# Self-learning Monte Carlo method with Behler-Parrinello neural networks

Yuki Nagai,[1,2] Masahiko Okumura,[1] and Akinori Tanaka[2,3,4]

[1]*CCSE, Japan Atomic Energy Agency, 178-4-4, Wakashiba, Kashiwa, Chiba 277-0871, Japan*
[2]*Mathematical Science Team, RIKEN Center for Advanced Intelligence Project (AIP), 1-4-1 Nihonbashi, Chuo-ku, Tokyo 103-0027, Japan*
[3]*Department of Mathematics, Faculty of Science and Technology, Keio University, 3-14-1 Hiyoshi, Kouhoku-ku, Yokohama 223-8522, Japan*
[4]*Interdisciplinary Theoretical & Mathematical Sciences Program (iTHEMS) RIKEN 2-1, Hirosawa, Wako, Saitama 351-0198, Japan*

(Received 30 July 2018; revised manuscript received 17 February 2020; accepted 19 February 2020; published 9 March 2020)

We propose a general way to construct an effective Hamiltonian in the self-learning Monte Carlo method (SLMC), which speeds up Monte Carlo simulations by training an effective model to propose uncorrelated configurations in the Markov chain. Its applications are, however, limited. This is because it is not obvious to find the explicit form of the effective Hamiltonians. Particularly, it is difficult to make trainable effective Hamiltonians including many-body interactions. In order to overcome this critical difficulty, we introduce the Behler-Parrinello neural networks (BPNNs) as effective Hamiltonian without any prior knowledge, which is used to construct the potential-energy surfaces in interacting many particle systems for molecular dynamics. We combine SLMC with BPNN by focusing on a divisibility of Hamiltonian and propose how to construct the elementwise configurations. We apply it to quantum impurity models. We observed significant improvement of the acceptance ratio from 0.01 (the effective Hamiltonian with the explicit form) to 0.76 (BPNN). This drastic improvement implies that the BPNN effective Hamiltonian includes many-body interaction, which is omitted in the effective Hamiltonian with the explicit forms. The BPNNs make SLMC more promising.

DOI: 10.1103/PhysRevB.101.115111

## I. INTRODUCTION

Quantum Monte Carlo (QMC) is one of the unbiased numerical methods for studying quantum many-body systems [1–4]. The developments of the continuous-time QMC have had great successes in strongly correlated electron systems [5–8]. In this algorithm, the partition function is expanded in a power series of the perturbation terms. Both the number and position of perturbation terms on the imaginary-time interval change constantly during the simulation. To compute the weight of each configuration, the continuous-time QMC methods require integrating out the degree of freedom of the fermions, which is very time consuming.

The self-learning Monte Carlo (SLMC) method [9–12] was recently introduced as a general method, which speeds up the MC simulation by designing and training a model to propose efficient global updates. There are many kinds of applications, such as classical statistical mechanics models [9,13], classical spin-fermion models [10], determinant QMC [11,14,15], continuous-time QMC [12,16], and hybrid MC in high-energy physics [17]. The SLMC is one of the successes in machine-learning techniques in physics [18–32]. The philosophy behind SLMC is first learn, then earn. In the learning stage, we perform trial simulations to generate a large set of configurations and their weights. These configurations and weights are then used to train an effective model $H_{\rm eff}$, whose Boltzmann weight $e^{-\beta H_{\rm eff}}$ fits the probability distribution of the original problem. Next, in the actual simulation, $H_{\rm eff}$ is used as a guide to propose highly efficient global moves in configuration space.

A good effective model makes simulations with the SLMC more efficient. The efficient effective model is usually invented based on the human understanding of the original system [9–12,14]. However, it is not easy to construct effective models including many-body interactions, since we do not know systematic procedure applicable in arbitrary systems. Only for the Hirsch-Fye QMC method in a quantum impurity model, the convolutional deep neural network (CNN) was proposed to construct the effective Hamiltonian without explicit forms [33].

The machine learning including artificial neural networks has been used for about 20 years in the field of the molecular dynamics (MD) to construct the potential-energy surfaces (PESs) providing interatom forces with accuracy and computational complexity, respectively, comparable to quantum and classical mechanical calculations. In the method, the neural network is trained using a large data set consisting of pairs of an atom configuration with continuous position index and corresponding total energy in some systems given by quantum mechanical calculation (e.g., the density functional theory) [34]. We point out that the neural network PESs can be considered as a general scheme to construct effective Hamiltonians of systems consisting of interacting particles with continuous indices. The wide applicability of this method allows us to apply it to complex problems like the imaginary-time MC calculation of electrons in a solid, which has both discrete and continuous coordinates corresponding to positions on a lattice and imaginary time, respectively.

In this paper, we propose a method to construct the effective Hamiltonians with Behler-Parrinello neural networks (BPNN) [34], which is one of most succeeded methods in the field of the molecular dynamics with machine learning. We regard the configuration and effective Hamiltonian in

SLMC as the positions of the atoms and the PESs in the MD, respectively. We use the recently proposed method [35] in the MD field to map continuous coordinates (atom positions) onto discrete variables (inputs of BPNN), whose advantage is availability of systematic improvement of the mapping accuracy. As a concrete example, we demonstrate self-learning continuous-time interaction-expansion (CTINT) QMC with BPNNs on quantum impurity models. We implement the simplest neural networks and test their performances. We also develop the fast updates, which is applicable even with deep neural networks to reduce the computational cost significantly in the SLMC simulation.

The paper is organized as follows. The self-learning Monte Carlo method is introduced in Sec. II, the effective model in the SLMC is discussed in Sec. III, the SLMC and BPNN is combined in Sec. IV, an application of the SLMC to quantum impurity models is demonstrated in Sec. V, the discussion is given in Sec. VI, and the conclusion is given in Sec. VII. In Appendix A, we briefly describe the Markov chain Monte Carlo method. The machine-learning technique in molecular dynamics and Behler-Parrinello neural networks are introduced in Appendix B. In Appendix C, the technical details of the batch-atom normalization used in the self-learning CTINT.

## II. SELF-LEARNING MONTE CARLO METHOD

### A. Metropolis-Hastings algorithm

The Markov chain Monte Carlo method (MCMC) is a powerful method for an integration in high-dimensional space. In physics, the partition functions and physical exception values are often calculated by the MCMC. Some difficulties of the MCMC method in physics are described in Appendix A.

In the Monte Carlo method, we have to generate a configuration $\mathcal{C}$ with a probability distribution $w(\mathcal{C})$. By constructing a Markov chain that has the desired distribution as its equilibrium distribution, we can obtain a sample of the desired distribution by observing the chain after a number of steps. To construct the Markov chain, we introduce the condition of the detailed balance expressed as

$$w(\mathcal{C})P(\mathcal{C}'|\mathcal{C}) = w(\mathcal{C}')P(\mathcal{C}|\mathcal{C}'). \tag{1}$$

Here, $P(\mathcal{C}'|\mathcal{C})$ is the probability of transitioning from another configuration $\mathcal{C}$ to a configuration $\mathcal{C}'$. The Metropolis-Hastings approach is to separate the transition in two substeps:

$$P(\mathcal{C}'|\mathcal{C}) = g(\mathcal{C}'|\mathcal{C})A(\mathcal{C}',\mathcal{C}), \tag{2}$$

where the proposal distribution $g(\mathcal{C}'|\mathcal{C})$ is the conditional probability of proposing a configuration $\mathcal{C}'$ when a configuration $\mathcal{C}$ is given, and the acceptance ratio $A(\mathcal{C}',\mathcal{C})$ is the probability to accept the proposed configuration $\mathcal{C}'$. By inserting the above expression into Eq. (1), we have

$$\frac{A(\mathcal{C}',\mathcal{C})}{A(\mathcal{C},\mathcal{C}')} = \frac{w(\mathcal{C}')}{w(\mathcal{C})}\frac{g(\mathcal{C}|\mathcal{C}')}{g(\mathcal{C}'|\mathcal{C})}. \tag{3}$$

The Markov chain that has the desired distribution $w(\mathcal{C})$ is obtained when the acceptance ratio is given as [36]

$$A(\mathcal{C}',\mathcal{C}) = \min\left(1, \frac{w(\mathcal{C}')}{w(\mathcal{C})}\frac{g(\mathcal{C}|\mathcal{C}')}{g(\mathcal{C}'|\mathcal{C})}\right). \tag{4}$$

Then, we can generate the Markov chain expressed as

$$\mathcal{C}_1 \to \cdots \to \mathcal{C}_i \to \cdots . \tag{5}$$

One can design various kinds of the Monte Carlo method based on Eq. (4) (see Appendix A). We need an update method from $\mathcal{C}$ to $\mathcal{C}'$ with the high acceptance ratio for good efficiency of the MCMC. The most simple update method is so-called local update, where the configuration is updated locally. One randomly chooses a single site in the current configuration and proposes a new configuration by changing the variable on this site. However, the local update suffers heavily from a critical slowing down when the system is close to phase transitions. In such cases, the autocorrelation time within the Markov chain $\tau$ becomes very large. To overcome the increase of the autocorrelation time for the local update, various kinds of global update method have been developed [37–41]. In all these global update methods, variables on a large number of sites are simultaneously changed in a single Monte Carlo update. For a given generic model, it is hard to design an efficient global update method.

We note that the hybrid Monte Carlo (HMC) method is known as the one of the good global update methods, which is widely used in the lattice quantum chromodynamics (QCD) (see, Appendix A 2 b). Recently, the HMC method is revisited in the condensed matter physics to treat strongly correlated electron systems [42], since the HMC method is general to obtain uncorrelated configurations. Although the HMC method might be suitable for reducing the autocorrelation time, its computational cost is not small.

