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

Yuki Nagai,[1] 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*

(Dated: July 16, 2018)

Self-learning Monte Carlo method (SLMC) is a general-purpose numerical method that 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 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 construct self-learning continuous-time interaction-expansion quantum Monte Carlo method with BPNNs and 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.

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 made great successes for strongly correlated electron systems[5–8]. In this algorithm, the partition function is expanded in the powers 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 fermions, which is very time consuming.

Recently, we introduced a general method called self-learning Monte Carlo (SLMC), which speeds up the MC simulation by designing and training a model to propose efficient global updates, first in classical statistical mechanics models[9, 10], later extended to classical spin-fermion models[11], determinant QMC[12, 13], continuous-time QMC[14, 15] and hybrid MC in high-energy physics[16]. The SLMC is one of the successes in machine learning techniques in physics [17–31]. The philosophy behind SLMC is "first learn, then earn". 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 $H_{\rm eff}$, 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, 11–13, 15]. However, it is not easy to construct effective models including many body interaction, since we do not know systematic procedure applicable in arbitrary systems. Recently, H. Shen *et al.* gave pioneering work to construct the effective Hamiltonian including many body interaction without explicit forms using the convolutional deep neural network (CNN) [32]. Although the CNN method is powerful, its applicability is limited to the system whose particle configurations are given by discrete indices. This is because the inputs of the CNN is usually discretized like pixels in a digital picture. For example, SLMC with the CNN cannot be applied to the continuous-time QMC, since the inputs of the continuous-time QMC have both discrete position on a lattice and continuous imaginary-time. Therefore, further extension of SLMC with neural networks are needed.

The machine learning and neural networks have been used for about twenty years in the field of the molecular dynamics (MD) to construct the potential-energy surfaces (PESs) providing inter-atom 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). 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 allow 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)[36], which is one of most succeeded methods in the field of the molecular dynamics with machine learning. We regard the 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 [37] to map continuous coordinates (atom positions) onto discrete (inputs of BPNN) variables, 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.

Continuous-time QMC for fermions. The partition function Z is calculated by the Monte Carlo summations expressed as

$$Z = \sum_{\mathcal{C}} W(\mathcal{C}). \tag{1}$$

Here, $\mathcal{C}$ is a configuration. In continuous-time QMC simulations, by splitting the Hamiltonian into non-perturbative and perturbative parts $H = H_0 + H_1$, the partition function is expanded as

$$\begin{aligned} Z &= \mathrm{Tr}\left[e^{-\beta H_0} T_\tau e^{-\int_0^\beta H_1(\tau)d\tau}\right], &(2)\\ &= \sum_{N=0}^{\infty} \frac{(-1)^N}{N!} \int_0^\beta d\tau_1 \cdots \int_0^\beta d\tau_N \mathrm{Tr}\left[T_\tau e^{-\beta H_0} H_1(\tau_N)\right. \\ &\times \left. H_{N-1}(\tau_{N-1}) \cdots H_1(\tau_1)\right], &(3)\\ &= \sum_{\mathcal{C}} W(\mathcal{C}). &(4) \end{aligned}$$

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

Effective Hamiltonian in SLMC. We assume that there is an effective Hamiltonian H_{eff} which describes the weight $W(\mathcal{C})$ in a continuous-time QMC simulation:

$$W^{\mathrm{eff}}(\mathcal{C}) = \exp\left[-\beta H_{\mathrm{eff}}(\mathcal{C})\right] \sim W(\mathcal{C}). \tag{5}$$

In the SLMC method, this effective model is used as a guide to propose highly efficient global moves in configuration space. Starting from a given configuration $\mathcal{C}_1$, one first performs standard Monte Carlo simulations on the effective model $\mathcal{C}_1 \to \mathcal{C}_2 \to \cdots \to \mathcal{C}_n$. Configuration $\mathcal{C}_n$ is accepted in the Metropolis-Hastings algorithm with the probability [9, 11, 32]:

$$p(\mathcal{C}_1 \to \mathcal{C}_n) = \min\left(1, \frac{W(\mathcal{C}_n)}{W(\mathcal{C}_1)} \frac{W^{\mathrm{eff}}(\mathcal{C}_1)}{W^{\mathrm{eff}}(\mathcal{C}_n)}\right). \tag{6}$$

The average acceptance rate $\langle p \rangle$ can be estimated by $\langle p \rangle = \exp\left[-\sqrt{\mathrm{MSE}}\right]$ with the mean squared error $\mathrm{MSE} = (1/n)\sum_i (\ln W^{\mathrm{eff}}(\mathcal{C}_i) - \ln W(\mathcal{C}_i))^2$ [32]. If the effective model is perfect, the global move is always accepted. It is important to obtain good effective models in the SLMC simulations. In the previous study[15], 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[14]. However, it seems hard to construct the effective Hamiltonian in other systems or other methods.

