC}}} \\ &= \frac{1}{N\beta\eta^2} \langle n(R) \rangle_{\text{MC}}, \end{aligned}$$

where $n(R)$ is the average number of times the head passes, during a cycle, the point whose distance from the origin is R .

3. Numerical Recipe

In §2, the mathematical framework of several algorithms have been described. In the present section, we present detailed descriptions of the procedures for realizing the algorithms. Since the efficiencies of the algorithms have

been compared only for a very limited number of models, there is not much hope of presenting here a flawless recommendation as to which algorithm should be used for a given instance. However, there are some properties that we know already. For example, the loop algorithm is the best of all the algorithms, or at least not much worse than any other algorithm, in the cases where the relationship [such as (65)] between the relevant susceptibility and the cluster size holds. The XXZ model of an easy-axis anisotropy with no magnetic field, and the bilinear-biquadratic Heisenberg model are among these cases. When a finite magnetic field is present, however, the cluster size in the loop algorithm does not in general reflect the correlation length correctly, and it may perform poorer even than the local updating algorithm. The directed-loop algorithm and the worm algorithm work much better for such cases. It should be also pointed out that applications of these two algorithms are usually more straight-forward than that of the loop algorithm; for the loop algorithm we first need to find a good graphical decomposition of the Hamiltonian, which largely depends on the Hamiltonian under consideration. However, the two algorithms become essentially equivalent to the local updating algorithm for the models with the Ising-like anisotropy, and hence are not very efficient for reducing the critical slowing-down in the Ising-like models.

The description of the procedures in the following subsections is given mostly in graphical terms, such as *segments* and *vertices* defined in the first subsection. Therefore, we have to translate it into one of the computer languages. While the actual coding with a specific computer language is out of the scope of the present article, a remark on the data structure may be helpful here. The coding can be done, in principle, with any commonly-used computer language. However, the graphical objects that we have to deal with are created and annihilated during the simulation and their number varies. In addition, when we work with the infinite L limit, there is no discrete lattice that usually provides us with the index system of arrays. Therefore, we feel that some sort of a linked-list data structure is necessary, and that this data structure naturally fit in the object-oriented programming. While the object-oriented programming is possible with any language, working with object-oriented languages such as the C++ and the Java might make the programming easier. For example, the above-mentioned graphical entities may be most conveniently defined as objects, such as a "class" in the C++ language. A segment may be defined as an object that has some (or all) of the following member variables (i.e., properties): the local spin state (see §3.1 below), the spatial location, the beginning time, the ending time, (the pointers to) the vertices that delimit the segment, and a variable used for cluster identification (see Appendix B). In a sample program which may be found in our web site,³¹⁾ the graphical objects are handled through a container that has a built-in linked-list structure and is provided as a part of a standard library of the language.

3.1 Graphical terms for world-line Monte Carlo

Using the path-integral representation, our Monte Carlo simulation can be formulated as a Markov process in the space of graphical objects called *world-lines*. A world-line is

a curve on a $(d + 1)$ -dimensional space-time lattice, where d denotes the real-space dimension. The additional dimension, depicted as the vertical dimension throughout the present article, is called the imaginary-time dimension. The periodic boundary condition is imposed in this direction, while an arbitrary boundary condition may be used for the other (spatial) dimension. The height of the system, i.e., the system size in the imaginary-time direction is the inverse temperature $\beta = 1/k_B T$. Therefore, each site i in the real space is represented by a vertical line of the length β . Along each vertical line, an integral-valued function $n(i, \tau)$ is defined, which is constant (as a function of τ) almost everywhere and is discontinuous only at kinks. The value of the function is called the *local state* of the space-time point (i, τ) . A point at which the function is discontinuous is called a *kink*. A *spin configuration* (or simply a *configuration*) is the set of the functions along all the vertical lines. In other words, a configuration is equivalent to an assignment of integers to all the space-time points.

In particular, when the local state is binary, say 0 or 1, and the sum of the variables over the real space is conserved in the imaginary-time direction, which is the case with the $s = 1/2$ XXZ model, a configuration can be represented by a set of lines that connect the space-time points at which the local state is 1. These are the world-lines. In Fig. 21, such a world-line configuration of a one-dimensional quantum spin model is shown on a $(1 + 1)$ -dimensional space-time lattice.

For a quantum spin model of the spin size s , the integral variable $n(i, \tau)$ takes $2s + 1$ values from 0 to $2s$. The eigenstate of the operator $S_i^z + s$ is customarily used as the local spin variable, where S_i^z is the z component of the spin at the site i . We may also call it the *number of particles*, even when we are considering a spin model. Although configurations in the case $s > 1/2$ cannot be represented by the world-lines, we still refer to the configuration as a “world-line configuration”.

In addition to the local spin variables, we introduce auxiliary variables that can be represented conveniently by a graph that consists of vertical lines called *segments* and objects called *vertices* that delimit the segments. A vertex is represented by a graph element in the loop algorithm, whereas it is represented simply by a single horizontal line in the directed-loop algorithm. An example in the case of the loop algorithm is shown in Fig. 21, where a vertex is

represented by a pair of two horizontal lines.

3.2 Loop/cluster algorithm

The loop algorithm can be applied to various quantum spin models and gives much better performance than traditional quantum Monte Carlo methods.¹³⁾ The models that can be handled efficiently with this algorithm include the XXZ model with no magnetic field, the bilinear-biquadratic model, and the $SU(N)$ symmetric model as discussed below. Note, however, that it may not be efficient in the cases where terms in the Hamiltonian conflict with each other, such as the case of the anti-ferromagnetic XXZ model in an uniform magnetic field. In such cases, one should consider using the worm or the directed-loop algorithm discussed in the following subsections.

One cycle in a loop algorithm generally consists of the following operations: (i) assigning graphs to a given world-line configuration probabilistically; (ii) decomposing the world-line configuration into loops or clusters defined by graphs; (iii) assigning new values to integral variables on each loop or cluster probabilistically. The types of the graphs to be assigned and the probability of the assignment depend on the details of the model.

In this subsection, we describe the loop algorithm in detail for some characteristic models. Discussed in the following are (i) the quantum $s = 1/2$ XYZ spin model,^{2,28)} (ii) the quantum $s \geq 1$ XYZ spin model, (iii) the transverse field Ising model,⁶²⁾ (iv) the quantum $s = 1$ bilinear-biquadratic model,²⁹⁾ and (v) the quantum $SU(N)$ model.⁴⁴⁾

3.2.1 Quantum $s = 1/2$ XYZ spin model^{2,28)}

First we consider the loop algorithm for the $s = 1/2$ XYZ model

$$\mathcal{H} = - \sum_{\langle ij \rangle} \left(J_x \sigma_i^x \sigma_j^x + J_y \sigma_i^y \sigma_j^y + J_z \sigma_i^z \sigma_j^z \right) - B \sum_i \sigma_i^z, \quad (66)$$

where σ_i^α is an $s = 1/2$ spin operator in the α direction at the i th site, whose eigenvalues are $\pm 1/2$, and the summation is over all the pairs of interacting spins. In what follows, we consider the case, $J_x \geq J_y \geq 0$. (Other cases can be transformed to this case if the lattice is a bipartite lattice.) Figure 21 shows a world-line configuration of this model. On the world-line (the thick gray lines on the left panel of Fig. 21), the integral variable $n(i, \tau) = \sigma_i^z + 1/2$ takes 1, whereas it takes 0 elsewhere.

One sweep of the simulation with the loop algorithm consists of two phases: the graph assignment and the cluster flip. The graph assignment is done by (i) deleting the existing graph, (ii) assigning the graph elements to the uniform intervals (UI) with a certain density, and (iii) connecting the graph elements by vertical lines to form loops/clusters. Here, a UI for a pair of sites (ij) is an interval in which no change takes place in the local spin variable $n(i, \tau)$ or in $n(j, \tau)$. The cluster flip is done by (i) identifying the clusters, and (ii) flipping the clusters. The cluster identification is done as the assignment of a number to every segment so that the number uniquely specifies the cluster to which the segment belongs. The flipping of a cluster is simply the simultaneous inversion of the local spin states (from 0 to 1 or *vice versa*) for all the segments in the cluster. As shown in Table II, we need three types of graph

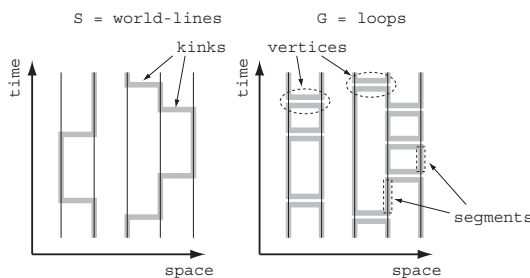









Fig. 21. Various graphical objects associated with a world-line configuration S and a graph G that match each other. Shown is the case of a one-dimensional system. A *vertex*, in this case, is the graph element that consists of a pair of two horizontal lines. A *kink* is located on the vertex at which the local state changes. A *segment* is a part of a vertical line that is delimited by two vertices.