### B. Basic concept of the SLMC

In MCMC, we can design the proposal probability $g(\mathcal{C}|\mathcal{C}')$. If the ratio of the probability $g(\mathcal{C}|\mathcal{C}')/g(\mathcal{C}'|\mathcal{C}) = w(\mathcal{C})/w(\mathcal{C}')$, the new configuration $\mathcal{C}'$ is always accepted (i.e., $A(\mathcal{C}',\mathcal{C}) = 1$). To design the proposal probability, we use the another Markov chain with the probability $w_{\rm prop}(\mathcal{C})$. We consider that the configuration $\mathcal{C}'$ is obtained by the random walk from $\mathcal{C}$ on this proposal Markov chain. Its detailed balance condition is given as

$$w_{\rm prop}(\mathcal{C})P_{\rm prop}(\mathcal{C}'|\mathcal{C}) = w_{\rm prop}(\mathcal{C}')P_{\rm prop}(\mathcal{C}|\mathcal{C}'). \tag{6}$$

This proposal probability $P_{\rm prop}(\mathcal{C}'|\mathcal{C})$ can be regarded as the conditional probability $g(\mathcal{C}'|\mathcal{C})$ on the original Markov chain, which proposes a configuration $\mathcal{C}'$ from given $\mathcal{C}$. Thus, with the use of the relation:

$$\frac{g(\mathcal{C}|\mathcal{C}')}{g(\mathcal{C}'|\mathcal{C})} = \frac{w_{\rm prop}(\mathcal{C})}{w_{\rm prop}(\mathcal{C}')}, \tag{7}$$

the acceptance ratio in the SLMC is given as

$$A(\mathcal{C}',\mathcal{C}) = \min\left(1, \frac{w(\mathcal{C}')}{w(\mathcal{C})}\frac{w_{\rm prop}(\mathcal{C})}{w_{\rm prop}(\mathcal{C}')}\right). \tag{8}$$

If we can design the proposal Markov chain whose probability is equal to that of the original Markov chain $w_{\rm prop}(\mathcal{C}) = w(\mathcal{C})$, the proposed configuration $\mathcal{C}'$ is always accepted. The average acceptance rate $\langle A\rangle$ can be estimated by $\langle A\rangle = \exp[-\sqrt{\rm MSE}]$ with the mean squared error $\mathrm{MSE} = (1/n)\sum_i(\ln w_{\rm prop}(\mathcal{C}_i) - \ln w(\mathcal{C}_i))^2$ [33]. Here, $n$ is the number of the measurements in Monte Carlo simulations.

TABLE I. Difference between three update methods.

| | Hand-designed | HMC | SLMC |
|---|---|---|---|
| Proposed method | by hand | MD | Markov chain |
| $g(\mathcal{C}\|\mathcal{C}')/g(\mathcal{C}'\|\mathcal{C})$ | usually 1 | 1 | $w_{\rm eff}(\mathcal{C})/w_{\rm eff}(\mathcal{C}')$ |

The difference between several update methods is shown in Table I. HMC and SLMC are the global updates where the configurations are changed globally. In the SLMC, the configuration is proposed by the proposal Markov chain. We have to find a good proposal Markov chain whose probability $w_{\rm prop}(\mathcal{C})$ is similar to original one $w(\mathcal{C})$.

To design the proposal Markov chain, we have to construct an effective model. We introduce an effective Hamiltonian $H_{\rm eff}(\mathcal{C})$:

$$w_{\rm prop}(\mathcal{C}) = w^{\rm eff}(\mathcal{C}) = \exp\left[-\beta H_{\rm eff}(\mathcal{C})\right]. \tag{9}$$

This model $H_{\rm eff}(\mathcal{C})$ can be constructed by a supervised machine-learning technique [9]. In the learning stage, trial simulations are performed to generate a large set of configurations and their weights. These data are then used to train an effective model $H_{\rm eff}(\mathcal{C})$, whose weight fits the probability distribution of the original problem $w(\mathcal{C})$. Next, in the actual simulation, $H_{\rm eff}(\mathcal{C})$ is used as a guide to propose highly efficient global moves in configuration space. It is important to obtain good effective models in the SLMC simulations. In the previous study [12], we have successfully obtained the form of the effective Hamiltonian with two-body interactions in the continuous-time auxiliary-field QMC (CTAUX) for the Anderson impurity model. In the CTINT simulations, Huang *et al.* have produced the classical Hamiltonian with two- and three-body interactions to reproduce the weights [16]. However, it seems hard to construct the effective Hamiltonian in other systems or other methods.

### C. Calculation cost of the SLMC: Reduction of the autocorrelation time

A computational cost of MCMC simulations is usually estimated by autocorrelation time. However, autocorrelation time depends on a kind of the correlation function, which we focus on. As shown in Fig. 1, the SLMC can bypass links of the computational costly weight estimations. Thus, in SLMC, autocorrelation time for all correlation function is shorter than that in the original simulation. We should note that, although the elapsed time comparison in actual calculations might be regarded as the other estimation of the computational cost, this estimation can not be a fair estimation in the case of the SLMC, since the elapsed time of the original MC and SLMC both depends on the computer architecture and coding techniques of optimization.

We propose the general estimation of the computational cost of the SLMC as follows. We define $u_{\rm original}$ as the computational complexity for calculating the weight of the original model $w(\mathcal{C})$. We consider the autocorrelation time of the original MC simulation $\tau_{\rm original}$. The total calculation cost of the original simulation is written as

$$U_{\rm original} \propto \tau_{\rm original} u_{\rm original}. \tag{10}$$

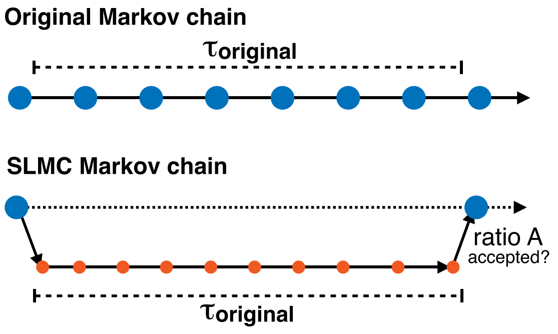

FIG. 1. Schematic figure for the Markov chains of the original Monte Carlo and SLMC. The SLMC can bypass links of the computational costly weight estimations. Large blue (small red) circles denote the computational complexity $u_{\rm original}$ ($u_{\rm SLMC}$) for the calculating the weight of the original model $w(\mathcal{C})$ [$w(\mathcal{C})$]. $\tau_{\rm original}$ is the autocorrelation time of the original MC simulation. Since a complexity of a matrix determinant for Fermion systems is quite large, $u_{\rm SLMC} < u_{\rm original}$ is satisfied even if the neural networks are adopted. If the average acceptance ratio $\langle A\rangle$ is not too small, the autocorrelation time calculated as the number of the large blue circles is much shorter than that of the original simulation.

On the other hand, the total calculation cost of the SLMC is written as

$$U_{\rm SLMC} \propto \frac{1}{\langle A\rangle}\left(N_{\rm steps}^{\rm prop} u_{\rm SLMC} + u_{\rm original}\right), \tag{11}$$

$$= \left(\frac{N_{\rm steps}^{\rm prop}}{\langle A\rangle}\frac{u_{\rm SLMC}}{u_{\rm original}} + \frac{1}{\langle A\rangle}\right) u_{\rm original} \tag{12}$$

Here, $\langle A\rangle$ is the average acceptance ratio and $N_{\rm steps}^{\rm prop}$ is the length of the proposal Markov chain. We define $u_{\rm SLMC}$ as the computational complexity for calculating the weight of the original model $w(\mathcal{C})$. The ratio of the computational cost is given as

$$\frac{U_{\rm SLMC}}{U_{\rm original}} \propto \frac{1}{\langle A\rangle}\left(\frac{N_{\rm steps}^{\rm prop}}{\tau_{\rm original}}\frac{u_{\rm SLMC}}{u_{\rm original}} + \frac{1}{\tau_{\rm original}}\right). \tag{13}$$

We note that $N_{\rm steps}^{\rm prop} = \tau_{\rm original}$ if the proposal Markov chain uses the same update method of the original simulation. If the effective Hamiltonian gives a perfect weight (i.e., $w^{\rm eff}(\mathcal{C}) = w(\mathcal{C})$), the average acceptance ratio is always one: $\langle A\rangle = 1$. In this case, if the length of the proposal Markov chain $N_{\rm steps}^{\rm prop}$ is longer than $\tau_{\rm original}$, there is no correlation between the previous and proposed configurations $\mathcal{C}$ and $\mathcal{C}'$. The calculation cost depends on the ratio of $u_{\rm SLMC}/u_{\rm original}$. Thus, the SLMC is faster than the original simulation when the computational cost for effective weight $w^{\rm eff}(\mathcal{C})$ is smaller than that for the original weight $w(\mathcal{C})$ and the average acceptance ratio $\langle A\rangle$ is not too small.

With the use of Eq. (12), the number of the weight calculation $u_{\rm original}$ can be regarded as the definition of the autocorrelation time of the SLMC. Thus, the autocorrelation time $\tau_{\rm SLMC}$ is expressed as

$$\tau_{\rm SLMC} \sim \frac{1}{\langle A\rangle}. \tag{14}$$

TABLE II. Similarity between *ab initio* MD and QMC.

| | MD | QMC |
|---|---|---|
| Input | atom-positions | configurations |
| Time-consuming | energy | MC weight |
| Eff. Hamiltonian (two-body) | BP | SLMC |
| Eff. Hamiltonian (many-body) | BP | — |

## III. EFFECTIVE MODELS

The critical problem in SLMC is construction of an effective Hamiltonian $\ln w^{\rm eff}(\mathcal{C})$, which mimics the original one $\ln w(\mathcal{C})$. In this section, we introduce the Behler-Parrinello neural networks as a general method to construct the effective Hamiltonian.

### A. Effective Hamiltonians

We discuss the simplest effective Hamiltonian is one with only two-body interaction terms. It can be represented as

$$H_{\rm eff}(\mathcal{C}) = \sum_{ij}^{M} L(\mathcal{C}_i, \mathcal{C}_j), \tag{15}$$

where $\mathcal{C}_i$ is the $i$th element of the configuration $\mathcal{C} = (\mathcal{C}_1, \dots, \mathcal{C}_M)$, and $M$ is the number of the elements. The element is usually coordinate of the atoms in the imaginary time and/or real space. In the previous work, we introduced an explicit form of $L(\mathcal{C}_i, \mathcal{C}_j)$, which has been known. But, in general, it is not easy to find explicit forms of effective Hamiltonians even including only two-body interactions for arbitrary Hamiltonians.