The problem in SLMC is how to construct effective potential $\ln W^{\mathrm{eff}}(\mathcal{C})$ to fit the potential $\ln W(\mathcal{C})$ as a functional of $\mathcal{C}$. In the field of MD, the machine learning and neural networks have been used for about twenty years to construct effective inter-atom or inter-molecule potentials to fit the PESs calculated by the first-principle calculations. We point out that the BPNNs can be considered as a general scheme to construct effective Hamiltonians. We show the method to reproduce the effective Hamiltonian with many-body interactions in SLMC as follows.

We have to find the map from the configuration $\mathcal{C}$ to the weight $\ln W^{\mathrm{eff}}(\mathcal{C})$. The configuration with N-th order in CTQMC includes N vertices on the imaginary-time axis $\mathcal{C} = \{\tau_1, \tau_2, \cdots, \tau_N\}$. Now we omit other degrees of freedom for simplicity. In the MD with BPNNs, the configuration includes the position of atoms $\mathcal{C}^{\mathrm{MD}} = \{\boldsymbol{r}_1, \boldsymbol{r}_2, \cdots, \boldsymbol{r}_N\}$. To implement the translational symmetry, the distance between two atoms is a key quantity. Thus, we map the configuration $\mathcal{C}$ onto the set of the local configurations $\mathcal{C}^{\mathrm{loc}}(\tau_i - \tau_j)$ around a vertex j: $\mathcal{C} \to \{\mathcal{C}^{\mathrm{loc}}(\tau_1 - \tau_j), \mathcal{C}^{\mathrm{loc}}(\tau_2 - \tau_j), \cdots, \mathcal{C}^{\mathrm{loc}}(\tau_N - \tau_j)\}$. The local configuration is expressed by the basis functions. We introduce the density distribution functions defined as

$$\rho(\tau, \mathcal{C}^{\mathrm{loc}}(\tau_i - \tau_j)) = \sum_{i=1}^{N} \delta(\tau - \tau_{ij}), \tag{7}$$

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 [37]:

$$\rho(\tau, \mathcal{C}^{\mathrm{loc}}(\tau_i - \tau_j)) = \sum_m c_m^j \phi_m(\tau), \tag{8}$$

with $c_m^j = \sum_i^N \phi_m(\tau_{ij})$. Here, $\phi_m(x)$ is the Chebyshev polynomial function $\phi_m(x) = \cos(m \arccos(x))$. Thus,