Table II. The density of graph elements for uniform intervals (the second and the third columns), and the probability of choosing graph elements for kinks (the forth and the last), in the case where $0 \leq J_x \leq J_z$. The top row shows the local spin states, in which a solid (dashed) line denote an up (down) spin. The densities and the probabilities for all the other local states can be derived easily using the symmetries with respect to the time inversion, the exchange of the two sites, and the spin inversion. The first column shows graphs. The spins connected by a solid (dashed) line must be parallel (anti-parallel). The second and the third columns show the density of graph elements. The forth and the fifth column shows the probability of choosing the graph element on a kink.

state graph				
	$\frac{1}{4}(J_x + J_y)$	0	1	0
	$\frac{1}{4}(J_x - J_y)$	0	0	1
	$\frac{1}{2}(J_z - J_x)$	0	0	0

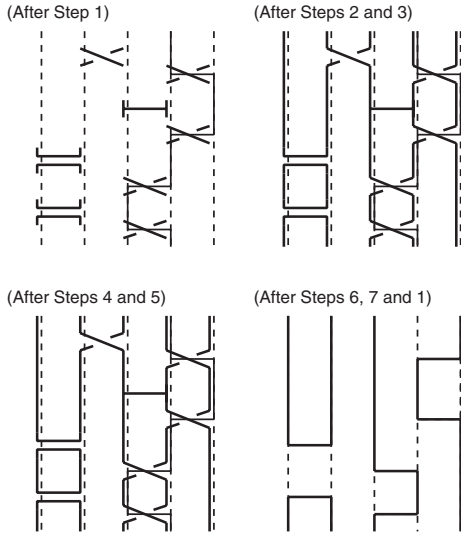


Fig. 22. The procedures of one step in the loop algorithm for the $s = 1/2$ XYZ model ($0 < J_x < J_z$).

elements for the loop algorithm for the XYZ model. The graph elements are called *cross*, *horizontal* and *binding* from the top to the bottom. Each graph element represents a certain constraint on the integral variables at four space-time points, namely, *legs*. The integral variables on the points connected by a line belong to the same loop (or cluster) and must be changed simultaneously when a loop is flipped. Although the graph elements in Table II are drawn as if they had a finite height for clarity of the illustration, they have in fact no temporal extension. The types of the graph elements as well as the density and the probability of the graph assignment depend on the anisotropy. There are four cases to be considered separately: (i) $J_z \geq J_x \geq 0$; (ii) $J_x \geq J_z \geq 0$; (iii) $J_z \leq 0, |J_z| \geq J_x$; (iv) $J_z \leq 0, J_x \geq |J_z|$.

To be specific, the updating procedure of the integral

Table III. The same as Table II for the case $0 \leq J_z \leq J_x$.








state graph				
	$\frac{1}{4}(J_z + J_y)$	0	$\frac{J_z + J_y}{J_x + J_y}$	0
	$\frac{1}{4}(J_x - J_y)$	0	0	1
	0	$\frac{1}{4}(J_x - J_z)$	$\frac{J_x - J_z}{J_x + J_y}$	0

Table IV. The same as Table II for the case $0 \geq J_z, J_x \leq |J_z|$.















state graph				
	0	$\frac{1}{4}(J_x - J_y)$	0	1
	0	$\frac{1}{4}(J_x + J_y)$	1	0
	0	$\frac{1}{2}(J_z - J_x)$	0	0

Table V. The same as Table II for the case $0 \geq J_z, |J_z| \leq J_x$.

state graph				
	$\frac{1}{4}(J_x - J_z)$	0	$\frac{J_x - J_z }{J_x + J_y}$	0
	0	$\frac{1}{4}(J_x - J_y)$	0	1
	0	$\frac{1}{4}(J_z + J_y)$	$\frac{ J_z + J_y}{J_x + J_y}$	0

variables $n(i, \tau)$ is as follows (see also Fig. 22):

Step 1 Delete the whole graph.

Step 2 For each pair of interacting sites (ij), do the following. Decompose the interval $(0, \beta)$ into UIs. Then for each UI, place graph elements randomly with the density given in Tables II–V. (See Appendix A for the procedure of generating the temporal positions for the placement of the graph elements.)

Step 3 Place a graph element on every kink with the probability given in Tables II–V.

Step 4 Draw vertical lines between two graph elements and connect two legs (one from each graph element). We refer to the resulting graph as G .

Step 5 Identify clusters. (For the identification algorithm, see Appendix B.)

Spep 6 For each cluster, flip the values of the variables $n(i, \tau)$ (i.e., $n(i, \tau) := 1 - n(i, \tau)$) for all the segments on it, with the probability $p(c) = \exp(-Bm_c)/(\exp(Bm_c) + \exp(-Bm_c))$. Here m_c is the cluster magnetization of the cluster c defined as

$$m_c \equiv \int_c dX \left(n(X) - \frac{1}{2} \right), \quad (67)$$

where $\int_c dX$ stands for the sum-integral on the cluster (the summation with respect to the site index and the integration with respect to the time).

Step 7 Do measurements. (For the estimators, see §3.2.6.)

3.2.2 Quantum XYZ spin model with large spins

For the XYZ model with spins larger than $s = 1/2$, we can in general use the split-spin technique presented below. The resulting algorithm are generally efficient when the corresponding algorithm for $s = 1/2$ are efficient. However, for models with the easy-plane anisotropy, such as the XY model, the coarse-grained algorithm described in §3.4.2 is recommended since there is no need for working with multiple split-spins for each site. On the other hand, for models with the easy-axis anisotropy with no magnetic field, the split-spin algorithm presented below, to our knowledge, is the best choice among the algorithms discussed in the present article.

We consider $2s$, instead of one, vertical lines at each site, each representing a split spin, or a Pauli spin that carries $s = 1/2$. Correspondingly, for each pair of nearest-neighbor sites, we apply the same graph-assignment procedure as in the $s = 1/2$ algorithm to each one of $(2s)^2$ combinations of split spins. In addition, in order to eliminate unphysical states, we stochastically assign a graph that represents a permutation among the $2s$ split spins, to the end points of the $2s$ vertical lines. (One example is shown in Fig. 8 for $s = 1$.) All the permutation graphs that match the current state is chosen with equal probability. For example, when the l split spins among $2s$ are up and the others are down, there are $l!$ ways to connect up-spins and $(\bar{l} \equiv 2s - l)!$ for down-spins. Therefore, every matching graph is chosen with the probability $1/(l!\bar{l}!)$.

Thus the split-spin algorithm for the XYZ model with an arbitrary s can be obtained by replacing Step 2 in §3.2.1 by

Step 2: For each pair of interacting split-spins, (i, μ) and (j, ν) , do the following. Decompose the interval $(0, \beta)$ into UIs. Then for each UI, place graph elements randomly with the density given in Tables II–V.

and inserting between Step 3 and Step 4 the following

(Insertion of the permutation graphs): For each site, connect $2s$ end points of vertical lines at $\tau = \beta$ to those at $\tau = 0$, pairwise, so that an up-spin is connected to an up-spin and likewise for down-spins. (Choose one of $(l!)(\bar{l}!)$ ways of connection with equal probability.)

3.2.3 Transverse Ising model⁶²⁾

Next we consider the transverse Ising model,

$$\mathcal{H} = -J \sum_{\langle ij \rangle} \sigma_i^z \sigma_j^z - B \sum_i \sigma_i^x \quad (68)$$

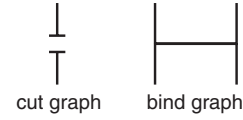


Fig. 23. The graphs for the transverse Ising model.

with $J \geq 0$ and $B \geq 0$. Since the transverse field breaks the conservation of the total magnetization, the world-lines are discontinuous; a world-line can terminate anywhere. Accordingly, we need a new type of graphs that cut segments at a single point. Therefore, we need two types of graph elements in this case as shown in Fig. 23.

Except for the difference in the types of the graph elements, the algorithm is almost the same as the one for the XYZ model described above. One cycle of the loop algorithm for the transverse-field Ising model is as follows:

Step 1 Delete the graph.

Step 2 For each pair of the interacting sites (ij) , do the following. Decompose the interval $(0, \beta)$ into UIs. Then for every UI in which the two spins are parallel (i.e., the local spin state on i is the same as that on j), place one of the graph elements randomly with the density $J/2$.

Step 3 For each site i (i.e., vertical line), do the following. Place “cut” graphs along the line with the density $B/2$.

Step 4 Place a “cut” graph on every kink.

Step 5 Draw vertical lines to connect legs of the graph elements.

Step 6 Identify the clusters.

Step 7 Flip every cluster independently with probability $1/2$.