### B. Beyond the two-body interaction: Behler-Parrinello neural networks

We introduce the BPNNs as a general method to construct effective Hamiltonians including many-body interaction in the SLMC. Before showing detailed formulations, we give brief discussions on the BPNNs and its applicability for SLMC.

In the research field of MD, the machine-learning techniques have been used for about 20 years to evaluate effective interatom or intermolecule forces. The BPNNs method is one of most successful methods using them. See Appendix B for details of the BPNNs-based method. Originally, the BPNNs are used to obtain forces among atoms without time-consuming calculations of *ab initio* MD. In the method, the forces are obtained as derivatives of a PES with respect to atom positions, which is represented by artificial neural networks trained using data sets obtained by *ab initio* MD. It should be mentioned that the method provides the PES including not only two-body but many-body interactions, which are contained in *ab initio* MD.

In order to clarify applicability of the BPNNs for SLMC, let us discuss the similarity between *ab initio* MD and QMC. Correspondences are shown in Table II. The most time-consuming part in the whole calculation is evaluation of energy, which depends on the coordinates. The BP method avoids the bottleneck of the calculations, i.e., DFT calculations, using the PES represented by trained artificial neural networks. On the other hand, in MC simulations, the Monte Carlo weights are critical quantity. Calculation of them is most time-consuming in the MC simulations. SLMC is one of promising methods to shorten the calculation time on the part by replacing the Hamiltonian with the effective one. But, as mentioned above, there was no systematic way to find the effective Hamiltonian especially including many-body interactions (Table II). The BP method can give the effective Hamiltonian depending on the configuration in MC, which is completely equivalent with the PES depending on the atom coordinates.

## IV. COMBINE SLMC WITH BPNN

### A. Divisibility of Hamiltonian

The discussion in the previous section clarified general correspondence between these two methods. In this section, we show deeper correspondence between them in fermion systems.

In the BP method, the total energy in the system $E_{\rm tot}^{(\rm S)}(\mathcal{R}^{(\rm S)})$ is assumed to be divided into partial energies associated with each atom in the system, which are determined by environment around each atom, i.e.,

$$E_{\rm tot}^{(\rm S)}(\mathcal{R}^{(\rm S)}) = \sum_{i=1}^{N^{(\rm s)}} E_{\rm part}^{(\rm S)}\big(\Delta_i^{(\rm S)}\big), \tag{16}$$

where

$$\Delta_i^{(\rm S)} = \{\boldsymbol{r}_{ij} \mid j = 1, \dots, i-1, i+1, \dots, N^{(\rm s)}\} \tag{17}$$

is a set of relative coordinates with respect to the $i$th atom ($\boldsymbol{r}_{ij} = \boldsymbol{r}_j - \boldsymbol{r}_i$). This divisibility of the Hamiltonian is an assumption, but the artificial neural networks are trained to find a set of the partial energies to reproduce the original Hamilton.

On the other hand, in the MC calculations of fermion systems, the primary quantity is the partition function $Z$ expressed as the determinant of the matrix:

$$Z = \sum_{\mathcal{C}} \det M(\mathcal{C}) = \sum_{\mathcal{C}} w(\mathcal{C}). \tag{18}$$

The dimension of the matrix depends on variation of the MC methods. For example, in continuous-time QMC (e.g., CT-AUX and CT-INT) for an impurity model, the dimension of the matrix $M$ is the number of the vertices on the imaginary-time axis. Here, we mention that the logarithm of the weight $\ln w(\mathcal{C})$ can be divided into $N$ parts as

$$\ln w(\mathcal{C}) = \ln \det M(\mathcal{C}), \tag{19}$$

$$= \sum_{j=1}^{N} [\ln M(\mathcal{C})]_{jj}. \tag{20}$$

This equation leads partitioning of an arbitrary effective Hamiltonian into the partial Hamiltonians, i.e.,

$$H_{\rm eff}(\mathcal{C}) = \sum_{j=1}^{N} h_{\rm eff}^{\alpha_j}(N, \boldsymbol{c}^j). \tag{21}$$

We emphasize that this equation is not an assumption but an exact expression.

There is no rigorous correspondence between them although Eqs. (16) and (20) have similar expressions. In this paper, we show, however, the BP method give appropriate partitioning of the effective Hamiltonian.

### B. Effective Hamiltonian in SLMC

In actual SLMC simulations, we adopt the effective Hamiltonian defined as

$$H_{\rm eff}(\mathcal{C}) = \frac{1}{N}\sum_{j=1}^{N} h_{\rm eff}^{\alpha_j}(\boldsymbol{c}^j) + f(N). \tag{22}$$

Here, $f(N)$ is a polynomial function $f(N) = \sum_{k=0}^{n_{\rm max}} f_k N^k$. We assume $h_{\rm eff}^{\alpha_j}(N, \boldsymbol{c}^j) = (1/N)h_{\rm eff}^{\alpha_j}(\boldsymbol{c}^j)$. This factor $1/N$ is introduced as the normalization, which is appropriate for the CTAUX in the previous paper. The $h_{\rm eff}^{\alpha_j}(\boldsymbol{c}^j)$ is constructed by neural networks. Note that the nonlinear function $h_{\rm eff}^{\alpha_j}(\boldsymbol{c}^j)$ does not depend on $j$ and only depends on $\alpha_j$, since we assume that the same kind of atoms feel the same interactions.

### C. Construction of the element-wise configurations

We have to generate elementwise configurations $\boldsymbol{c}^j$ from $\mathcal{C}$. Usually, in QMC simulations, the configuration $\mathcal{C}$ is a set of positions of atoms (e.g., spins or vertices) on real and/or imaginary-time axes. We can make the elementwise configurations $\boldsymbol{c}^j$, which consist of the distances between the atom $j$ and other atoms. However, it is not a good representation of the configuration for NNs, since the number of elements of the input vector $\boldsymbol{c}^j$ varies depending on the number of atoms although the number of inputs of NNs must be fixed. Actually, in continuous-time QMC simulations, the number of atoms changes during simulations.

One of the methods to construct the elementwise configurations on continuous axis is the Chebyshev polynomial expansion method in the field of machine-learning MD simulations [35]. The elementwise configuration is expressed by coefficients of the basis functions as follows. For simplicity, we consider a configuration that has vertices on the imaginary-time axis such as a quantum impurity model. We map the configuration $\mathcal{C}$ onto the set of the elementwise configurations $\mathcal{C}^{\rm ele}(\tau_i - \tau_j)$ around a vertex $j$: $\mathcal{C} \rightarrow \{\mathcal{C}^{\rm ele}(\tau_1 - \tau_j), \mathcal{C}^{\rm ele}(\tau_2 - \tau_j), \dots, \mathcal{C}^{\rm ele}(\tau_N - \tau_j)\}$. The elementwise configuration is expressed by the basis functions. We introduce the density distribution functions defined as

$$\rho(\tau, \mathcal{C}^{\rm ele}(\tau_i - \tau_j)) = \sum_{i=1}^{N} \delta(\tau - \tau_{ij}), \tag{23}$$

where $\tau_{ij} = 2|\tau_i - \tau_j|/\beta - 1$ is the distance between atom $i$ and atom $j$. This distribution is expanded by the Chebyshev polynomial functions [35]:

$$\rho(\tau, \mathcal{C}^{\rm ele}(\tau_i - \tau_j)) = \sum_m c_m^j \phi_m(\tau), \tag{24}$$

where

$$c_m^j = \sum_{i=1}^{N} \phi_m(\tau_{ij}). \tag{25}$$

Here, $\phi_m(x)$ is the Chebyshev polynomial function $\phi_m(x) = \cos{(m \arccos(x))}$. Thus, we map the elementwise configuration $\mathcal{C}^{\rm ele}(\tau_i - \tau_j)$ onto the set of $m_{\rm cut}$ coefficients of the Chebyshev polynomials $\mathcal{C}^{\rm ele}(\tau_i - \tau_j) \rightarrow \{c_0^j, \dots, c_{m_{\rm cut}-1}^j\}$, as shown in Fig. 2. The elementwise configuration $\boldsymbol{c}^j$ is expressed as $\boldsymbol{c}^j \equiv \{c_0^j, \dots, c_{m_{\rm cut}-1}^j\}$, which does not depend on the number of atoms.

To introduce a difference of species, we can add another distribution function. For example, if the vertex has a spin index $s_j$ and there is spin-reversal symmetry (i.e., the weight

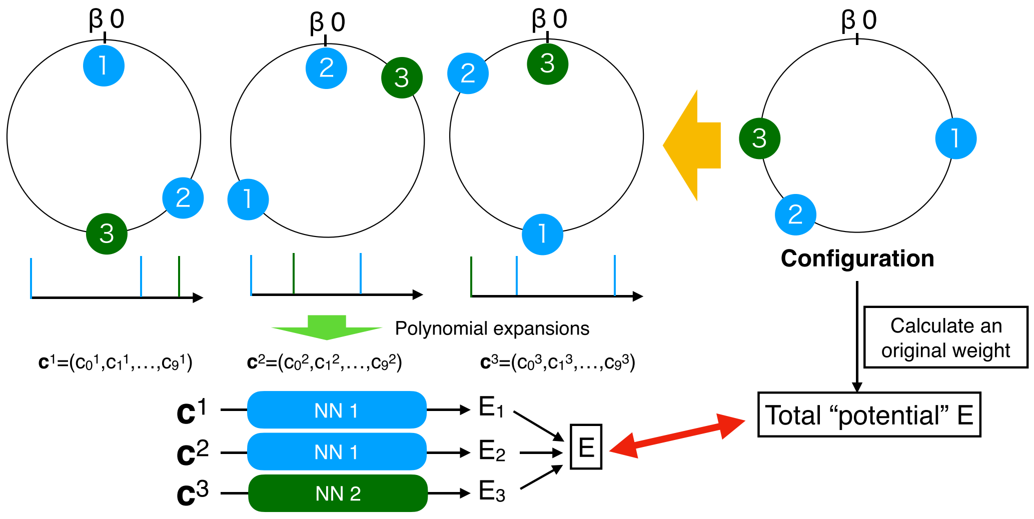

FIG. 2. Schematic figures of Behler-Parrinello neural networks for the self-learning continuous-time interaction-expansion quantum Monte Carlo method. The configuration with $N$ vertices on the imaginary-time axis $\mathcal{C} = \{\tau_1, \tau_2, \dots, \tau_N\}$ is mapped to the set of the elementwise configurations $\mathcal{C}^{\rm ele}(\tau_i - \tau_j)$ around a vertex $j$: $\mathcal{C} \rightarrow \{\mathcal{C}^{\rm ele}(\tau_1 - \tau_j), \mathcal{C}^{\rm ele}(\tau_2 - \tau_j), \dots, \mathcal{C}^{\rm ele}(\tau_N - \tau_j)\}$. The elementwise configuration is expressed as a distribution function with $\delta$ functions. The total potential $E$ of the original calculation is expressed as $E = \log w(\mathcal{C})/(-\beta)$. $E$ obtained by the neural networks is expressed as $E = \log w^{\rm eff}(\mathcal{C})/(-\beta) = H_{\rm eff}(\mathcal{C})$. The partial energy $E_i$ is defined as $E_i = h_{\rm eff}^{\alpha_i}(\boldsymbol{c}^j)/(-\beta)$. The color of the circle denotes a kind of a vertex. Same neural networks are used if the kind of a vertex is same.

