Solving Schrodinger equations using a physically constrained neural network*

Kai-Fang Pu (蒲开放)[1] Han-Lin Li (李汉林)[1†] Hong-Liang Lü (吕宏亮)[2] Long-Gang Pang (庞龙刚)[3‡]

[1]College of Science, Wuhan University of Science and Technology, Wuhan 430065, China
[2]HiSilicon Research Department, Huawei Technologies Co., Ltd., Shenzhen 518000, China
[3]Key Laboratory of Quark and Lepton Physics (MOE) and Institute of Particle Physics, Central China Normal University, Wuhan 430079, China

Abstract: Deep neural networks (DNNs) and auto differentiation have been widely used in computational physics to solve variational problems. When a DNN is used to represent the wave function and solve quantum many-body problems using variational optimization, various physical constraints have to be injected into the neural network by construction to increase the data and learning efficiency. We build the unitary constraint to the variational wave function using a monotonic neural network to represent the cumulative distribution function (CDF) $F(x)=\int_{-\infty}^{x}\psi^*\psi \mathrm{d}x'$. Using this constrained neural network to represent the variational wave function, we solve Schrodinger equations using auto-differentiation and stochastic gradient descent (SGD) by minimizing the violation of the trial wave function $\psi(x)$ to the Schrodinger equation. For several classical problems in quantum mechanics, we obtain their ground state wave function and energy with very low errors. The method developed in the present paper may pave a new way for solving nuclear many-body problems in the future.

Keywords: deep neural network, auto differentiation, variational problems, the cumulative distribution function, ground state wave function

DOI: 10.1088/1674-1137/acc518

I. INTRODUCTION

The universal approximation theorem of the deep neural network (DNN) [1] makes it powerful for representing a variational function $y=f(x,\theta)$ with trainable parameters θ. In physics, this function can be used as solution of many different partial differential equations (PDEs) $\hat{L}f=0$, such as Maxwell equations in the electromagnetic field, Navier-Stokes equations in fluid dynamics, Schrodinger equations in quantum mechanics, and Einstein field equations for gravity. The traditional way to solve this problem is to use physical models. These models face great challenges in solving inverse problems with complex geometric regions and high-dimensional space. Unlike these models, the deep learning method developed in this study provides a new direction to solve these problems. As the parameters of a DNN are initialized with random numbers, the variational function $f(x,\theta)$ violates the PDEs, and the residuals $\delta=|\hat{L}f|$ are usually the optimization objectives that can be minimized to the desired precision. In this way, many physical problems [2] are naturally mapped into optimization problems [3] that can be solved using the modern deep learning libraries.

The main advantages of machine learning are that (1) it directly establishes the function mapping between input and output data, and (2) ordinary differential equations (ODEs) and PDEs can be transformed into variational problems that can be solved using optimization. Machine learning can be helpful in finding low-dimensional manifolds in a high-dimensional space, which is crucial for the quantum many-body problem, which suffers from the curse of dimensionality. The associated disadvantage is that it is at an early stage of development and its applicability to computational physics has not been fully tested.

With strong information encapsulation capability, deep learning has been proved to be a powerful tool in solving quantum many-body problems [4–8]. The most typical application is to use the DNN to represent the wave function of quantum many-body states for many-electron systems [9]. In subsequent developments, artificial neural network (ANN) applications were extended to prototypical spin lattice systems and quantum systems in

Received 1 January 2023; Accepted 17 March 2023; Published online 18 March 2023

* Supported by the National Natural Science Foundation of China (12035006, 12075098), the Natural Science Foundation of Hubei Province (2019CFB563), the Hubei Province Department of Education (D20201108), Hubei Province Department of Science and Technology (2021BLB171)

† E-mail: lihl@wust.edu.cn (Corresponding author)
‡ E-mail: lgpang@ccnu.edu.cn (Corresponding author)

a continuous space [10–12]. Recently, machine learning has been used to deal with ab-initio problems [13–15]. The Feynman path integral [16] is another method for solving quantum state problems. Modern generative models can represent a probability distribution with high computational efficiency. A Fourier-flow generative model has been proposed to simulate the Feynman propagator and generate paths for quantum systems [17]. Further, Ref. [18] proposed a Feynman path generator that can estimate the Euclidean propagator and the ground state wave function with high accuracy.