Step 8 Do measurements. (For the estimators, see §3.2.6.)

3.2.4 Quantum $s = 1$ bilinear-biquadratic model²⁹⁾

The Hamiltonian for the model is

$$\mathcal{H} = - \sum_{\langle ij \rangle} (J_L S_i \cdot S_j + J_Q (S_i \cdot S_j)^2). \quad (69)$$

It is convenient to introduce the parameter θ by

$$J_L = -J \cos \theta, \quad J_Q = -J \sin \theta \quad (J > 0). \quad (70)$$

There is no negative-sign problem when the coupling constant J_Q is positive ($-\pi \leq \theta \leq 0$). We consider this case in what follows.

Using the split-spin technique, the original spin is decomposed into two $s = 1/2$ spins. In addition to the ordinary kinks, we have *double kinks* at which two particles jump to the neighboring site. (This is because of the biquadratic term.) The biquadratic term also requires new types of graph elements that have eight legs as shown in Fig. 24 in addition to the ordinary four-legged graph elements. (The third and the fourth graph elements in Fig. 24 are eight-legged representation of the ordinary single

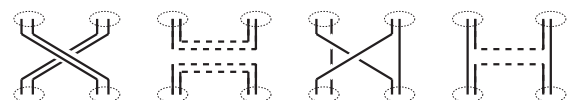













Fig. 24. The graph elements for quantum $s = 1$ bilinear-biquadratic model. As before, the spins connected by a solid (dashed) line must be parallel (anti-parallel).

Table VI. The density (a) and the probability (b) of the graph assignment for the bilinear-biquadratic model with $-\pi \leq \theta \leq -3\pi/4$.

(a)

state graph				
	$2(J_L - J_Q)$	$J_L - J_Q$	0	$J_L - J_Q$
	J_Q	J_Q	0	0

(b)

state graph			
	$1 - \frac{J_Q}{J_L}$	1	0
	$\frac{J_Q}{J_L}$	0	1

graphs, whereas the first and second ones are the new ones.) We call the first *double-cross* and the second *double-horizontal*.

From an algorithmic point of view, the region $-\pi \leq \theta \leq 0$ is decomposed into three sub-regions: $[-\pi, -3\pi/4]$, $[-3\pi/4, -\pi/2]$ and $[-\pi/2, 0]$. In the first region, we need only single-cross and double-cross graphs. In the third region, only single-horizontal and double-horizontal graphs are required. In the second region, no single graph is used. In this region, only double-cross and double-horizontal graphs are sufficient for decomposing the Hamiltonian. The procedure of a cycle can be summarized as follows.

Step 1 Delete the graph.

Step 2 For each pair of the nearest-neighbor sites (ij), do the following. First decompose the interval $(0, \beta)$ into UIs. Then, for each UI, and for each type of the graph elements, do the following. Place graph elements on the UI uniform-randomly with the density given in Tables VI–VIII. There are generally multiple ways of placing a graph element of a given type to a given position, i.e., there are multiple ways of wiring so that the spins connected by a solid (dashed) line are parallel (anti-parallel). Choose one of the consistent ways of wiring with equal probability.

Step 3 Place a graph element on every kink with the probability given in Tables VI–VIII. Again choose one of the consistent ways of wiring with equal probability.

Step 4 For each site, connect two end points of vertical lines at $\tau = \beta$ to those at $\tau = 0$, pairwise, so that an up-spin is connected to an up-spin and likewise for down-spins. Choose one of such ways of wiring with equal probability.







Step 5 Draw vertical lines so that each line connects a leg of a graph element to a leg of another.

Step 6 Identify clusters.

Step 7 Flip every cluster independently with probability $1/2$.

Table VII. The same as Table VI for the case $-3\pi/4 \leq \theta \leq -\pi/2$.

(a)

state graph				
	J_L	J_L	0	0
	0	$J_Q - J_L$	$J_Q - J_L$	0

(b)

















state graph			
	1	0	$\frac{J_L}{J_Q}$
	0	1	$1 - \frac{J_L}{J_Q}$

Table VIII. The same as Table VI for the case $-\pi/2 \leq \theta \leq 0$.

(a)

state graph				
	0	$-J_L$	$-2J_L$	$-J_L$
	0	J_Q	J_Q	0

(b)

state graph			
	1	$\frac{-J_L}{J_Q - J_L}$	0
	0	$\frac{J_Q}{J_Q - J_L}$	1

Step 8 Do measurements.

We have described the loop algorithm with the split-spin representation, in which all the loops are binary loops. In the region $-3\pi/4 \leq \theta \leq -\pi/2$, however, we can apply the loop algorithm with non-binary loops as described below.

In the algorithm with the non-binary loops, we do not need to split spins into $2s$ Pauli spins. We need two types of graph elements, cross and horizontal, as we use for the $s = 1/2$ XY model. However, each loop has three states, 0, 1 and 2, rather than two. Accordingly, the constraint imposed by the graph elements must be generalized; the local spin variables on the two points connected by a cross graph must be equal whereas those connected by the horizontal one must be complementary to each other (0 and 2, or 1 and 1). To be more specific, the procedure of a cycle is as follows:

Step 1 Delete the graph.

Step 2 For each pair of the interacting sites (ij), do the following. Decompose the interval $(0, \beta)$ into UIs. Place cross graphs with the density $\rho_C \equiv J_L$ to the UIs for which the local spin states on i and j are the same. Then, place horizontal ones with the density $\rho_H \equiv J_Q - J_L$ to the UIs for which the local spin states are complementary to each other.

Step 3 For each kink, place a graph (cross or horizontal) that matches the local state. The only local states that match both the graph elements are

$$\begin{pmatrix} l' & m' \\ l & m \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix}.$$

For these kinks, we choose the cross graph with the probability $\frac{J_L}{J_Q}$ and the horizontal graph with $1 - \frac{J_L}{J_Q}$. For kinks with the other local states, the matching graph element is unique.

Step 4 Draw vertical lines to connect legs of graph elements.

Step 5 Identify the clusters.

Step 6 Choose one of the three possible states (0, 1 or 2) for each loop with equal probability.

Step 7 Do measurements.

3.2.5 Quantum $SU(N)$ model⁴⁴⁾

An $SU(N)$ -invariant generalization of the Heisenberg model can be formally written as

$$H = \sum_{(ij)} H_{ij} = -J \sum_{(ij)} \sum_{\alpha, \beta} J_i^{\alpha\beta} J_j^{\beta\alpha}. \quad (71)$$

The symbols $J_i^{\alpha\beta}$ ($\alpha, \beta = 1, 2, \dots, N$) denote the generators of the $SU(N)$ algebra that satisfy

$$[J_i^{\alpha\beta}, J_j^{\mu\nu}] = \delta_{ij} (\delta_{\alpha, \nu} J_i^{\mu\beta} - \delta_{\mu, \beta} J_i^{\alpha\nu}). \quad (72)$$

We consider the model with the fundamental representation (and its dual representation) where the local Hilbert space is N -dimensional. The matrix elements of the pair Hamiltonian H_{ij} can be specifically written as

$$\langle l', m' | H_{ij} | l, m \rangle = \begin{cases} -J \delta_{l, m'} \delta_{l', m} & (J > 0) \\ J \delta_{l, \bar{m}} \delta_{l', \bar{m}'} & (J < 0) \end{cases}, \quad (73)$$

where $l, m, l', m' = 0, 1, \dots, N-1$ are the local ‘spin’ variables analogous to the ones used in the algorithms for the $SU(2)$ spin models, and $\bar{l} \equiv (N-1) - l$. Then, it can be easily shown that the Hamiltonian is expressed by a single type of graph elements; a cross graph if J is positive (ferromagnetic) or a horizontal graph if J is negative (antiferromagnetic). (Two special cases have been described in §3.2.4. The $s=1$ bilinear-biquadratic models at $\theta = -\pi/2$ and $\theta = -3\pi/4$ are the same as the present $SU(3)$ model with $J < 0$ and $J > 0$, respectively.)

The prescription for the $SU(N)$ model directly follows from this graphical expression:

Step 1 Delete the graph.

Step 2 For each pair of the interacting sites (ij), do the following. Decompose the interval $(0, \beta)$ into UIs. Place cross (horizontal) graphs with the density $|J|$ to the UIs for which the local spin states on i and j are the same (complementary to each other), if $J > 0$ ($J < 0$).

Step 3 For each kink, place a cross ($J > 0$) or a horizontal ($J < 0$) graph.

Step 4 Draw vertical lines to connect the legs of graph elements and form closed loops.

Step 5 Identify the loops.

Step 6 Choose one of the N possible states for each loop with equal probability.

Step 7 Do measurements.