is not changed by flipping all spins), it is better to add spin-density distribution functions defined as

$$\rho_s(\tau, \mathcal{C}^{\rm ele}(\tau_i-\tau_j)) = \sum_{i=1}^{N} s_i s_j \delta(\tau-\tau_{ij}), \tag{26}$$

With the use of the Chebyshev polynomial expansions, we have

$$\rho_s(\tau, \mathcal{C}^{\rm ele}(\tau_i-\tau_j)) = \sum_m d_m^j \phi_m(\tau), \tag{27}$$

where

$$d_m^j = \sum_{i=1}^{N} s_i s_j \phi_m(\tau_{ij}). \tag{28}$$

Then, the element-wise configuration $\boldsymbol{c}^j$ is $\boldsymbol{c}^j \equiv \{c_0^j, \dots, c_{m_{\rm cut}-1}^j, d_0^j, \dots, d_{m_{\rm cut}-1}^j\}$.

### D. Relation between the SLMC with BPNN and previous effective model

In the previous paper about the CTAUX QMC, the configuration $\mathcal{C}$ consists of vertices with a spin index on the imaginary-time continuous axis [12]. We show that the effective Hamiltonian in this previous paper can be regarded as Eq. (22) with linear functions $h_{\rm eff}^{\alpha_j}(\boldsymbol{c}^j)$ without hidden layers. If the function $h_{\rm eff}^{\alpha_j}(\boldsymbol{c}^j)$ is linear, the effective Hamiltonian is expressed as

$$H_{\rm eff}(\mathcal{C}) = \frac{1}{N}\sum_{j=1}^{N} W^T \boldsymbol{c}^j + b + f(N). \tag{29}$$

Here, $W$ is an $M$-dimensional vector and $M$ is the number of the input elements. With the use of the elementwise configuration $\boldsymbol{c}^j$ is $\boldsymbol{c}^j \equiv \{c_0^j, \dots, c_{m_{\rm cut}-1}^j, d_0^j, \dots, d_{m_{\rm cut}-1}^j\}$, this effective Hamiltonian is rewritten as

$$H_{\rm eff}(\mathcal{C}) = \frac{1}{N}\sum_{j=1}^{N}\sum_{m=1}^{m_{\rm cut}} W_l^c c_{l-1}^j + \frac{1}{N}\sum_{j=1}^{N}\sum_{m=1}^{m_{\rm cut}} W_l^d d_{l-1}^j + f(N). \tag{30}$$

Here, the coefficient $b$ is included in $f(N)$. With the use of Eqs. (25) and (28), we obtain

$$H_{\rm eff}(\mathcal{C}) = \frac{1}{N}\sum_{i,j=1}^{N} L(\tau_i-\tau_j) + \frac{1}{N}\sum_{i,j=1}^{N} J(\tau_i-\tau_j) s_i s_j + f(N), \tag{31}$$

where

$$L(\tau_i-\tau_j) \equiv \sum_{m=1}^{m_{\rm cut}} W_l^c \phi_m(\tau_{ij}), \tag{32}$$

$$J(\tau_i-\tau_j) \equiv \sum_{m=1}^{m_{\rm cut}} W_l^d \phi_m(\tau_{ij}). \tag{33}$$

This effective Hamiltonian is completely equivalent to that used in the previous paper [12].

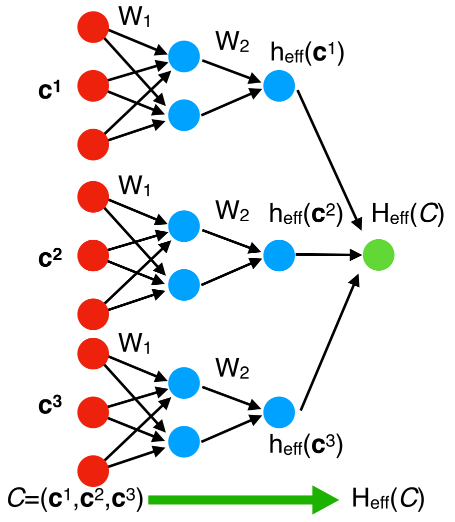

FIG. 3. Schematic figures of input and output data in Behler-Parrinello neural networks. The weight is expressed as a sum of the elementwise effective Hamiltonians. If we neglect hidden layers, the effective Hamiltonian is constructed by linear combinations.

### E. Many-body interactions with neural networks

The effective Hamiltonian with linear functions $h_{\rm eff}^{\alpha_j}(\boldsymbol{c}^j)$ includes only two-body interactions as shown in Eq. (31). To include the many-body interactions, we can add hidden layers. For example, in the case of the neural networks with one hidden layer with $N_u$ units, the effective elementwise Hamiltonian is expressed as

$$h_{\rm eff}^{\alpha_j}(\boldsymbol{c}^j) = \hat{W}_2^{\alpha_j} F\left(\hat{W}_1^{\alpha_j}\boldsymbol{c}^j + \boldsymbol{b}_1^{\alpha_j}\right) + b_2^{\alpha_j}, \tag{34}$$

with the activation function $[F(\boldsymbol{x})]_i = F([\boldsymbol{x}]_i)$. Here, $\hat{W}_1^{\alpha_j}$ is a $N_u \times M$ matrix, $\hat{W}_2^{\alpha_j}$ is a $1\times N_u$ matrix, $\boldsymbol{b}_1^{\alpha_j}$ is a $N_u$-dimensional bias vector, and $b_2^{\alpha_j}$ is a bias. The activation function $F(x)$ makes the effective Hamiltonian nonlinear, which can represent many-body interactions. The schematic figure of the effective Hamiltonian is shown in Fig. 3. We use a common weight matrix $W$ to same atomic species.

## V. DEMONSTRATIONS ON QUANTUM IMPURITY MODELS

### A. Continuous-time QMC for fermions

In continuous-time QMC simulations, by splitting the Hamiltonian into nonperturbative and perturbative parts $H = H_0 + H_1$, the partition function is expanded as

$$Z = {\rm Tr}\left[e^{-\beta H_0} T_\tau e^{-\int_0^\beta H_1(\tau)d\tau}\right], \tag{35}$$

$$= \sum_{N=0}^{\infty} \frac{(-1)^N}{N!}\int_0^\beta d\tau_1 \cdots \int_0^\beta d\tau_N \times {\rm Tr}[T_\tau e^{-\beta H_0} H_1(\tau_N) H_1(\tau_{N-1}) \cdots H_1(\tau_1)], \tag{36}$$

$$= \sum_{\mathcal{C}} W(\mathcal{C}), \tag{37}$$

where $T_\tau$ is the imaginary-time-ordering operator. Here, the configuration $\mathcal{C}$ has $N$ vertices on the imaginary-time axis [7]. The number of vertices $N$ changes during simulation.

### B. Continuous-time interaction expansion quantum Monte Carlo in impurity models

We demonstrate self-learning CTINT with BPNNs of the impurity model. The CTINT method is a good example to demonstrate the SLMC with BPNN, since Huang *et al.* claimed that the three-body interactions are necessary to construct good effective model in CTINT method [16].

We consider the Hamiltonian of the single impurity Anderson model, which is written as the combination of the free fermion part and the interaction part [7,12],

$$H = -\mu \sum_\sigma n_\sigma + \sum_{\sigma,p} (V c_\sigma^\dagger a_{p,\sigma} + \text{H.c.}) + \sum_{\sigma,p} \epsilon_p a_{p,\sigma}^\dagger a_{p,\sigma} + U n_\uparrow n_\downarrow, \quad (38)$$

where $\sigma = \uparrow, \downarrow$, $c_\sigma^\dagger$, and $a_{p,\sigma}^\dagger$ are the fermion creation operators for an impurity electron with spin $\sigma$, and that for a bath electron with spin $\sigma$ and momentum $p$, respectively. $n_\sigma$ is the impurity electron number operator. We consider a bath with a semicircular density of states $\rho_0(\epsilon) = [2/(\pi D)\sqrt{1-(\epsilon/D)^2}]$ and set the half-bandwidth $D = 1$ as the energy unit. In the CTINT algorithm, we rewrite the interaction part expressed as

$$H_1 = \frac{U}{2} \sum_{s=\pm 1} \left(n_\uparrow - \frac{\rho}{2} - s\delta\right)\left(n_\downarrow - \frac{\rho}{2} + s\delta\right). \quad (39)$$

Here, we introduce additional Ising variable $s$ and parameter $\delta$, and $\rho$ corresponds to the average electron density [43–45]. We consider the half-filling ($\rho = 1$). The nonperturbative part is $H_0 = (-\mu + \frac{U}{2}\rho)\sum_\sigma n_\sigma + \sum_{\sigma,p}(V c_\sigma^\dagger a_{p,\sigma} + \text{H.c.}) + \sum_{\sigma,p} \epsilon_p a_{p,\sigma}^\dagger a_{p,\sigma}$. The partition function is expanded as

$$\frac{Z}{Z_0} = \sum_{\mathcal{C}} W(\mathcal{C}) = \sum_{\mathcal{C}} \left(\frac{-U}{2}\right)^N \frac{1}{N!} \prod_\sigma \det M_\sigma(\mathcal{C}). \quad (40)$$