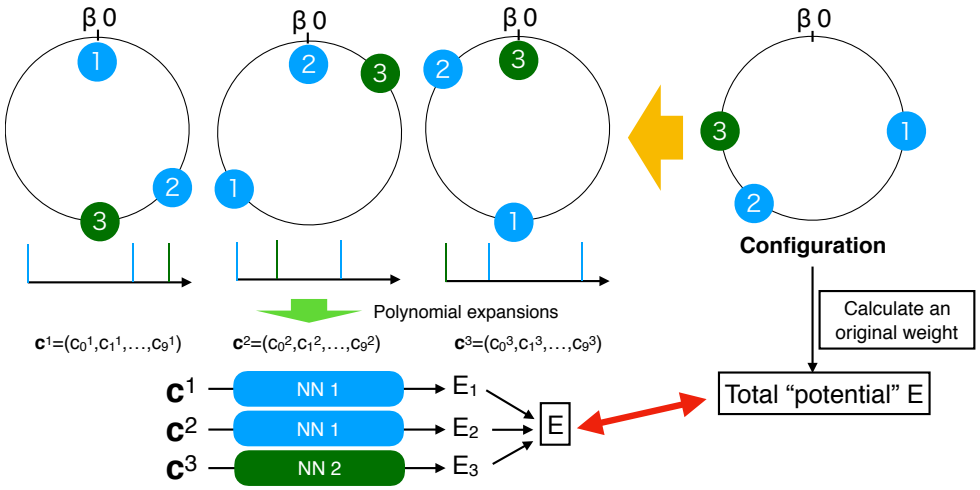

FIG. 1. (Color online) Schematic figures of Behler-Parrinello neural networks. The configuration with N vertices on the imaginary-time axis $\mathcal{C} = \{\tau_1, \tau_2, \cdots, \tau_N\}$ is mapped to the set of the local configurations $\mathcal{C}^{\rm loc}(\tau_i - \tau_j)$ around a vertex j: $\mathcal{C} \to \{\mathcal{C}^{\rm loc}(\tau_1 - \tau_j), \mathcal{C}^{\rm loc}(\tau_2 - \tau_j), \cdots, \mathcal{C}^{\rm loc}(\tau_N - \tau_j)\}$. The local configuration is expressed as a distribution function with δ 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 $E_i = h^{\alpha_i}_{\rm eff}(\boldsymbol{c}^j)/(-\beta)$.

we map the local configuration $\mathcal{C}^{\rm loc}(\tau_i - \tau_j)$ onto the set of m_{cut} coefficients of the Chebyshev polynomials $\mathcal{C}^{\rm loc}(\tau_i - \tau_j) \to \{c^j_0, \cdots, c^j_{m_{cut}-1}\}$, as shown in Fig. 1. The effective Hamiltonian with the configuration $\mathcal{C}$ is introduced as

$$H_{\rm eff}(\mathcal{C}) = \frac{1}{N}\sum_j h^{\alpha_j}_{\rm eff}(\boldsymbol{c}^j) + f(N), \tag{9}$$

with $\boldsymbol{c}^j = (c^j_0, \cdots, c^j_{m_{cut}-1})^T$. Here, $f(N)$ is a polynomial function $f(N) = \sum_{k=0}^{n_{\max}} f_k N^k$. Note that the nonlinear function $h^{\alpha_j}_{\rm eff}(\boldsymbol{c}^j)$ does not depend on j and only depends on a kind of atoms α_j, since we assume that the same kind of atoms feels same interactions as shown in Fig. 1. In the cases of the CTAUX and CTINT, we add other density distribution $\rho_s(\tau, \mathcal{C}^{\rm loc}(\tau_i - \tau_j, s_j, \{s_i\})) = \sum_m d^j_m s_i s_j \phi_m(\tau)$ to express the pseudo spin degree of freedom. In these cases, the vector is $\boldsymbol{c}^j = (c^j_0, \cdots, c^j_{m_{cut}-1}, d^j_0, \cdots, d_{m_{cut}-1})^T$[44].

The above effective Hamiltonian can reproduce the effective interaction used in the previous paper for the CTAUX[15]. If we assume $h_{\rm eff}(\boldsymbol{c}^j)$ is linear $h_{\rm eff}(\boldsymbol{c}^j) = \hat{W}\boldsymbol{c}^j + b$, the effective Hamiltonian is rewritten as $H_{\rm eff}(\mathcal{C}) = \sum_{ij} L(\tau_i j)/N + \sum_{ij} s_i s_j J(\tau_{ij})/N + f(N)$, where the effective two-body interactions are written as $L(\tau_{ij}) = \sum_{m=0}^{m_{cut}-1} [\hat{W}]_m \phi_m(\tau_{ij})$ and $J(\tau_{ij}) = \sum_{m=0}^{m_{cut}-1} [\hat{W}]_{m+m_{cut}} \phi_m(\tau_{ij})$.

Using neural networks Let us introduce the BPNNs. For example, in the case of the neural networks with one hidden layer with N_u units, the effective local Hamiltonian is expressed as

$$h^{\alpha_j}_{\rm eff}(\boldsymbol{c}^j) = \hat{W}^{\alpha_j}_2 F\left(\hat{W}^{\alpha_j}_1 \boldsymbol{c}^j + \boldsymbol{b}^{\alpha_j}_1\right) + b^{\alpha_j}_2, \tag{10}$$

with the activation function $[F(\boldsymbol{x})]_i = F([\boldsymbol{x}]_i)$. Here, $\hat{W}^{\alpha_j}_1$ is the $N_u \times M$ matrix, $\hat{W}^{\alpha_j}_2$ is $1 \times N_u$ matrix, $\boldsymbol{b}^{\alpha_j}_1$ is the N_u-dimensional bias vector, $b^{\alpha_j}_2$ is the bias, and M is a number of coefficients $\boldsymbol{c}^j$. The activation function $F(x)$ makes the effective Hamiltonian nonlinear, which can represent many-body interactions.