3.2.6 Estimators

In order to obtain the thermal average of a quantity \hat{Q} , we compute the Monte Carlo averages of the corresponding estimator $E(S, G)$. In other words,

$$\langle \hat{Q} \rangle_{\text{thermal}} = \langle E(S, G) \rangle_{\text{MC}}.$$

While the correspondence between a quantity and an estimator is straightforward in many cases, it is not so obvious in some other cases. In the following, we present a list of the estimators of frequently computed quantities. Many estimators depends on the world-line configuration, S , only, whereas improved estimators depend only on the graph G . For the derivation, see §2.16.

Estimator of a diagonal operator: In this case, the estimator is simply

$$E_Q(S) \equiv Q(\psi(0)) \equiv \langle \psi(0) | \hat{Q} | \psi(0) \rangle,$$

where \hat{Q} is the diagonal operator and $\psi(0)$ is the spin configuration at $\tau = 0$. Because of the time-translational invariance, a better estimator is

$$E_Q(S) \equiv \frac{1}{\beta} \int_0^\beta d\tau Q(\psi(\tau)). \quad (74)$$

For a conserved quantity such as $\hat{Q} \equiv M^z \equiv \sum_i S_i^z$ in the XXZ quantum spin model, these two estimators are identical. In this particular case, the estimator is simply

$$E_{M^z}(S) = \sum_i \psi_i(0).$$

The time-dependent correlation function of two diagonal operators

$$\Gamma_{AB}(\tau, \tau') \equiv \langle \hat{A}(\tau) \hat{B}(\tau') \rangle$$

can be computed with the estimator

$$E_{AB}(S) \equiv A(\psi(\tau)) B(\psi(\tau')).$$

By integrating over the imaginary time, we obtain an estimator for the generalized susceptibility $\chi_{AB} \equiv \int_0^\beta d\tau \Gamma_{AB}(\tau, 0)$ as

$$E_{\chi_{AB}}(S) \equiv \beta^{-1} \left(\int_0^\beta d\tau A(\psi(\tau)) \right) \left(\int_0^\beta d\tau B(\psi(\tau)) \right). \quad (75)$$

Energy and Specific Heat: The estimator for the total energy, $\langle \mathcal{H} \rangle$, is

$$E_{\mathcal{H}}(S) = E_{\text{diag}(\mathcal{H})}(S) - \frac{1}{\beta} n_{\text{kink}}(S), \quad (76)$$

where $n_{\text{kink}}(S)$ is the total number of kinks in S , and $E_{\text{diag}(\mathcal{H})}(S)$ is the estimator for the diagonal part of the Hamiltonian, which is defined by (74) with Q being $\text{diag}(\mathcal{H})$. The specific heat is not measured with a single estimator. Instead, it is computed using the following expression.

$$C = \beta^2 \left[\langle E_{\text{diag}(\mathcal{H})}^2 \rangle_{\text{MC}} - \langle E_{\text{diag}(\mathcal{H})} \rangle_{\text{MC}}^2 \right] + \langle n_{\text{kink}}^2 \rangle_{\text{MC}} - \langle n_{\text{kink}} \rangle_{\text{MC}}^2 - \langle n_{\text{kink}} \rangle_{\text{MC}}. \quad (77)$$

Improved operator: For the diagonal magnetic susceptibility,

$$\chi_{zz} \equiv \int_0^\beta d\tau \langle M^z(\tau) M^z(0) \rangle_{\text{thermal}}$$

at zero field, the following graphical estimator is often useful. For the cases where the clusters can be flipped with probability 1/2, i.e., if no field breaks the up-down symmetry, the estimator is

$$E_{\chi_{zz}}(G) = \frac{1}{\beta} \sum_c M_c^2, \quad (78)$$

where the summation is over all the clusters in G , and M_c is the cluster magnetization

$$M_c \equiv \left| \int_c dX \psi(X) \right|.$$

The symbol $\int_c dX$ denotes the summation/integration over the cluster c .

Non-diagonal susceptibility: The susceptibility of the spin components perpendicular to the quantization axis, i.e.,

$$\chi_{xx} \equiv \int_0^\beta d\tau \langle M^x(\tau) M^x(0) \rangle_{\text{thermal}}$$

can be measured for the XXZ spin model with the estimator

$$E_{\chi_{xx}}(G) = \frac{1}{4\beta} \sum_c V_c^2, \quad (79)$$

where V_c is the cluster volume

$$V_c \equiv \int_c dX 1.$$

3.3 Worm algorithm³⁾

The loop algorithm often becomes very inefficient (worse than the local updating algorithm) when the Hamiltonian contains some terms that conflict with each other. Such is the case in the anti-ferromagnetic XXZ model in a uniform magnetic field. This difficulty can be removed by introducing discontinuity points in the world-lines. However, the XXZ models with an easy-axis anisotropy should not be dealt with the worm algorithm if the magnetic field is not conflicting with the exchange couplings, since the loop algorithm usually performs much better in such cases especially near the transition point.

A cycle in the worm algorithm consists of a creation and an annihilation of a worm, and a vertical move, a jump and an anti-jump of the head. (See Fig. 9.) A head or a tail of the worm is called positive (negative) if the local state above it is greater (smaller) than the local state below by one. If the head is positive the tail must be negative and *vice versa*. When the positive one is created above the negative one, the worm is called a “lowering” worm since the local state between the two discontinuity points is lower than the original state by one (Fig. 12). A “raising” worm is the one

in which the positive discontinuity point is below the negative one.

In what follows, we describe the worm algorithm for the generic Hamiltonian

$$\mathcal{H} = \sum_{(ij)} (U_{ij} + V_{ij}) - \eta \sum_i Q_i, \quad (80)$$

where $V_{ij} \equiv \text{diag}(\mathcal{H}_{ij})$ is the diagonal part of the pair Hamiltonian, $U_{ij} \equiv \mathcal{H}_{ij} - V_{ij}$ is the off-diagonal part, and Q_i is an operator defined on the site i that has no diagonal part. The last term is a source term which is included only for a technical purpose. The constant η is any finite real number of $O((N\beta)^{-1/2})$ where N is the total number of sites in the whole system.

The procedure of one cycle, starting from a state with no worm, is the following:

Step 1-1 [Creation (Fig. 9(a))] Choose one of the two types (raising or lowering) of the worm with equal probability.

Step 1-2 Choose a point from the whole space-time. (In what follows, we consider a continuous part of a vertical line delimited by kinks in which the site under consideration is involved. The head and the tail of the worm themselves are also regarded as kinks here. We call such a part, a single-site UI.) We denote the single-site UI in which the chosen point resides by $I \equiv \{(i, \tau) | \tau \in [t_1, t_2]\}$.

Step 1-3 Choose two points (i, τ_1) and (i, τ_2) from I so that $\tau_1 < \tau_2$ with the probability density

$$p_c(\tau_1, \tau_2) = \frac{\exp[-\overline{\Delta V} \times (\tau_2 - \tau_1)]}{A}. \quad (81)$$

These points are candidates for the positions at which a worm may be created. The constant A is the normalization factor and $\overline{\Delta V}$ is the average excess axion per unit time defined as

$$\overline{\Delta V} \equiv \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} dt \sum_{j \in \delta(i)} \Delta V_{ij}^{\text{create}}(t), \quad (82)$$

where $\delta(i)$ denotes the set of the nearest neighbors of i , and $\Delta V_{ij}^{\text{create}}(t)$ is the increase in $V_{ij}(t) \equiv \langle \psi(t) | V_{ij} | \psi(t) \rangle$ that occurs if the two discontinuity points are created at the bottom and the top of I .

Step 1-4 Accept the proposed creation of a worm at τ_1 and τ_2 with the probability $\min(1, R_{\text{create}})$ where R_{create} is defined as

$$R_{\text{create}} \equiv \frac{2N\beta\eta^2 A}{t_2 - t_1} \times \exp \left[\int_{\tau_1}^{\tau_2} dt \left(\overline{\Delta V} - \sum_{j \in \delta(i)} \Delta V_{ij}^{\text{create}}(t) \right) \right] \times \langle \psi'(\tau_1 + 0) | Q_i | \psi'(\tau_1 - 0) \rangle \times \langle \psi'(\tau_2 + 0) | Q_i | \psi'(\tau_2 - 0) \rangle, \quad (83)$$

where ψ' is the state after the creation was accepted. If the proposal is rejected, go to Step 6.

Step 1-5 Choose one of the two discontinuities with equal probability, and make it the head.

Step 2 [Vertical Move (Fig. 9(b))] Move the head along the UI where it currently resides. The new time τ is chosen with the probability density

$$p_{\text{vertical}}(\tau) \equiv \frac{\exp\left[-\sum_{j \in \delta(i)} \Delta V_{ij}^{\text{move}}(\tau)\right]}{B}, \quad (84)$$

where B is the normalization factor and $\Delta V_{ij}^{\text{move}}(\tau)$ is the increase in the diagonal contribution $\int_{\text{UI}} dt V_{ij}(t)$ that occurs if the vertical move to the time τ is accepted.