Here, the $N \times N$ matrix $M_\sigma(\mathcal{C})$ is defined by $[M_\sigma(\mathcal{C})]_{ij} \equiv g_0(\tau_i - \tau_j) - \alpha_\sigma(s_n)\delta_{ij}$ with $\alpha_\sigma(s) \equiv \rho/2 + \sigma s\delta$. $g_0(\tau_i - \tau_j)$ is the free fermion Green's function at the impurity site. We implement the CTINT with the Julia language 0.6.2. We train the neural networks with one hidden layer as shown in Fig. 3 with 50000 training data, which is done by the TENSORFLOW 1.4, one of the deep learning frameworks, in PYTHON 3.6.5. The sigmoid function $F(x) = 1/[1 + \exp(-x)]$ is used as the activation function. We develop the batch-atom normalization, variant of the batch normalization [46], which is one of the modern techniques accelerating training procedures for neural network by normalizing along batch index. In the batch-atom normalization, we normalize both batch and atomic index $j$ (see Appendix A). The input vector is a $M = 2m_{\text{cut}}$ dimensional vector: $\boldsymbol{c}^j = (c_0^j, \ldots, c_{m_{\text{cut}}-1}^j, d_0^j, \ldots, d_{m_{\text{cut}}-1})^T$. The total number of parameters in neural networks with one hidden layer with $N_u$ units is $MN_u + N_u + 4N_u + N_u + n_{\max} + 1 = MN_u + 6N_u + n_{\max} + 1$. We usually set the Chebyshev expansion cutoff $m_{\text{cut}} = 10$, which determines the resolution of the input vector on the imaginary-time axis.

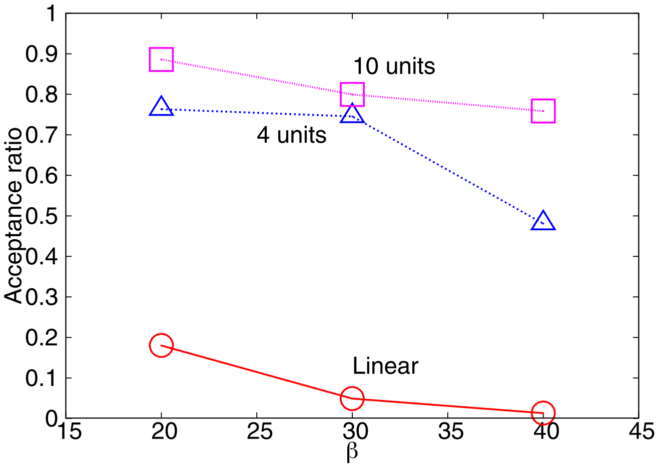

FIG. 4. Inverse-temperature dependence of the acceptance ratio in the self-learning continuous-time interaction expansion quantum Monte Carlo simulation in the Anderson impurity model. We consider $U = 3D$, $\delta = 0.5$, $V = 1D$, and 500 SLMC steps.

Figure 4 shows the inverse-temperature dependence of the acceptance ratio of the SLMC. We consider $U = 3D$, $\delta = 0.5$, $V = 1D$, $m_{\text{cut}} = 10$, and $n_{\max} = 3$. We set the number of the SLMC steps $n$ is $n = 500$. We also consider the linear SLMC, which is equivalent to the previous effective Hamiltonian [12]. In the case with $\beta = 40$, the acceptance ratio with 10 units ($N_u = 10$) is around 0.8, while that of the linear SLMC is less than 0.02. The results indicates that the BPNNs can systematically improve the effective Hamiltonian with increasing the number of units.

To demonstrate the speedup of the SLMC, we compute the autocorrelation function of the auxiliary spin polarization defined by $m \equiv (1/N)\sum_{i=1}^N s_i$. Figure 5 shows the autocorrelation time of the original CT-INT method and the SLMC with the BPNN with 10 units ($N_u = 10$), defined in terms of the number of local updates. We consider $\beta = 40$, $U =$

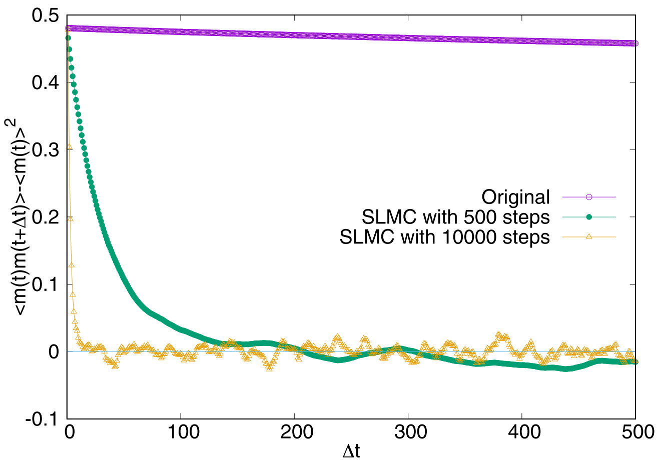

FIG. 5. Autocorrelation function of the auxiliary-spin magnetization for a quantum impurity model with $\beta = 40$, $U = 3D$, $\delta = 0.5$, and $V = 1D$. Unit time is defined in the main text.

$3D$, $\delta = 0.5$, $V = 1D$, $m_{\text{cut}} = 10$, and $n_{\max} = 3$. We set the numbers for the SLMC steps $n$ are $n = 500$ and $n = 10000$, whose acceptance ratios are 76% and 63%, respectively. It is clear that the fluctuation of the polarization $\langle m(t)^2\rangle - \langle m(t)\rangle^2$ is same in all three cases. In this parameter region, the autocorrelation time is very long in the original CT-INT method. The autocorrelation function decays rapidly with the number of global moves proposed by the SLMC. As discussed in Sec. II C, the autocorrelation time is expressed as $1 \sim 1/\langle A\rangle$ when the number of steps is longer than the autocorrelation time of the original simulation.

## VI. DISCUSSION

### Computational cost and fast updates

We estimate the computational cost of SLMC with BPNNs. There are two factors for evaluating this cost, the ratio of the computational cost $u_{\text{SLMC}}/u_{\text{original}}$ and the acceptance ratio $\langle A\rangle$, as shown in Sec. II C. The computational cost to calculate the effective Hamiltonian $H_{\text{eff}}(\mathcal{C})$ is estimated as follows. The computational cost to calculate the elementwise effective Hamiltonian $h_{\text{eff}}(\boldsymbol{c}^j)$ with the neural networks with one hidden layer is $O(N_u M)$ when the $M$-dimensional input vector $\boldsymbol{c}^j$ are given. The computational cost to calculate the coefficients $\boldsymbol{c}^j$ is $O(N)$ since there are $N$ $\delta$ functions shown in Eq. (24). We can reduce this cost from $O(N)$ to $O(1)$ with fast local updates. If we consider the insertion of the vertex, the new coefficient $c_m^{j,\text{new}}$ is calculated as $c_m^{j,\text{new}} = c_m^{j,\text{old}} + \phi_m(\tau_{iN+1})$, whose calculation cost is $O(1)$. With use of this local update, the total computational cost to calculate the effective Hamiltonian is $u_{\text{SLMC}} = O(N_u M N)$. Here, $N$ comes from the fact that there are $N$ input vectors $\boldsymbol{c}^j$. We note that the order of the computational cost of SLMC with BPNNs as a functional of $N$ is equivalent to that in SLMC in the previous paper for the CTAUX [12,47]. Since the calculation cost of the original CTINT simulation with fast updates [7,12] is $u_{\text{original}} = O(N^2)$, the SLMC is faster than the original simulation when the acceptance ratio $\langle A\rangle$ is not too small.

## VII. CONCLUSION

We developedthe SLMC method with BPNNs, which can be considered as a general scheme to construct effective Hamiltonians with many-body interactions even on continuous axis. We demonstrated the continuous-time interaction-expansion (CTINT) SLMC with BPNNs on quantum impurity models. The effective Hamiltonian without any prior knowledge was obtained. We obtained the significant improvement of the acceptance rate with respect to the SLMC with the effective Hamiltonian using explicit expression. This improvement implies that obtained effective Hamiltonian of SLMC with BPNNs includes many-body interaction effects, which is omitted in the effective Hamiltonians with the explicit forms. Our SLMC with BPNNs has many potential applications, since this method can accept both continuous and discrete indices of interacting particles as inputs of neural networks.

## ACKNOWLEDGMENTS

Y.N. would like to acknowledge H. Shen, Y. Qi, and L. Fu for helpful discussions and comments. The calculations were partially performed by the supercomputing systems SGI ICE X at the Japan Atomic Energy Agency. This work was partially supported by JSPS–KAKENHI Grants No. 18K03552 to Y.N. and No. 18K05208 to M.O. and the "Topological Materials Science" (No. 18H04228) JSPS–KAKENHI on Innovative Areas to Y.N.

## APPENDIX A: MARKOV PROCESS MONTE CARLO METHOD IN PHYSICS

The Markov chain Monte Carlo method (MCMC) is a powerful method for an integration in high-dimensional space. Partition functions and physical exception values can be calculated by MCMC. We describe the MCMC method used in physics and point out its problems and difficulties in the following.

### 1. Multidimensional integration

Let us consider the following multidimensional integration:

$$I = \int \dots \int dx_1 \dots dx_N w(x_1, \dots, x_N) f(x_1, \dots, x_N), \quad \text{(A1)}$$

where $w(x_1, \dots, x_N)$ is a localized function in $N$-dimensional space, and $f(x_1, \dots, x_N)$ is an arbitral function. In the Monte Carlo method, the integration $I$ is approximated as

$$I \sim \sum_{\mathcal{C}} f(x_1, \dots, x_N), \quad \text{(A2)}$$

where $\mathcal{C} = (x_1, \dots, x_N)$ is randomly generated with the probability $w(x_1, \dots, x_N)$, which is called a configuration.