Demonstrations on quantum impurity models. We demonstrate self-learning CTINT with BPNNs on impurity model. We consider the Hamiltonian of the single impurity Anderson model, which is written as the combination of a free fermion part and interaction part[7, 15],

$$H = -\mu\sum_\sigma n_\sigma + \sum_{\sigma,p}(V c^\dagger_\sigma a_{p,\sigma} + \text{H.c.}) + \sum_{\sigma,p}\epsilon_p a^\dagger_{p,\sigma} a_{p,\sigma} + U n_\uparrow n_\downarrow, \tag{11}$$

where $\sigma = \uparrow, \downarrow$, $c^\dagger_\sigma$ and $a^\dagger_{p,\sigma}$ are the fermion creation operators for an impurity electron with spin σ, and that for a bath electron with spin σ and momentum p, respectively. n_σ 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

$$H_1 = \frac{U}{2}\sum_{s=\pm 1}\left(n_\uparrow - \frac{\rho}{2} - s\delta\right)\left(n_\downarrow - \frac{\rho}{2} + s\delta\right). \tag{12}$$

Here, we introduce additional Ising variable s and parameter δ, and ρ corresponds to the average electron density[40–42]. We consider the half filling ($\rho = 1$). The non-perturbative part is $H_0 = \left(-\mu + \frac{U}{2}\rho\right)\sum_\sigma n_\sigma +$

$\sum_{\sigma,p}(Vc^\dagger_\sigma a_{p,\sigma} + \text{H.c.}) + \sum_{\sigma,p}\epsilon_p a^\dagger_{p,\sigma}a_{p,\sigma}$. The partition function is expanded as

$$\frac{Z}{Z_0} = \sum_{\mathcal{C}} W(\mathcal{C}) = \sum_{\mathcal{C}} \left(\frac{-U}{2}\right)^N \frac{1}{N!} \prod_\sigma \det M_\sigma(\mathcal{C}). \quad (13)$$

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 and gather training data in this code. We train the neural networks with one hidden layer as shown in Fig. 2 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 [43] 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[46]. The input vector is $M = 2m_{\text{cut}}$ dimensional vector: $\boldsymbol{c}^j = (c^j_0, \cdots, c^j_{m_{cut}-1}, d^j_0, \cdots, d_{m_{cut}-1})^T$. The total number of parameters in neural networks with one hidden layer with N_u units is $MN_u + N_u + 4N_u + N_u + n_{\max} + 1 = MN_u + 6N_u + n_{\max} + 1$.

Figure 3 shows the inverse-temperature dependence of the acceptance ratio of the SLMC. We consider $U = 3D$, $\delta = 0.5$, $V = 1D$, and $n_{\max} = 3$. We set the number of the SLMC steps n defined in Eq. (6) is $n = 500$. We also consider the linear SLMC, which is equivalent to the previous effective Hamiltonian[15]. 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.

Computational cost and fast updates. We estimate the computational cost of SLMC with BPNNs. The computational cost to calculate the local effective Hamiltonian $h_{\text{eff}}(\boldsymbol{c}^j)$ with the neural networks with one hidden layer is $\mathcal{O}(N_u M)$ when the M-dimensional input vector $\boldsymbol{c}^j$ are given. The computational cost to calculate the coefficients $\boldsymbol{c}^j$ is $\mathcal{O}(N)$ since there are N δ-functions shown in Eq. (8). Thus, the computational cost to calculate the effective Hamiltonian $H_{\text{eff}}(\mathcal{C})$ is $\mathcal{O}(N_u M N^2)$, whose order is equivalent to that in the original CTINT simulation with fast updates[7, 15]. To speedup SLMC, we reduce the cost to calculate the coefficients $\mathcal{O}(N)$ to $\mathcal{O}(1)$ with fast local updates. If we consider the insertion of the vertex, the new coefficient $c^{j,\text{new}}_m$ is calculated as $c^{j,\text{new}}_m = c^{j,\text{old}}_m + \phi_m(\tau_{iN+1})$, whose calculation cost is $\mathcal{O}(1)$. With use of this local update, the total computational cost to calculate the effective Hamiltonian is $\mathcal{O}(N_u MN)$. Therefore, 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[15].