Step 3 [Jump (Fig. 9(c))] For each nearest-neighbor site j of the current site i , consider two intervals; the one delimited by the head itself and the nearest kink above it, and the other delimited by the head and the nearest kink below it. (Here, we only consider kinks in which the sites i and/or j are involved.) For each of the intervals, do Steps 3-1 and 3-2.

Step 3-1 Generate a time τ , which is the candidate for the temporal position of placing a kink, with the probability density

$$p_{\text{jump}}(\tau) \equiv \frac{\exp(-\Delta V_{\text{jump}}(\tau))}{C}, \quad (85)$$

where C is the normalization factor and $\Delta V_{\text{jump}}(\tau)$ is the increase in the diagonal contribution to the total weight that occurs if the kink is created at the proposed time τ . Specifically,

$$\Delta V_{\text{jump}}(\tau) \equiv \int_{t_1}^{t_2} dt \left[\sum_{k \in \delta(i)} \Delta V_{ik}(t) + \sum_{k \in \delta(j)} \Delta V_{jk}(t) - \Delta V_{ij}(t) \right],$$

where t_1 and t_2 denote the starting and the ending time of the interval, respectively, and $\Delta V_{ij}(t)$ is the increase in the $V_{ij}(t)$ after the jump was accepted.

Step 3-2 Let the head jump from i to j at τ , creating a kink there, with the probability $\min(1, R_{\text{jump}})$. (Otherwise, do nothing.) Here, R_{jump} is defined as

$$R_{\text{jump}} \equiv C \frac{\langle \psi'(\tau+0) | U_{ij} | \psi'(\tau-0) \rangle}{\langle \psi'(\tau_w+0) | Q_j | \psi'(\tau_w-0) \rangle} \times \frac{\langle \psi'(\tau_w+0) | Q_j | \psi'(\tau_w-0) \rangle}{\langle \psi(\tau_w+0) | Q_i | \psi(\tau_w-0) \rangle}, \quad (86)$$

where ψ' is the state after the jump was accepted and τ_w is the head's current temporal position.

Step 4 [Anti-Jump (Fig. 9(c))] Consider two directions, upward and downward. For each nearest-neighbor site j , and for each direction, do the following. If the kink, involving i and/or j and the nearest to the head in the chosen direction, is not the one between i and j , do nothing and go to Step 5. Otherwise, we consider the second nearest kink among those which involves i and/or j , and the interval delimited by it and the head itself. Let the head anti-jump, i.e., let it jump from j to i so that the kink between i and j is annihilated, with the probability $\min(1, R_{\text{anti-jump}})$, where $R_{\text{anti-jump}}$ is the reciprocal of (86) with ψ' being the current state, ψ the state that the anti-jump would result in, and τ the temporal position of the nearest kink to be erased by the anti-jump.

Step 5 [Annihilation (Fig. 9(a))] If the head and the tail are located on the same site and there are no kinks between the two, annihilate the worm with the probability $\min(1, R_{\text{annihilate}})$ where $R_{\text{annihilate}}$ is the reciprocal of

(83) with ψ' being the current state, ψ the state that the annihilation would result in, and τ_1 and τ_2 the temporal positions of the head and the tail. Go to Step 6. If annihilation is not chosen, go to Step 2.

Step 6 Do measurements. (The end of the cycle.)

3.4 Directed-loop algorithm

The directed-loop algorithm can be thought of as a generalization of the worm algorithm. Therefore, remarks similar to the ones presented at the beginning of the last subsection apply to the directed-loop algorithm. As for the simulations of XXZ spin model, the directed algorithm should be used in the cases with the isotropic couplings or the easy-plane couplings, with or without a magnetic field. For the easy-axis couplings, the loop algorithm should be used.

In the directed-loop algorithm proposed by Syljuåsen and Sandvik,⁴⁾ the spin configuration is updated through movements of the worm as in the worm algorithm. One “sweep” of the directed-loop algorithm consists of an assignment of vertices and a number of cycles of worm update. Vertices are represented as horizontal lines connecting nearest neighbor sites. Unlike the worm algorithm, the head of the worm has a direction of motion and it can only move in this direction. (Fig. 13 shows a local configuration in which a worm and vertices are involved.) It can change the direction of motion and its spatial location only when it hits a vertex. One cycle of the worm update, therefore, consists of the creation of a worm, the vertical movements in the direction of motion, the scatterings at vertices, and the annihilation of the worm.

3.4.1 Directed-loop algorithm for the $s = 1/2$ XXZ model

In the present subsection, we specialize in the description of the $s = 1/2$ XXZ model,

$$\mathcal{H}_{ij} = -J(\sigma_i^x \sigma_j^x + \sigma_i^y \sigma_j^y) - J' \sigma_i^z \sigma_j^z - \frac{h}{2}(\sigma_i^z + \sigma_j^z) - E_0$$

with $J > 0$ and $h \geq 0$. The whole parameter space is divided into six regions as shown in Fig. 25. The constant E_0 is chosen as $E_0 = (J - h)s^2 + hs$ for the regions I and V, $E_0 = -J's^2 + hs$ for the regions II, III, and IV, and $E_0 = J's^2 + h(s - 2s^2)$ for the region VI. (While $s = 1/2$ in the present case, the same expression can be used for the general s case discussed in the next section.) Within each region, the scattering probability and the vertex density are simple analytic functions of J , J' and h .

When the head hits a vertex, one of four ways of

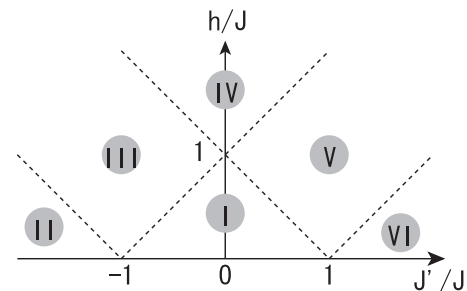


Fig. 25. The six regions in the parameter space of the XXZ model.

Table XI. The coarse-grained algorithm for the quantum XXZ spin model with arbitrary s . The density of vertices, ρ , and the scattering probabilities of the head $P(\Gamma|\Sigma)$ are shown. The symbol h denotes the magnetic field per pair of Pauli spins, which is related to the magnetic field per original spin, H , by $h = H/(2ds)$ for the d -dimensional hyper cubic lattice. ($\bar{l} \equiv 2s - l$, $\bar{m} \equiv 2s - m$.)

Σ		Region I	Region II	Region III	Region IV	Region V	Region VI
$\left(\begin{smallmatrix} l & m \\ l_- & m \end{smallmatrix}\right)$	$\rho(\Sigma) =$	A	B	B	B	A	C
$\left(\begin{smallmatrix} l & m \\ l_- & m \end{smallmatrix}\right)$	$P(\downarrow \Sigma) =$	0	$\frac{\bar{m}(-J-J'-h)}{2B}$	0	0	0	$\frac{\bar{m}(-J+J'-h)}{2C}$
	$P(\nearrow \Sigma) =$	$\frac{\bar{m}(J+J'-h)}{4A}$	0	0	0	$\frac{\bar{m}(J+J'-h)}{4A}$	$\frac{\bar{m}J}{2C}$
	$P(\rightarrow \Sigma) =$	$\frac{m(J-J'-h)}{4A}$	$\frac{mJ}{2B}$	$\frac{m(J-J'-h)}{4B}$	0	0	0
$\left(\begin{smallmatrix} l & m \\ l_+ & m \end{smallmatrix}\right)$	$P(\downarrow \Sigma) =$	0	$\frac{\bar{m}(-J-J'+h)}{2B}$	$\frac{\bar{m}(-J-J'+h)}{2B}$	$\frac{m(-J+J'+h)+\bar{m}(-J-J'+h)}{2B}$	$\frac{m(-J+J'+h)}{2A}$	$\frac{m(-J+J'+h)}{2C}$
	$P(\nearrow \Sigma) =$	$\frac{m(J+J'+h)}{4A}$	0	$\frac{m(J+J'+h)}{4B}$	$\frac{mJ}{2B}$	$\frac{mJ}{2A}$	$\frac{mJ}{2C}$
	$P(\rightarrow \Sigma) =$	$\frac{\bar{m}(J-J'+h)}{4A}$	$\frac{\bar{m}J}{2B}$	$\frac{\bar{m}J}{2B}$	$\frac{\bar{m}J}{2B}$	$\frac{\bar{m}(J-J'+h)}{4A}$	0
$\left(\begin{smallmatrix} l+1 & m \\ l_- & m+1 \end{smallmatrix}\right)$	$P(\downarrow \Sigma) =$	0	0	0	0	0	0
	$P(\nearrow \Sigma) =$	$\frac{J+J'+h}{l-2J}$	0	$\frac{J+J'+h}{l-2J}$	$\frac{1}{l}$	$\frac{1}{l}$	$\frac{1}{l}$
	$P(\rightarrow \Sigma) =$	$\frac{J-J'-h}{l-2J}$	$\frac{1}{l}$	$\frac{J-J'-h}{l-2J}$	0	0	0
$\left(\begin{smallmatrix} l-1 & m \\ l_+ & m-1 \end{smallmatrix}\right)$	$P(\downarrow \Sigma) =$	0	0	0	0	0	0
	$P(\nearrow \Sigma) =$	$\frac{J+J'-h}{l-2J}$	0	0	0	$\frac{J+J'-h}{l-2J}$	$\frac{1}{l}$
	$P(\rightarrow \Sigma) =$	$\frac{J-J'+h}{l-2J}$	$\frac{1}{l}$	$\frac{1}{l}$	$\frac{1}{l}$	$\frac{J-J'+h}{l-2J}$	0
$\left(\begin{smallmatrix} l+1 & m \\ l_+ & m+1 \end{smallmatrix}\right)$	$P(\downarrow \Sigma) = P(\nearrow \Sigma) = P(\rightarrow \Sigma) = 0, \quad \text{and} \quad P(\uparrow \Sigma) = 1$						
$\left(\begin{smallmatrix} l-1 & m \\ l_- & m-1 \end{smallmatrix}\right)$	$P(\downarrow \Sigma) = P(\nearrow \Sigma) = P(\rightarrow \Sigma) = 0, \quad \text{and} \quad P(\uparrow \Sigma) = 1$						
$A \equiv \frac{1}{4}[lm(J+J'+3h) + (l\bar{m} + \bar{l}m)(J-J'+h) + \bar{l}\bar{m}(J+J'-h)]$ $B \equiv lmh + (l\bar{m} + \bar{l}m)\frac{-J'+h}{2}, \quad C \equiv \frac{1}{2}[lm(J'+h) + \bar{l}\bar{m}(J'-h)]$							