In physics, the partition function $Z$ and physical exception value $\langle A\rangle$ are expressed by the multidimensional integration:

$$Z = \int \dots \int d\phi_1 \dots d\phi_N e^{-S(\phi_1,\dots,\phi_N)}, \quad \text{(A3)}$$

$$\langle A\rangle = \frac{1}{Z}\int \dots \int d\phi_1 \dots d\phi_N A(\phi_1, \dots, \phi_N) e^{-S(\phi_1,\dots,\phi_N)}. \quad \text{(A4)}$$

For example, the physical exception value $\langle A\rangle$ in the classical Ising model on one-dimensional lattice is expressed as

$$\langle A\rangle = \frac{1}{Z}\sum_{s_1,\dots,s_N} A(s_1, \dots, s_N) e^{-\beta E(s_1,\dots,s_N)}, \quad \text{(A5)}$$

$$\sim \frac{1}{Z}\sum_{\mathcal{C}} A(s_1, \dots, s_N). \quad \text{(A6)}$$

where $Z = \sum_{s_1,\dots,s_N} \exp[-\beta E(s_1, \dots, s_N)]$, $\beta$ is an inverse temperature, $N$ is a number of spins, $s_i = \pm 1$, $E(s_1, \dots, s_N) = J\sum_{\langle i,j\rangle} s_i s_j$, and $\sum_{\langle i,j\rangle}$ is a summation of the nearest-neighbor spins. The configuration $\mathcal{C} = (s_1, \dots, s_N)$ is randomly generated with the Boltzmann weight $\exp[-\beta E(s_1, \dots, s_N)]$.

### 2. Design of the proposals

One can design various kinds of the Monte Carlo method based on Eq. (4). We need the update method from $\mathcal{C}$ to $\mathcal{C}'$ with the high acceptance ratio for good efficiency of the MCMC.

#### *a. Hand-designed proposals: Conventional local and global updates*

The most simple update method is so-called local update, where the configuration is updated locally. One randomly chooses a single site in the current configuration and proposes a new configuration by changing the variable on this site. For example, in the Ising spin model, the candidate of the next configuration is generated by flipping a randomly chosen spin. In this case, the proposal probability from $C'$ to $C$ $g(\mathcal{C}|\mathcal{C}')$ equals 1 from $C$ to $C'$ $g(\mathcal{C}'|\mathcal{C})$. The acceptance ratio in Eq. (4) becomes $A(\mathcal{C}',\mathcal{C}) = \min(1, \frac{w(\mathcal{C}')}{w(\mathcal{C})})$. In the local update, one can expect that $w(\mathcal{C})$ is similar to $w(\mathcal{C}')$ so that the acceptance rate is high, since the configuration $\mathcal{C}$ is similar to $\mathcal{C}'$. Although the local update is a general-purpose, model-independent method, there is an evident disadvantage. When $w(\mathcal{C}')$ is not similar to $w(\mathcal{C})$ even if $\mathcal{C}'$ is similar to $\mathcal{C}$, it is hard to obtain uncorrelated configurations. Thus, the local update suffers heavily from a critical slowing down close to phase transitions. In such cases, the autocorrelation time within the Markov chain $\tau$ becomes very large.

To overcome the increase of the autocorrelation time for the local update, various kinds of global update method have been developed [37–41]. In all these global update methods, variables on a large number of sites are simultaneously changed in a single Monte Carlo update. The autocorrelation time $\tau$ can be reduced in these methods. The global update method is designed for specific model, since one has to use properties that a model has to obtain good efficiency. For example, the Wolff method can simulate the two-dimensional Ising model. However, by adding the interaction among the four spins in the same plaquette to this model, no simple and efficient global update method is known [9]. For a given generic model, it is hard to design an efficient global update method.

#### *b. Proposals by time evolution: Hybrid Monte Carlo updates*

The hybrid Monte Carlo (HMC) method is known as the one of the good global update methods, which is widely used in the lattice quantum chromodynamics (QCD). In the HMC, by introducing pseudomomentum, the pseudo-time-evolution from the configuration $\mathcal{C}$ generates the next configuration $\mathcal{C}'$, which is called a molecular dynamics (MD) evolution in analogy to simulations of classical particles. In this method, $g(\mathcal{C}|\mathcal{C}')$ equals $g(\mathcal{C}'|\mathcal{C})$, since the time evolution due to the Hamiltonian dynamics is time-reversal symmetric. The exact MD evolution due to the Hamilton equation conserves energy $\log w$. In actual simulations, the discretized time step breaks the energy conservation. Thus, the MD evolution generates the configuration $\mathcal{C}'$ whose probability $w(\mathcal{C}')$ is not equal but similar to $w(\mathcal{C})$ so that the acceptance ratio $A(\mathcal{C}',\mathcal{C}) = \min(1, \frac{w(\mathcal{C}')}{w(\mathcal{C})})$ is high. Recently, the HMC method is revisited in the condensed matter physics to treat strongly correlated electron systems [42], since the HMC method is general to obtain uncorrelated configurations. Although the HMC method might be suitable for reducing the autocorrelation time, its computational cost of the HMC is not small. One has to calculate the forces, which is the derivative of the energy to do the MD evolution. For example, in the lattice QCD simulation, the inverse matrices related to the Dirac operator on a four-dimensional lattice have to be calculated.

## APPENDIX B: MACHINE-LEARNING TECHNIQUE IN MOLECULAR DYNAMICS: BEHLER-PARRINELLO NEURAL NETWORKS

Machine learning techniques are widely used in the research fields of physics and chemistry. For example, material informatics has been developing rapidly [48]. Another remarkable development can be found in MD simulations. Machine learning techniques enable us to perform MD simulations in large systems with high accuracy comparable to quantum mechanical calculation (e.g., density functional theory) and low computational costs close to classical MD [49–55]. We briefly introduce the machine learning techniques in MD simulations in this section.

### 1. Potential energy surfaces

The basic procedure to realize large-system MD simulation is construction of PES in a large systems by patching PESs obtained in small systems. The outline is shown below. First, we prepare a small system consisting of $N^{(\mathrm{S})}$ atoms. Quantum mechanical calculations are performed to obtain an accurate PES $E^{(\mathrm{S})}_{\mathrm{tot}}(\mathcal{R}^{(\mathrm{S})})$, where $\mathcal{R}^{(\mathrm{S})} = \{\boldsymbol{r}_i \mid i = 1,\dots,N^{(\mathrm{s})}\}$ and $\boldsymbol{r}_i$ are a set representing an atom configuration and the coordinate of the $i$th atom, respectively.

Next, the total energy in the small system $E^{(\mathrm{S})}_{\mathrm{tot}}(\mathcal{R}^{(\mathrm{S})})$ is assumed to be divided into partial energies associated with each atoms in the system, which are determined by environment around each atoms, i.e.,

$$E^{(\mathrm{S})}_{\mathrm{tot}}(\mathcal{R}^{(\mathrm{S})}) = \sum_{i=1}^{N^{(\mathrm{s})}} E^{(\mathrm{S})}_{\mathrm{part}}\big(\Delta^{(\mathrm{S})}_i\big), \tag{B1}$$

where

$$\Delta^{(\mathrm{S})}_i = \{\boldsymbol{r}_{ij} \mid j = 1,\dots,i-1,i+1,\dots,N^{(\mathrm{s})}\} \tag{B2}$$

is a set representing an atom configuration around the $i$th atom ($\boldsymbol{r}_{ij} = \boldsymbol{r}_j - \boldsymbol{r}_i$).

### 2. Symmetry and descriptor

Here, we consider energy degeneracy due to translational and rotational invariance of the configurations, i.e., energies determined by a configuration and translated and rotated ones have a same value. Elimination of the degeneracy is desired for accurate evaluation of PES. Therefore, a set called descriptor (or fingerprint) [48] is introduced as

$$\mathcal{F}^{(\mathrm{S})}_i = \big\{F_I\big(\Delta^{(\mathrm{S})}_i\big)\,\big|\, I = 1,\dots,M\big\}, \tag{B3}$$

where $F_I$ and $M$ are an function $F_I : \Delta^{(\mathrm{S})}_i \to \mathbb{R}$ and the number of them, respectively. Using the descriptor, the total energy is given as

$$E^{(\mathrm{S})}_{\mathrm{tot}}(\mathcal{R}^{(\mathrm{S})}) = \sum_{i=1}^{N^{(\mathrm{s})}} E^{(\mathrm{S})}_{\mathrm{part}}\big(\mathcal{F}^{(\mathrm{S})}_i\big), \tag{B4}$$

The descriptor $\mathcal{F}^{(\mathrm{S})}_i$ is typically a set of functions of relative distances between atoms and angles between two vectors from one atom to other two atoms, which are obviously translational and rotational invariant. For example, the following function was proposed [34,56]:

$$F_I^{(\mathrm{d})}\big(\Delta_i^{(\mathrm{S})}\big) = \sum_{j=1,j\neq i}^{N^{(\mathrm{S})}} f_{i,j;I}^{(\mathrm{d})}, \tag{B5}$$

$$f_{i,j;I}^{(\mathrm{d})} = \exp[-\eta_I(r_{ij}-r_I)^2], \tag{B6}$$

where $r_{ij} = |\boldsymbol{r}_{ij}|$ and $\eta_I$ and $r_I$ are parameters. The function (B6) works as a detector of bond length around $r_I$, and it is obviously translational and rotational invariant. Here is an another example of the function [34,56],

$$F_I^{(\mathrm{a})}\big(\Delta_i^{(\mathrm{S})}\big) = \sum_{j=1,j\neq i}^{N^{(\mathrm{S})}} \sum_{k=1,k\neq i,j}^{N^{(\mathrm{S})}} f_{i,j,k;I}^{(\mathrm{a})}, \tag{B7}$$

$$f_{i,j,k;I}^{(\mathrm{a})} = 2^{1-\zeta_I}(1+\lambda_I\cos\theta_{ijk})^{\zeta_I}\mathrm{e}^{-\eta_I(r_{ij}+r_{ik})}, \tag{B8}$$

where $\theta_{ijk} = \boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ik}/r_{ij}r_{ik}$ and $\zeta_I$ and $\lambda_I$ are parameters. This function can detect the angle, and it is also translational and rotational invariant. These functions are usually used as elements of a descriptor, i.e.,

$$F_I = \begin{cases} F_I^{(\mathrm{d})} & (I = 1,\dots,m) \\ F_I^{(\mathrm{a})} & (I = m+1,\dots,M) \end{cases}. \tag{B9}$$