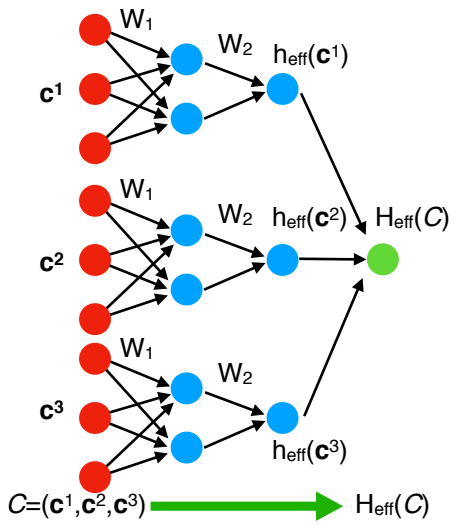

FIG. 2. (Color online) Schematic figures of input and output data in Behler-Parrinello neural networks. The weight is expressed as a sum of the local effective Hamiltonians. If we neglect hidden layers, the effective Hamiltonian is constructed by linear combinations.

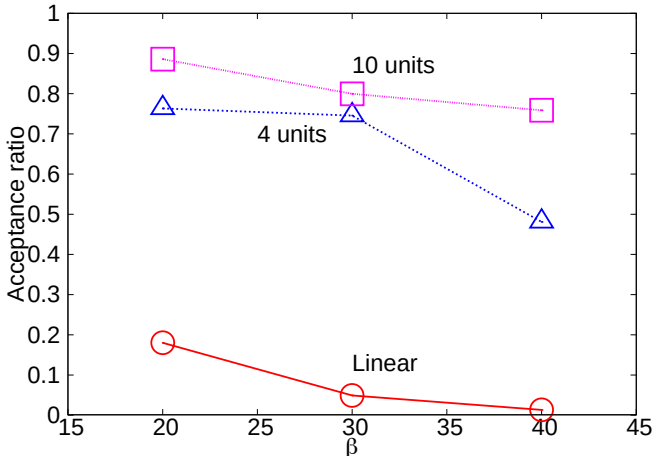

FIG. 3. (Color online) Inverse-temperature dependence of the acceptance ratio in the self-learning continuous-time interaction expansion quantum Monte Carlo simulation in the Anderson impurity model. We consider $U = 3D$, $\delta = 0.5$, $V = 1D$ and 500 SLMC steps.

Conclusion. We developed 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 ob-

tained. 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.

Acknowledgment 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 Grant Numbers 18K03552 to Y.N. and 18K05208 to M.O. and the "Topological Materials Science" (No. 18H04228) JSPS–KAKENHI on Innovative Areas to Y.N..

[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] L. Huang and L. Wang, Accelerated Monte Carlo simulations with restricted Boltzmann machines, Phys. Rev. B 95, 035105 (2017).
[11] 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).
[12] 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).
[13] Z. H. Liu, X. Y. Xu, Y. Qi, K. Sun, Z. Y. Meng, Itinerant quantum critical point with frustration and non-Fermi-liquid, arXiv:1706.10004
[14] L. Huang, Y.-f. Yang, and L. Wang, Recommender engine for continuous-time quantum Monte Carlo methods, Phys. Rev. E **95**, 031301(R) (2017).
[15] 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).
[16] A. Tanaka and A. Tomiya, Towards reduction of autocorrelation in HMC by machine learning, arXiv:1712.03893
[17] L. Zdeborova, Machine learning: New tool in the box, Nat. Phys. **13**, 420 (2017).
[18] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nat. Phys. **13**, 431 (2017).
[19] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, 602 (2017).
[20] A. Tanaka and A. Tomiya, Detection of Phase Transition via Convolutional Neural Networks, J. Phys. Soc. Jpn. **86**, 063001 (2017).
[21] E. P. L. van Nieuwenburg, Y.-H. Liu, and Sebastian D. Huber, Learning phase transitions by confusion, Nat. Phys. **13**, 435 (2017).
[22] Yi Zhang, and E.-A. Kim, Quantum Loop Topography for Machine Learning, Phys. Rev. Lett. **118**, 216401 (2017).
[23] 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).
[24] D.-L. Deng, X. Li, and S. Das Sarma, Quantum Entanglement in Neural Network States, Phys. Rev. X **7**, 021021 (2017).
[25] Y. Huang, and J. E. Moore, Neural network representation of tensor network and chiral states, arXiv:1701.06246.
[26] 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).
[27] Z. Cai, Approximating quantum many-body wavefunctions using artificial neural networks, Phys. Rev. B **97**, 035116 (2018).
[28] 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)
[29] 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).
[30] S. Iso, S. Shiba, S. Yokoo Scale-invariant Feature Extraction of Neural Network and Renormalization Group Flow, Phys. Rev. E **97**, 053304 (2018).
[31] 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).
[32] H. Shen, J. Liu, and L. Fu, Self-learning Monte Carlo with deep neural networks Phys. Rev. B 97, 205140 (2018).
[33] T.B. Blank, S.D. Brown, A.W. Calhoun, and D.J. Doren, J. Chem. Phys. **103**, 4129 (1995).
[34] 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).