detailed balance condition. In §2.11, we have explained how we can use a solution to (43) for $s = 1/2$ to obtain a solution for larger spins.⁴³⁾ Since the whole parameter space (i.e., J - J' - H space) is divided in the six regions in the solution for $s = 1/2$ (Fig. 25), the same division applies to the larger spins. It should be also noted that the turning-back probability in Region I is always zero for any s . The directed-loop algorithm for large spins is different from that for $s = 1/2$ in the creation and the annihilation of the worm. Otherwise, the procedure is the same as the one in the $s = 1/2$ case described in §3.4.1. Therefore, one Monte Carlo step of the coarse-grained algorithm for spin $s > 1/2$ can be obtained by replacing the Table X by Table XI, and Step 3-1 and Step 3-4' by the following operations:

Step 3-1 Choose a point in the whole space-time uniform-randomly, and create a worm there. Then, choose the initial type of the worm, raising or lowering, with the probability $\bar{l}/(2s)$ or $l/(2s)$, respectively. Here $l = 0, 1, \dots, 2s$ is the spin-state variable at the chosen point, and $\bar{l} \equiv 2s - l$.

and

Step 3-4' Let the head annihilate with the tail, or let it go through. The probability of the annihilation depends on the type of the worm and the local spin state l between the head and the tail just before the collision. If the worm is of the raising type, the annihilation probability is l^{-1} , while it is $(\bar{l})^{-1}$ otherwise. If the head goes through, go back to Step 3-3. If the worm is annihilated, go to Step 4.

3.4.3 Soft-core boson model with repulsive interactions

It was pointed out that the algorithm for general s presented above can be used for bosonic models by taking the $s \rightarrow \infty$ limit.⁴⁷⁾ Here, we consider the tight-binding soft-

core boson model (or the boson Hubbard model)

$$\mathcal{H} = - \sum_{\langle ij \rangle} \left[\frac{t}{2} (b_i^\dagger b_j + b_i b_j^\dagger) - V_1 n_i n_j \right] - \sum_i \left[\mu n_i - \frac{V_0}{2} n_i (n_i - 1) \right], \quad (87)$$

where b_i and b_i^\dagger are an annihilation and a creation operator, respectively, on the site i , and $n_i \equiv b_i^\dagger b_i$. We here assume that $V_0 \geq 0$ and $V_1 \geq 0$. The model can be considered as a spin model with $s = \infty$. In the present algorithm, the state is updated in much the same way as the worm algorithm. A cycle consists of (i) the creation of a worm, (ii) the movements of the head, and (iii) the annihilation of the worm. We do not use vertices explicitly. Instead, “scattering” points are dynamically created with some density ahead of the head, and it is scattered at the nearest scattering point. One cycle of the coarse-grained algorithm for the soft-core boson model can be stated as follows:

Step 1 Choose a point uniform-randomly in the whole space-time, and create there a worm. (One is the head and the other is the tail.) The initial type of the worm is raising (Fig. 12).

Step 2 Choose the initial direction of motion, upward or downward, with equal probability.

Step 3 For each nearest-neighbor site j to the current site i , consider the UI between i and j ahead of the head. For each one of the four scattering directions Γ , generate a time $\tau_{j,\Gamma}$ in the UI. Here $\tau_{j,\Gamma}$ is the closest to the current temporal position of the head among those generated with the density given in Table XII. When this is done for all nearest neighbors j and all directions Γ , choose from the $\tau_{j,\Gamma}$ s the closest to the current temporal position. (Let it be τ_{j_0,Γ_0} .)

Table XII. The densities of scattering points $\rho(\Gamma|\Sigma)$ and the scattering probability at kinks $P(\Gamma|\Sigma)$ for a local state Σ . The densities are shown only in the case where the head is moving upward. (The densities in the other case can be obtained simply by changing the sign of the head.) The Γ specifies the direction of the head after the scattering. The z denotes the coordination number, e.g., $z = 2d$ for the d -dimensional hyper-cubic lattice.

Σ	$\mu/dt \leq -1$	$-1 \leq \mu/dt \leq 1$	$1 \leq \mu/dt$
$(n_- \quad m)$	$\rho(\downarrow \Sigma) = -\frac{t}{2} + \frac{nV_0 - \mu}{z} + mV_1$ $\rho(\nearrow \Sigma) = \frac{t}{2}$ $\rho(\rightarrow \Sigma) = 0$	$\frac{nV_0}{z} + mV_1$ $\frac{1}{2} \left(\frac{t}{2} - \frac{\mu}{z} \right)$ 0	$\max \left(0, \frac{t}{2} + \frac{nV_0 - \mu}{z} + mV_1 \right)$ 0 0
$(n_+ \quad m)$	$\rho(\downarrow \Sigma) = 0$ $\rho(\nearrow \Sigma) = 0$ $\rho(\rightarrow \Sigma) = 0$	0 0 $\frac{1}{2} \left(\frac{t}{2} + \frac{\mu}{z} \right)$	$\max \left(0, -\frac{t}{2} - \frac{(n-1)V_0 - \mu}{z} - mV_1 \right)$ 0 $\frac{t}{2}$
$\begin{pmatrix} n-1 & m \\ n_+ & m-1 \end{pmatrix}$	$P(\downarrow \Sigma) = 0$ $P(\nearrow \Sigma) = \frac{1}{n}$ $P(\rightarrow \Sigma) = 0$ $P(\uparrow \Sigma) = 1 - \frac{1}{n}$	0 $\frac{1}{n} \left(\frac{1}{2} - \frac{\mu}{zt} \right)$ $\frac{1}{n} \left(\frac{1}{2} + \frac{\mu}{zt} \right)$ $1 - \frac{1}{n}$	0 0 $\frac{1}{n}$ $1 - \frac{1}{n}$
$\begin{pmatrix} n-1 & m \\ n_- & m-1 \end{pmatrix}$	$P(\downarrow \Sigma) = P(\nearrow \Sigma) = P(\rightarrow \Sigma) = 0$, and $P(\uparrow \Sigma) = 1$		
$\begin{pmatrix} n+1 & m \\ n_+ & m+1 \end{pmatrix}$	$P(\downarrow \Sigma) = P(\nearrow \Sigma) = P(\rightarrow \Sigma) = 0$, and $P(\uparrow \Sigma) = 1$		
$\begin{pmatrix} n+1 & m \\ n_- & m+1 \end{pmatrix}$	$P(\downarrow \Sigma) = P(\nearrow \Sigma) = P(\rightarrow \Sigma) = 0$, and $P(\uparrow \Sigma) = 1$		

Step 4 Compare the nearest scattering point generated in Step 3, the nearest kink ahead of the head, and the tail if it resides on the same site as the head. If the nearest among these three is the scattering point, go to Step 5. If it is the kink, go to Step 5'. If it is the tail, go to Step 6.

Step 5 Let the head scatter as specified by j_0 and Γ_0 . Go to Step 3.

Step 5' Choose the direction of the scattering with the probability $P(\Gamma|\Sigma)$ given in Table XII, and let the head scatter. Go to Step 3.