### 3. Extension from small systems to large systems

Toward the extension from the small systems to the large system, we introduce a further assumption: the partial energy $E_{\mathrm{part}}^{(\mathrm{S})}(\mathcal{F}_i^{(\mathrm{S})})$ can be approximately determined by the configuration of the atoms whose distances from the $i$th atom are less than a cutoff radius $r_{\mathrm{c}}$. It is realized by introducing a new descriptor

$$\mathcal{G}_i^{(\mathrm{S})}(r_{\mathrm{c}}) = \big\{G_I\big(\Delta_i^{(\mathrm{S})}; r_{\mathrm{c}}\big)\,\big|\, I = 1,\dots,M\big\}, \tag{B10}$$

where the function $G_I$ satisfies the following condition:

$$G_I\big(\tilde{\Delta}_i^{(\mathrm{S})}; r_{\mathrm{c}}\big) = 0\,, \tag{B11}$$

$$\tilde{\Delta}_i^{(\mathrm{S})}(r_{\mathrm{c}}) = \{\boldsymbol{r}_{ij}\,|\, r_{ij} > r_{\mathrm{c}},\ j = 1,\dots,i-1,i+1,\dots,N^{(\mathrm{S})}\}. \tag{B12}$$

Using the functions, we obtain the following equation:

$$E_{\mathrm{part}}^{(\mathrm{S})}\big(\mathcal{F}_i^{(\mathrm{S})}\big) \simeq E_{\mathrm{part}}^{(\mathrm{S})}\big(\mathcal{G}_i^{(\mathrm{S})}(r_{\mathrm{c}})\big), \tag{B13}$$

This assumption means that the partial energies are determined by not global configurations $\Delta_i^{(\mathrm{S})}$ but local ones $\Delta_i^{(\mathrm{S})} \setminus \tilde{\Delta}_i^{(\mathrm{S})}(r_{\mathrm{c}}) = \{\boldsymbol{r}_{ij}\,|\, r_{ij} \leqslant r_{\mathrm{c}},\ j = 1,\dots,i-1,i+1,\dots,N^{(\mathrm{S})}\}$. Note that the assumption is justified when there is no long-range interaction among the atoms or long-range interactions are screened in a system. We show an example of functions for the local descriptor below,

$$G_I = \begin{cases} G_I^{(\mathrm{d})} & (I = 1,\dots,m) \\ G_I^{(\mathrm{a})} & (I = m+1,\dots,M) \end{cases}, \tag{B14}$$

$$G_I^{(\mathrm{d})}\big(\Delta_i^{(\mathrm{S})}; r_{\mathrm{c}}\big) = \sum_{j=1,j\neq i}^{N^{(\mathrm{S})}} f_{i,j;I}^{(\mathrm{d})}\,\xi_{ij}(r_{\mathrm{c}}), \tag{B15}$$

$$G_I^{(\mathrm{a})}\big(\Delta_i^{(\mathrm{S})}; r_{\mathrm{c}}\big) = \sum_{j,k=1,j,k\neq i,k\neq j}^{N^{(\mathrm{S})}} f_{i,j,k;I}^{(\mathrm{a})}\,\xi_{ij}(r_{\mathrm{c}})\,\xi_{ij}(r_{\mathrm{c}}), \tag{B16}$$

$$\xi_{ij}(r_{\mathrm{c}}) = \begin{cases} \frac{1}{2}\big[\cos\big(\frac{\pi r_{ij}}{r_{\mathrm{c}}}\big)+1\big] & (r_{ij} \leqslant r_{\mathrm{c}}) \\ 0 & (r_{ij} > r_{\mathrm{c}}) \end{cases}. \tag{B17}$$

Finally, we obtain the following partitioning of the total energy to the partial energies:

$$E_{\mathrm{tot}}^{(\mathrm{S})}(\mathcal{R}^{(\mathrm{S})}) = \sum_{i=1}^{N^{(\mathrm{S})}} E_{\mathrm{part}}^{(\mathrm{S})}\big(\mathcal{G}_i^{(\mathrm{S})}(r_{\mathrm{c}})\big), \tag{B18}$$

This partitioning enables us to extend MD simulations from the small systems to the large system by constructing the total energy in the large system $E_{\mathrm{tot}}^{(\mathrm{L})}$ as

$$E_{\mathrm{tot}}^{(\mathrm{L})}(\mathcal{R}^{(\mathrm{L})}) = \sum_{i=1}^{N^{(\mathrm{L})}} E_{\mathrm{part}}^{(\mathrm{S})}\big(\mathcal{G}_I^{(\mathrm{L})}(r_{\mathrm{c}})\big), \tag{B19}$$

where

$$\mathcal{R}^{(\mathrm{L})} = \{\boldsymbol{r}_i\,|\, i = 1,\dots,N^{(\mathrm{L})}\}, \tag{B20}$$

$$\mathcal{G}_i^{(\mathrm{L})}(r_{\mathrm{c}}) = \big\{G_I\big(\Delta_i^{(\mathrm{L})}; r_{\mathrm{c}}\big)\,\big|\, I = 1,\dots,M\big\}, \tag{B21}$$

$$\Delta_i^{(\mathrm{L})} = \{\boldsymbol{r}_{ij}\,|\, j = 1,\dots,i-1,i+1,\dots,N^{(\mathrm{L})}\}, \tag{B22}$$

and $N^{(\mathrm{L})}$ is the number of atoms in the large system. We mention that the function $E_{\mathrm{part}}^{(\mathrm{S})}$ defined in the small system works even with the local descriptor in the large system $\mathcal{G}_I^{(\mathrm{L})}(r_{\mathrm{c}})$. Because it contains information of the atom configuration within the cutoff $r_{\mathrm{c}}$, which is consistent with the local descriptor in the small system.

### 4. Behler-Parrinello neural networks

Now we have a method of the extension. The remaining problem is how to obtain the function $E_{\mathrm{part}}^{(\mathrm{S})}$. Machine learning techniques give a solution of the problem. Behler and Parrinello (BP) proposed an energy partitioning method using neural networks [34], i.e., the partial energies are determined by neural networks. Although this approach loses explicit physical meaning of the partial energies, it is effective.

We mention that there was a serious problem about the descriptors. The number of the descriptors blows up as increase of the number of atom types. For example, the number of descriptors is in proportion to ${}_{N_{\mathrm{t}}}C_3$ if they contain the angles between the centered atom and other two atoms when they have $N_{\mathrm{t}}$ types. Artrith *et al.* solved the problem by introducing new descriptor using the Chebyshev polynomials [35]. The new descriptor expresses information of the radial and angular distribution functions including information of the atom types by Chebyshev coefficients. They were succeeded in providing accurate energies of materials containing eleven atom types [35].

## APPENDIX C: BATCH-ATOM NORMALIZATION

We describe the detail of the batch-atom normalization in the CTINT QMC simulation. We consider neural networks with one hidden layer. The input vector is $\boldsymbol{c}^j =$

$(c^j_0, \dots, c^j_{m_{\rm cut}-1}, d^j_0, \dots, d^j_{m_{\rm cut}-1})^T$. Here, the coefficients are defined as

$$c^j_m = \sum_{i=1}^{N} \phi_m(\tau_{ij}), \tag{C1}$$

$$d^j_m = \sum_{i=1}^{N} s_i s_j \phi_m(\tau_{ij}), \tag{C2}$$

with the pseudospin index $s_i$. The effective Hamiltonian is defined as

$$H_{\rm eff}(\mathcal{C}) = \frac{1}{N}\sum_{j=1} h_{\rm eff}(\boldsymbol{c}^j) + f(N), \tag{C3}$$

where

$$h_{\rm eff}(\boldsymbol{c}^j) = \hat{W}_2 F(\hat{W}_1 \boldsymbol{c}^j + \boldsymbol{b}_1) + b_2, \tag{C4}$$

$$= \sum_{j_1=1}^{N_u} [\hat{W}_2]_{j_1} F\left([\boldsymbol{x}]^j_{j_1}\right) + b_2, \tag{C5}$$

where

$$[\boldsymbol{x}]^j_{j_1} \equiv \sum_{j_2=1}^{M} [\hat{W}_1]_{j_1 j_2} [\boldsymbol{c}^j]_{j_2} + [\boldsymbol{b}_1]_{j_1}. \tag{C6}$$

Here, $[F(\boldsymbol{x})]_i = F([\boldsymbol{x}]_i)$ denotes the activation function, $\hat{W}_1$ is the $N_u \times M$ matrix, $\hat{W}_2$ is $1 \times N_u$ matrix, $\boldsymbol{b}_1$ is the $N_u$-dimensional bias vector, $b_2$ is the bias, and $M$ is a number of coefficients $\boldsymbol{c}^j$. We introduce the batch-atom normalization function $G$ as

$$h_{\rm eff}(\boldsymbol{c}^j) = \sum_{j_1=1}^{N_u} [\hat{W}_2]_{j_1} G\left(F\left([\boldsymbol{x}]^j_{j_1}\right)\right) + b_2, \tag{C7}$$

where

$$G\left(F\left([\boldsymbol{x}]^j_{j_1}\right)\right) = \gamma_{j_1} \frac{F\left([\boldsymbol{x}]^j_{j_1}\right) - \mu_{j_1}}{\sqrt{\sigma^2_{j_1} + \epsilon^2}} + \beta_{j_1}. \tag{C8}$$

Here, the parameters $\gamma_{j_1}$ and $\beta_{j_1}$ are trainable parameters. $\epsilon$ is a small number. The batch-atom mean $\mu_{j_1}$ and variance $\sigma_{j_1}$ are defined as

$$\mu_{j_1} = \frac{1}{N N_{\rm batch}} \sum_{l=1}^{N_{\rm batch}} \sum_{j=1}^{N} F\left([\boldsymbol{x}^l]^j_{j_1}\right), \tag{C9}$$

$$\sigma^2_{j_1} = \frac{1}{N N_{\rm batch}} \sum_{l=1}^{N_{\rm batch}} \sum_{j=1}^{N} \left(F\left([\boldsymbol{x}^l]^j_{j_1}\right) - \mu_{j_1}\right)^2. \tag{C10}$$

The index $l$ in $\boldsymbol{x}^l$ is the index of the training data. Here, $N_{\rm batch}$ is the number of the batch size of the training data.