[35] 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).
[36] J. Behler and M. Parrinello, Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces, Phys. Rev. Lett. **98**, 146401 (2007).
[37] 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).
[38] 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).
[39] W. Li and Y. Ando, Construction of accurate machine learning force fields for copper and silicon dioxide, arXiv:1807.02042.
[40] 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).
[41] F. F. Assaad, Continuous-time QMC Solvers for Electronic Systems in Fermionic and Bosonic Baths, in E. Pavarini, E. Koch, D. Vollhardt, and A. Lichtenstein (eds.), DMFT at 25: Infinite Dimensions: Lecture Notes of the Autumn School on Correlated Electrons 4, (Forschungszentrum Julich), ISBN 978-3-89336-953-9 (2014).
[42] 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).
[43] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv:1502.03167.
[44] The method how to treat many kinds of species in BP neural networks was proposed in Ref. [37].
[45] There is a relation between CTINT and CTAUX. The perturbation order becomes same when the parameter of the CTAUX has the relation: $K = U\beta(\delta^2 - 1/4)$[41].
[46] See Supplemental Material at [URL will be inserted by publisher for the technical details of the batch-atom normalization.]

Supplemental material

S1. 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 $\boldsymbol{c}^j = (c_0^j, \cdots, c_{m_{cut}-1}^j, d_0^j, \cdots, d_{m_{cut}-1}{}^j)^T$. Here, the coefficients are defined as

$$c_m^j = \sum_{i=1}^{N} \phi_m(\tau_{ij}), \tag{S1}$$

$$d_m^j = \sum_{i=1}^{N} s_i s_j \phi_m(\tau_{ij}), \tag{S2}$$

with the pseudo-spin index s_i. The effective Hamiltonian is defined as

$$H_{\rm eff}(\mathcal{C}) = \frac{1}{N} \sum_{j=1} h_{\rm eff}(\boldsymbol{c}^j) + f(N), \tag{S3}$$

where

$$h_{\rm eff}(\boldsymbol{c}^j) = \hat{W}_2 F(\hat{W}_1 \boldsymbol{c}^j + \boldsymbol{b}_1) + b_2, \tag{S4}$$

$$= \sum_{j_1=1}^{N_u} [\hat{W}_2]_{j_1} F\left([\boldsymbol{x}]_{j_1}^j\right) + b_2, \tag{S5}$$

where

$$[\boldsymbol{x}]_{j_1}^j \equiv \sum_{j_2=1}^{M} [\hat{W}_1]_{j_1 j_2} [\boldsymbol{c}^j]_{j_2} + [\boldsymbol{b}_1]_{j_1} \tag{S6}$$

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

$$h_{\rm eff}(\boldsymbol{c}^j) = \sum_{j_1=1}^{N_u} [\hat{W}_2]_{j_1} G(F\left([\boldsymbol{x}]_{j_1}^j\right)) + b_2, \tag{S7}$$

where

$$G(F\left([\boldsymbol{x}]_{j_1}^j\right)) = \gamma_{j_1} \frac{F\left([\boldsymbol{x}]_{j_1}^j\right) - \mu_{j_1}}{\sqrt{\sigma_{j_1}^2 + \epsilon^2}} + \beta_{j_1}. \tag{S8}$$

Here, the parameters γ_{j_1} and β_{j_1} are trainable parameters. ϵ is a small number. The batch-atom mean μ_{j_1} and variance σ_{j_1} are defined as

$$\mu_{j_1} = \frac{1}{N N_{\rm batch}} \sum_{l=1}^{N_{\rm batch}} \sum_{j=1}^{N} F\left([\boldsymbol{x}^l]_{j_1}^j\right), \tag{S9}$$

$$\sigma_{j_1}^2 = \frac{1}{N N_{\rm batch}} \sum_{l=1}^{N_{\rm batch}} \sum_{j=1}^{N} (F\left([\boldsymbol{x}^l]_{j_1}^j\right) - \mu_{j_1})^2. \tag{S10}$$

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.