Step 6 If the head is negative and comes back to the tail from below, or if it is positive and comes back from above, let it pass the tail with the probability 1 and go to Step 3. Otherwise, let it pass with the probability $1 - 1/n$ where n is the number of the particles ahead of the head just before the collision. Go to Step 3.

Step 7 Let the worm annihilate.

Step 8 Do measurements.

3.4.4 Observables

In the directed-loop algorithm (and also in the algorithm with the series-expansion representation), the energy and the specific heat can simply be computed by counting the number of the vertices,⁴⁾ n_v , as

$$\langle \mathcal{H} \rangle = -\beta^{-1} \langle n_v \rangle$$

and

$$C = \langle n_v^2 \rangle - \langle n_v \rangle^2 - \langle n_v \rangle.$$

The off-diagonal Green's function $\Gamma(\mathbf{r}, \tau)$ is, as in the worm algorithm,³⁾ proportional to the frequency of the occurrence of the configurations in which the head is separated from the tail by the space-time vector $\mathbf{x} = (\mathbf{r}, \tau)$. Specifically, Green's function of the bosonic models, for which the source term is proportional to $b + b^\dagger$, can be expressed as

$$\langle b(\mathbf{x})b^\dagger(\mathbf{y}) \rangle = \langle n(\mathbf{x} - \mathbf{y}) \rangle \quad (88)$$

where N is the number of sites and $n(\mathbf{x})$ is the number of times the configurations with the head and the tail separated from each other by \mathbf{x} appear during one cycle of the update (i.e., from the creation through the annihilation of the worm). See Dorneich and Troyer⁶³⁾ for the measurement of the time-dependent Green's function in the series-expansion representation with finite L .

4. Summary and Future Problems

We have reviewed recent developments in the Monte Carlo simulation methods based on Markov processes in the space of the world-line configurations. A few original results are also included; the non-binary algorithm for the bilinear-biquadratic model (§2.7, §3.2.4) with the SU(2) symmetry, the worm-scattering probability of the coarse-grained algorithm for the boson Hubbard model (§3.4.3), and the continuous-time formulation of the extended ensemble method (§2.15). We have focused on the three methods of updating the world-line configurations; the loop algorithm, the worm algorithm, and the directed-loop algorithm. The detailed descriptions of the numerical procedures have been given. The methods described solve, to a large extent, the problems in the local updating algorithm. The solved (or eased) problems include (i) the slowing-down near the critical point or the zero temperature, (ii) the discretization slowing-down (the Wiesler freezing), (iii) the systematic error due to the discretization, (iv) the non-ergodicity due to the artificially conserved quantities (or the additional slowing-down due to the *ad hoc* global flips introduced for ergodicity), (v) the computation of off-diagonal quantities, (vi) the freezing due to the external magnetic field competing against the exchange couplings, and (vii) the negative-sign problem in a special case of spinless fermions. On the other hand, a number of problems remain to be solved.

Particularly important among them are (i) the general fermion sign problem, (ii) the general frustration sign problem, (iii) the models with interaction terms competing against each other, such as the spin model with a strong anisotropy (uniaxial, cubic, tetragonal, etc.), and (iv) the models for which the order parameter cannot be expressed as a simple sum of local operators. The nature of the negative-sign problem may be quite different, depending on its origin. Whether it is due to the fermion sign or the frustration, however, there is not even a clue to a general solution. It is possible that a general solution does not even exist. An example of the problems (iii) and (iv) can be found in the $s > 1$ Heisenberg models with the cubic anisotropy. While such an anisotropy is quite common in real materials, we are not aware of any efficient algorithms. For example, the model

$$\mathcal{H} = -J \sum_{(ij)} \mathbf{S}_i \cdot \mathbf{S}_j - D \sum_i ((S_i^x)^4 + (S_i^y)^4 + (S_i^z)^4).$$

with positive J and D does not cause the negative-sign problem when the ordinary S^z representation basis is used. In addition, it is easy to find a graphical decomposition such as (19) for the model.⁶⁴⁾ However, the auto-correlation time of the simulation is extremely long at low temperatures and the configuration is practically frozen. A similar situation can be seen in another model with the cubic anisotropy

$$\mathcal{H} = -J \sum_{(ij)} \mathbf{S}_i \cdot \mathbf{S}_j - D \sum_{(ij)} ((S_i^x)^2 (S_j^x)^2 + (S_i^y)^2 (S_j^y)^2 + (S_i^z)^2 (S_j^z)^2).$$

Again this model does not cause the negative sign problem when the S^z representation basis is used. However, when we use a representation basis in which the model's symmetry is manifest, i.e., when the three vectors $|\pm\rangle \equiv |\uparrow\rangle \pm |\downarrow\rangle$ and $|0\rangle$ are used as the basis set, the model exhibits negative signs while the computational auto-correlation time is small in this basis. The Ising spin-glass problems may be considered as another example of the problem (iii). It is well-known that the auto-correlation time of this model is so long that an accurate numerical simulation is extremely difficult near and below the critical temperature. However, when the S^x representation basis is used, it is easy to find a simulation method with a small auto-correlation time, although the negative-sign problem appears in the S^x representation basis. These facts appear to suggest that the difficulty of the problems (iii) and (iv) are closely related to (ii) in general.

Acknowledgment

N.K.'s work was supported by the Grant-in-Aid (Program No. 14540361) from the Ministry of Education, Culture, Sports, Science and Technology, Japan. K.H.'s work was supported by the 21st Century COE Program "Center of Excellence for Research and Education on Complex Functional Mechanical Systems" of the Ministry of Education, Culture, Sports, Science and Technology, Japan. Some computation was done on SGI 2800 at ISSP, University of Tokyo. The authors are grateful to N. Hatano for his critical reading of the draft and many helpful comments. They also thank F. Alet, H.-G. Evertz, N. Prokov'ev, A. Sandvik, J.

Šmakov, B. Svistunov, and M. Troyer for stimulating conversations and/or useful comments.

Appendix A: Generation of Temporal Positions in Graph (or Vertex) Assignment

If a stochastic process consists of a time series of events with a given density and if each event occurs independently, the process is a Poisson process. (Specifically in the present context, these events correspond to graph-elements, vertices or scattering points.) In a Poisson process, a time interval between successive events obeys the exponential distribution. When the density of events is n , the distribution of the intervals is

$$p(\Delta t) = \begin{cases} n \exp(-n\Delta t) & (\Delta t > 0) \\ 0 & (\Delta t \leq 0) \end{cases}. \quad (\text{A.1})$$

We can generate a random variable Δt that obeys this distribution from the uniform random variable $r \in (0, 1]$ by using the transformation

$$\Delta t = -\frac{\ln(r)}{n}. \quad (\text{A.2})$$

Therefore, placing events (objects) on a given time-window (segment) with a given density n can be done as follows:

- Step 1** Set the time variable t to be the starting time of the window.
- Step 2** Generate a uniform random number $r \in (0, 1]$ and compute Δt using (A.2).
- Step 3** Increase the time t by Δt , i.e., $t := t + \Delta t$.
- Step 4** If the new time t is smaller than the ending time of the window, place an object at t , and go back to Step 2. Otherwise, terminate the process.

Appendix B: Cluster Identification

In an actual computer program, the world-line configuration is represented as a linked-list data structure of objects, where each object corresponds to a segment. In order to identify loops and clusters, we define a variable, which will eventually be the cluster ID number, for each object. (In the C++ language, for example, we add a member variable to the "segment" object.) When a graph element is assigned and points are connected, the variables are updated as follows.

Let $c(s)$ be the variable of the object that is specified (or is pointed to) by an index (or by a pointer) s . In what follows, we identify an index with the object that is specified by it. The variables define a tree structure. That is, $c(s)$ is a parent of s , and $c(s) = s$ if it does not have a parent. Every object initially has no parent. When two segments, s and s' , are connected by an edge in a graph element, the following operations are applied:

- Step 1** Find the root of each object. Let r and r' be the roots of s and s' , respectively. This can be done by repeating $r := c(r)$ starting from $r := s$ until $r = c(r)$ holds. The same for r' .
- Step 2** Let $R := \min(r, r')$. Then, let $c(a) := R$ for all a , where a are the ancestors of s and s' .

After these procedures have been done for all connections, $c(s)$ is the unique identifier of the cluster, i.e., $c(s) = c(s')$ if and only if s and s' belong to the same cluster.

Choosing the root which has more children than the other

in Step 2, we can make this procedure more efficient. It can be done by using the new variable $n(r)$ which holds the number of children of a root r . Starting from all $n(s) = 1$, we only need to update the variable as $n(R) := n(r) + n(r')$ in Step 2.