[1] R. Blankenbecler, D. J. Scalapino, and R. L. Sugar, Monte Carlo calculations of coupled boson-fermion systems. I, Phys. Rev. D **24**, 2278 (1981).

[2] J. E. Hirsch, Two-dimensional Hubbard model: Numerical simulation study, Phys. Rev. B **31**, 4403 (1985).

[3] J. E. Hirsch and R. M. Fye, Monte Carlo Method for Magnetic Impurities In Metals, Phys. Rev. Lett. **56**, 2521 (1986).

[4] S. R. White, D. J. Scalapino, R. L. Sugar, E. Y. Loh, J. E. Gubernatis, and R. T. Scalettar, Numerical study of the two dimensional Hubbard model, Phys. Rev. B **40**, 506 (1989).

[5] A. N. Rubtsov, V. V. Savkin, and A. I. Lichtenstein, Continuous time quantum Monte Carlo method for fermions, Phys. Rev. B **72**, 035122 (2005).

[6] P. Werner, A. Comanac, L. de Medici, M. Troyer, and A. J. Millis, Continuous-Time Solver for Quantum Impurity Models, Phys. Rev. Lett. **97**, 076405 (2006).

[7] E. Gull, A. J. Millis, A. I. Lichtenstein, A. N. Rubtsov, M. Troyer, and P. Werner, Continuous-time Monte Carlo methods for quantum impurity models, Rev. Mod. Phys. **83**, 349 (2011).

[8] E. Gull, P. Werner, O. Parcollet, and M. Troyer, Continuous-time auxiliary-field Monte Carlo for quantum impurity models, Europhys. Lett. **82**, 57003 (2008).

[9] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, Self-learning Monte Carlo method, Phys. Rev. B **95**, 041101(R) (2017).

[10] J. Liu, H. Shen, Y. Qi, Z. Y. Meng, and L. Fu, Self-learning Monte Carlo method and cumulative update in fermion systems, Phys. Rev. B **95**, 241104(R) (2017).

[11] X. Y. Xu, Y. Qi, J. Liu, L. Fu, and Z. Y. Meng, Self-learning quantum Monte Carlo method in interacting fermion systems, Phys. Rev. B **96**, 041119(R) (2017).

[12] Y. Nagai, H. Shen, Y. Qi, J. Liu, and L. Fu, Self-learning Monte Carlo method: Continuous-time algorithm, Phys. Rev. B **96**, 161102(R) (2017).

[13] L. Huang and L. Wang, Accelerated Monte Carlo simulations with restricted Boltzmann machines, Phys. Rev. B **95**, 035105 (2017).

[14] Z. H. Liu, X. Y. Xu, Y. Qi, K. Sun, Z. Y. Meng, Itinerant quantum critical point with frustration and non-Fermi-liquid, Phys. Rev. B **98**, 045116 (2018).

[15] C. Chen, X. Y. Xu, J. Liu, G. Batrouni, R. Scalettar, and Z. Y. Meng, Symmetry-enforced self-learning Monte Carlo method applied to the Holstein model, Phys. Rev. B **98**, 041102(R) (2018).

[16] L. Huang, Y.-f. Yang, and L. Wang, Recommender engine for continuous-time quantum Monte Carlo methods, Phys. Rev. E **95**, 031301(R) (2017).

[17] A. Tanaka and A. Tomiya, Towards reduction of autocorrelation in HMC by machine learning, arXiv:1712.03893.

[18] L. Zdeborova, Machine learning: New tool in the box, Nature Phys. **13**, 420 (2017).

[19] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nature Phys. **13**, 431 (2017).

[20] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, 602 (2017).

[21] A. Tanaka and A. Tomiya, Detection of phase transition via convolutional neural networks, J. Phys. Soc. Jpn. **86**, 063001 (2017).

[22] E. P. L. van Nieuwenburg, Y.-H. Liu, and Sebastian D. Huber, Learning phase transitions by confusion, Nature Phys. **13**, 435 (2017).

[23] Y. Zhang and E.-A. Kim, Quantum Loop Topography for Machine Learning, Phys. Rev. Lett. **118**, 216401 (2017).

[24] J. Chen, S. Cheng, H. Xie, L. Wang, and T. Xiang, Equivalence of restricted Boltzmann machines and tensor network states, Phys. Rev. B **97**, 085104 (2018).

[25] D.-L. Deng, X. Li, and S. Das Sarma, Quantum Entanglement in Neural Network States, Phys. Rev. X **7**, 021021 (2017).

[26] Y. Huang and J. E. Moore, Neural network representation of tensor network and chiral states, arXiv:1701.06246.

[27] W. Hu, R. R. P. Singh, R. T. Scalettar, Discovering phases, phase transitions and crossovers through unsupervised machine learning: A critical examination, Phys. Rev. E **95**, 062122 (2017).

[28] Z. Cai, Approximating quantum many-body wave-functions using artificial neural networks, Phys. Rev. B **97**, 035116 (2018).

[29] H. Fujita, Y. O. Nakagawa, S. Sugiura, and M. Oshikawa, Construction of Hamiltonians by machine learning of energy and entanglement spectra, Phys. Rev. B **97**, 075114 (2018).

[30] S. J. Wetzel and M. Scherzer, Machine learning of explicit order parameters: From the Ising model to SU(2) lattice gauge theory, Phys. Rev. B **96**, 184410 (2017).

[31] S. Iso, S. Shiba, and S. Yokoo, Scale-invariant feature extraction of neural network and renormalization group flow, Phys. Rev. E **97**, 053304 (2018).

[32] T. Mano and T. Ohtsuki, Phase diagrams of three-dimensional anderson and quantum percolation models using deep three-dimensional convolutional neural network, J. Phys. Soc. Jpn. **86**, 113704 (2017).

[33] H. Shen, J. Liu, and L. Fu, Self-learning Monte Carlo with deep neural networks, Phys. Rev. B **97**, 205140 (2018).

[34] J. Behler and M. Parrinello, Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces, Phys. Rev. Lett. **98**, 146401 (2007).

[35] N. Artrith, A. Urban, and G. Ceder, Efficient and accurate machine-learning interpolation of atomic energies in compositions with many species, Phys. Rev. B **96**, 014112 (2017).

[36] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. (Springer, Berlin, 2006).

[37] R. H. Swendsen and J.-S. Wang, Nonuniversal Critical Dynamics in Monte Carlo Simulations, Phys. Rev. Lett. **58**, 86 (1987).

[38] U. Wolff, Collective Monte Carlo Updating for Spin Systems, Phys. Rev. Lett. **62**, 361 (1989).

[39] N. Prokof'ev, B. Svistunov, and I. Tupitsyn, "Worm" algorithm in quantum Monte Carlo simulation, Phys. Lett. A **238**, 253 (1998).

[40] H. G. Evertz, G. lana, and M. Marcu, Cluster Algorithm for Vertex Models, Phys. Rev. Lett. **70**, 875 (1993).

[41] H. G. Evertz, The loop algorithm, Adv. Phys. **52**, 1 (2003).

[42] S. Beyl, F. Goth, and F. F. Assaad, Revisiting the hybrid quantum Monte Carlo method for Hubbard and electron-phonon models, Phys. Rev. B **97**, 085144 (2018).

[43] F. F. Assaad and T. C. Lang, Diagrammatic determinantal quantum Monte Carlo methods: Projective schemes and applications to the Hubbard-Holstein model, Phys. Rev. B **76**, 035116 (2007).

[44] F. F. Assaad, in *Continuous-time QMC Solvers for Electronic Systems in Fermionic and Bosonic Baths, in DMFT at 25: Infinite Dimensions: Lecture Notes of the Autumn School on Correlated Electrons 4*, edited by E. Pavarini, E. Koch, D. Vollhardt, and A. Lichtenstein (Forschungszentrum Julich, 2014).

[45] D. J. Luitz and F. F. Assaad, Weak-coupling continuous-time quantum Monte Carlo study of the single impurity and periodic Anderson models with s-wave superconducting baths, Phys. Rev. B **81**, 024509 (2010).

[46] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning* (JMLR, 2015), Vol. 37, pp. 448–456.

[47] There is a relation between CTINT and CTAUX. The perturbation order becomes same when the parameter of the CTAUX has the relation: $K = U\beta(\delta^2 - 1/4)$ [44].

[48] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, and C. Kim, Machine learning in materials informatics: Recent applications and prospects, npj Comp. Mater. **3**, 54 (2017).

[49] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren, Neural network models of potential energy surfaces, J. Chem. Phys. **103**, 4129 (1995).

[50] D. F. R. Brown, Combining *ab initio* computations, neural networks, and diffusion Monte Carlo: An efficient method to treat weakly bound molecules, J. Chem. Phys. **105**, 7597 (1996).

[51] S. Lorenz, A. Groß, and M. Scheffler, Representing high-dimensional potential-energy surfaces for reactions at surfaces by neural networks, Chem. Phys. Lett. **395**, 210 (2004).

[52] N. Artrith, T. Morawietz, and J. Behler, High-dimensional neural-network potentials for multicomponent systems: Applications to zinc oxide, Phys. Rev. B **83**, 153101 (2011).

[53] N. Artrith and A. M. Kolpak, Understanding the composition and activity of electrocatalytic nanoalloys in aqueous solvents: A combination of DFT and accurate neural network potentials, Nano Lett. **14**, 2670 (2014).

[54] W. Li, Y. Ando, and S. Watanabe, Cu diffusion in amorphous Ta2O5 studied with a simplified neural network potential, J. Phys. Soc. Jpn. **86**, 104004 (2017).

[55] W. Li and Y. Ando, Construction of accurate machine learning force fields for copper and silicon dioxide, arXiv:1807.02042.

[56] J. Behler, Constructing high-dimensional neural network potentials: A tutorial review, Int. J. Quantum Chem. **115**, 1032 (2015).