- 1) M. Suzuki: Prog. Theor. Phys. **56** (1976) 1454.
- 2) H. G. Evertz, G. Lana and M. Marcu: Phys. Rev. Lett. **70** (1993) 875.
- 3) N. V. Prokof'ev, B. V. Svistunov and I. S. Tupitsyn: Sov. Phys. JETP **87** (1998) 310.
- 4) O. F. Syljuåsen and A. W. Sandvik: Phys. Rev. E **66** (2002) 046701.
- 5) N. Kawashima and J. E. Gubernatis: Phys. Rev. Lett. **73** (1994) 1295.
- 6) A. W. Sandvik: Phys. Rev. B **59** (1999) 14157.
- 7) B. B. Beard and U.-J. Wiese: Phys. Rev. Lett. **77** (1996) 5130.
- 8) R. Brower, S. Chandrasekharan and U.-W. Wiese: Physica A **261** (1998) 520.
- 9) V. A. Kashurnikov, N. V. Prokofev, B. V. Svistunov and M. Troyer: Phys. Rev. B **59** (1999) 1162.
- 10) S. Chandrasekharan and U.-J. Wiese: Phys. Rev. Lett. **83** (1999) 3116.
- 11) S. Chandrasekharan, J. Cox, J. C. Osborn and U.-J. Wiese: Nucl. Phys. B **673** (2003) 405, and the references cited therein.
- 12) N. Hatano and M. Suzuki: in *Quantum Monte Carlo Methods in Condensed Matter Physics*, ed. M. Suzuki (World Scientific, Singapore, 1994) p. 13.
- 13) H. G. Evertz: Adv. Phys. **52** (2003) 1.
- 14) R. H. Swendsen and J.-S. Wang: Phys. Rev. Lett. **58** (1987) 86.
- 15) N. Kawashima and J. E. Gubernatis: Phys. Rev. E **51** (1995) 1547.
- 16) U. Wolff: Phys. Rev. Lett. **62** (1989) 361.
- 17) D. Kandel and E. Domany: Phys. Rev. B **43** (1991) 8539.
- 18) N. Kawashima and J. E. Gubernatis: J. Stat. Phys. **80** (1995) 169.
- 19) P. W. Kasteleyn and C. M. Fortuin: J. Phys. Soc. Jpn. **26** (1969) Suppl., p. 11.
- 20) C. M. Fortuin and P. W. Kasteleyn: Physica **57** (1972) 536.
- 21) D. C. Handscomb: Proc. Cambridge Philos. Soc. **58** (1962) 594; *ibid.* **60** (1964) 115.
- 22) M. Suzuki, S. Miyashita and A. Kuroda: Prog. Theor. Phys. **58** (1977) 1377.
- 23) M. Marcu: in *Quantum Monte Carlo Methods in Equilibrium and Non-Equilibrium Systems*, ed. M. Suzuki (Springer-Verlag, Berlin, 1987) p. 64.
- 24) A. Wiesler: Phys. Lett. A **89** (1982) 359.
- 25) T. Nakamura and Y. Ito: J. Phys. Soc. Jpn. **72** (2003) 2405.
- 26) H. G. Evertz and M. Marcu: in *Quantum Monte Carlo Methods in Condensed Matter Physics*, ed. M. Suzuki (World Scientific, Singapore, 1994) p. 65.
- 27) M. Aizenman and B. Nachtergaele: Commun. Math. Phys. **164** (1994) 17.
- 28) N. Kawashima: J. Stat. Phys. **82** (1996) 131.
- 29) K. Harada and N. Kawashima: J. Phys. Soc. Jpn. **70** (2001) 13.
- 30) M. S. Makivic and H.-Q. Ding: Phys. Rev. B **43** (1991) 3562.
- 31) A sample program based on the directed loop algorithm (with the solution obtained by the coarse-graining technique) may be found in <http://ccmp18.phys.metro-u.ac.jp/QMC>
- 32) K. Harada and N. Kawashima: Phys. Rev. B **55** (1997) R11949.
- 33) K. Harada and N. Kawashima: J. Phys. Soc. Jpn. **67** (1998) 2768.
- 34) D. R. Nelson and J. M. Kosterlitz: Phys. Rev. Lett. **39** (1977) 1201.
- 35) E. L. Pollock and D. M. Ceperley: Phys. Rev. B **36** (1987) 8343.
- 36) M. Marcu and A. Wiesler: J. Phys. A **18** (1985) 2479.
- 37) N. Hatano: J. Phys. Soc. Jpn. **64** (1995) 1529.
- 38) A. W. Sandvik and J. Kurkijärvi: Phys. Rev. B **43** (1991) 5950.
- 39) A. W. Sandvik: J. Phys. A **25** (1992) 3667.
- 40) K. Harada, M. Troyer and N. Kawashima: J. Phys. Soc. Jpn. **67** (1998) 1130.
- 41) S. Todo and K. Kato: Phys. Rev. Lett. **87** (2001) 047203.
- 42) For physical results on this model, see, for example, A. Schmitt, K. H. Mütter, M. Karbach, Y. Yu and G. Müller: Phys. Rev. B **58** (1998) 5498, and references cited therein; also see K. Nomura and S. Takada: J. Phys. Soc. Jpn. **60** (1991) 389.
- 43) K. Harada and N. Kawashima: Phys. Rev. E **66** (2002) 056705; *ibid.* **67** (2003) 039903(E).
- 44) K. Harada, N. Kawashima and M. Troyer: Phys. Rev. Lett. **90** (2003) 117203.
- 45) We here use the word *worm* in the same way as in Prokof'ev *et al.*³⁾ whereas in our previous paper⁴³⁾ we used the same word to refer to the head or the tail, individually, because only these two points are the essential ingredients of the algorithm. 'The two-ladybug algorithm' might be a more appropriate name of the algorithm.
- 46) F. Alet, S. Wessel and M. Troyer: cond-mat/0308495.
- 47) J. Smakov, K. Harada and N. Kawashima: Phys. Rev. E **68** (2003) 046708.
- 48) J. Holstein and N. Primakoff: Phys. Rev. **58** (1940) 1098.
- 49) N. Hatano and M. Suzuki: Phys. Lett. A **163** (1992) 246.
- 50) H. G. Evertz and W. von der Linden: Phys. Rev. Lett. **86** (2001) 5164.
- 51) B. A. Berg and T. Neuhaus: Phys. Lett. B **267** (1991) 249.
- 52) B. A. Berg and T. Neuhaus: Phys. Rev. Lett. **68** (1992) 9.
- 53) J. Lee: Phys. Rev. Lett. **71** (1993) 211.
- 54) F. Wang and D. P. Landau: Phys. Rev. Lett. **86** (2001) 2050.
- 55) C. Yamaguchi and N. Kawashima: Phys. Rev. E **65** (2002) 056710.
- 56) M. Troyer, S. Wessel and F. Alet: Phys. Rev. Lett. **90** (2003) 120201.
- 57) C. Yamaguchi, N. Kawashima and Y. Okabe: Phys. Rev. E **66** (2002) 036704.
- 58) W. Janke and S. Kappler: Phys. Rev. Lett. **74** (1995) 212.
- 59) P. M. C. de Oliveira, T. J. P. Penna and H. J. Herrmann: Braz. J. Phys. **26** (1998) 677; Eur. Phys. J. B **1** (1998) 205.
- 60) J.-S. Wang, T. K. Tay and R. H. Swendsen: Phys. Rev. Lett. **82** (1999) 476.
- 61) C. Yamaguchi, N. Kawashima and Y. Okabe: unpublished.
- 62) H. Rieger and N. Kawashima: Eur. Phys. J. B **9** (1999) 233.
- 63) A. Dorneich and M. Troyer: Phys. Rev. E **64** (2001) 066701.
- 64) K. Harada and N. Kawashima: unpublished.



Naoki Kawashima was born in Muroran, Hokkaido, Japan in 1964. He got his B.Sc. (1987), M.Sc. (1989) and D.Sc. (1992) degrees from the University of Tokyo. He was a research associate at the University of Tokyo in the period of 1992–1995, during which he also worked as a post doctoral research fellow at Los Alamos National Laboratory in New Mexico, U.S.A. He then became a lecturer (1995–1998) at Toho University. He has been an associate professor at Tokyo Metropolitan University since 1998. He has been working on statistical and condensed-matter physics through classical and quantum spin models. He was awarded the 2002 Ryogo Kubo memorial prize for statistical physics for his contribution to the development of quantum Monte Carlo methods.



Kenji Harada was born in Osaka, Japan in 1970. Studying at Department of Applied Mathematics and Physics, Kyoto University, he received his B.E. from Kyoto University in 1992. Between 1992 and 1998 he studied at Division of Applied Systems Science, Kyoto University, and he received his M.E. and D.E. from Kyoto University in 1994 and 1998, respectively. Appointed research associate at Graduate School of Informatics, Kyoto University in 1998, he works there. His research is focused on the phase transition of various types of quantum models based on quantum Monte Carlo simulations and development of numerical algorithms.