PDEs usually have boundary and/or initial conditions. In an early study, these initial and boundary conditions were built into the neural network by construction, and the training objective was to minimize the residual δ alone. This method uses hard constraints such that $f(x,\theta)$ satisfies the initial and boundary conditions automatically. It is thus quite data efficient. The recent physics informed neural network [19–21] uses soft constraints where the violations to initial and boundary conditions are also added to the training objective $L=|\hat{L}f|+\beta_1|\delta_{BC}|+\beta_2|\delta_{IC}|$.

Some variational functions should obey physical constraints. For example, in solving the Maxwell equations, the magnetic field represented by the DNN should be divergence free. To include this constraint, the paper "Linearly constrained neural network" proposes a DNN that produces a vector field $\vec{A}(x,y,z,\theta)$ whose curl $\nabla\times\vec{A}$ is divergence free [22]. It is thus also possible to construct a scalar field $\phi(x,y,z,\theta)$ whose gradients $(\partial_x\phi,\partial_y\phi,\partial_z\phi)$ are curl free. Actually, a general method has been developed to construct neural networks with linear constraints. In solving the many-body Schrodinger equations, the many fermion wave function should be anti-symmetric. FermiNet and PauliNet use the Slater determinant to construct DNNs that are anti-symmetric. [23] In DFT [24–26] and molecular dynamics [26], the local chemical environment usually has translational or rotational symmetry that is considered using a gauge equivalent neural network [27]. In the lattice gauge field theory [28], gauge equivariant normalizing flows are employed to sample field configurations [29].

In the present work, we use a monotonic neural network to represent the cumulative distribution function $\int_{-\infty}^{x}f(x')\mathrm{d}x'$, whose first order derivative is the probability density $f(x)=\psi^*(x)\psi(x)$ that gives the ground state wave function. The present paper demonstrates that a neural network with physical constraints can be used as efficient trial wave functions of Schrodinger equations. Auto-diff helps to compute the required derivatives of the trial function with respect to the input variables. In this way, optimizing the violation of the trial function to PDEs allows solving the PDEs with high accuracy. Compared to previous methods, our method does not need to calculate any numerical integrals in the whole calculation and the unitary constraint we impose on the variational wave function increases the data learning efficiency. The improved algorithm greatly reduces the amount of computation required to solve the same Schrodinger equation. These advantages make our method more suitable for dealing with many-body states, which require a huge amount of computation.

II. METHODS

The traditional variational method for quantum mechanics [30, 31] usually uses a given function with unknown parameters as a variational function, *e.g.*, $\mathrm{e}^{-\alpha r}$ with α as the unknown parameter. Different from the previous variational artificial neural network (VANN) applications [23, 32], we do not use the DNN to represent the wave function directly; instead, we use the DNN to represent the cumulative distribution function (CDF), which is the integration of the probability density function on the spatial coordinate. The training objective is thus to minimize the violation of the wave function $\psi(x)$ represented by the neural network to the Schrodinger equation. Its relationship with the wave function is expressed as follows:

$$F(x)=\int_{-\infty}^{x}\psi^*(x')\psi(x')\mathrm{d}x', \tag{1}$$

$$\psi(x)=\sqrt{\frac{\mathrm{d}F(x)}{\mathrm{d}x}}, \tag{2}$$

where $F(x)$is the CDF represented by a neural network that is monotonic by construction. We decided to constrain the weights to make the algorithm data efficient. The derivative $\mathrm{d}F/\mathrm{d}x$ is calculated using auto differentiation (auto-diff) [33], which is provided by the deep learning libraries automatically, in analytical precision. There are two advantages of using the CDF. First, the wave function extracted from the CDF automatically satisfies the normalization condition. Thus, there is no numerical integral in the whole calculation. Second, the values of the CDF are between $(0,1)$, a much smaller range than that of the PDF, making the neural network much easier to train under the same learning rate. In practice, our training epochs are far fewer than those in the previous algorithm. Moreover, because we eliminate all the integrals, each epoch requires less computation than the previous algorithms. Therefore, our algorithm can achieve higher accuracy with less computation.

We use a feed forward neural network, or simply a multi-layer perceptron, to represent the CDF. The input of the neural network is the n-dimensional spatial coordinates x, and the first layer of the DNN consists of $m=32$ hidden neurons whose values are calculated by

$h_1 = \sigma(xW_1 + b_1)$, where W_1 is the weight matrix with $n \times m$ elements and b_1 is the bias vector with m values. σ is the activation function, which provides the neural network with non-linear representation ability. To increase the representation power, the values of neurons in the first hidden layer are feed forward to the second hidden layer with similar operations $h_2 = \sigma(h_1 W_2 + b_2)$. One can stack multiple hidden layers with one output neuron in the last layer to represent the value of the CDF function. The whole neural network can thus be thought of as one variational function $F(x,\theta)$ with θ being all the trainable parameters in the neural network.

To ensure that $F(x,\theta)$ is monotonic, we add a non-negative constraint to the weights W_i of the neural network. At the same time, the activation function should also be monotonic. In principle, sigmoid, tanh, as well as leaky relu functions can all be used to construct this monotonic neural network. In practice, we use a sigmoid activation function whose derivatives are also continuous. This is important when the second order derivatives are required in auto-diff. For example, if one uses a relu activation function, the second order derivatives of the output of the neural network to the input equal 0. The last layer also uses a sigmoid activation function to ensure that the output range is $(0,1)$.

The training objective is to find the ground state energy E_0 and its corresponding wave function ψ_0 by minimizing the violation of the wave function $\psi(x)$ to the following Schrodinger equation:

$$H|\psi\rangle = E_0|\psi\rangle, \tag{3}$$

where $H = -\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf{x})$ is the Hamiltonian operator and E_0 represents its smallest eigenvalue. The loss function is thus set to be

$$L(\theta) = |(H - E_0)|\psi\rangle + |F(x_{\min})| + |F(x_{\max}) - 1|, \tag{4}$$

where θ represents all the trainable parameters in the monotonic neural network, $\nabla^2|\psi\rangle$ is computed by the neural network through auto-diff, and E_0 is another trainable parameter initialized with a constant number, 0.0. Two additional loss terms are added to take into account the boundary condition of the CDF. We use it to limit the range of values of the CDF, which ensures that the wavefunction satisfies the normalization condition. In previous VANN applications, this term was written as $<\psi|H|\psi> / <\psi|\psi>$. We replace the numerical integration of the denominator with soft constraints, which simplifies the calculation.

Before optimization, the weight values of the neural network parameters are usually initialized randomly or through the Xavier scheme [34]. In our problem, we observe that the scheme of parameter initialization has little influence on the training process and the result of variational optimization.

We attempt to eliminate all the numerical integrals in the whole calculation because, in neural network calculations, the differential is easier to calculate than the integral. To calculate the integral, we have to use numerical approximation methods such as Monte Carlo sampling, which will certainly increase the amount of computation and may affect the accuracy. In practice, we found that we had to subtract the energy term if we did not want to include the integral.To find the ground state energy and wave function, we add another loss term $e^{0.001E_0}$. The basic logic is to decrease E_0 during the optimization of the overall loss. The function form of this loss term is designed to produce proper negative gradients at different stages of training. First, the gradients should be sufficiently large to make it converge fast when E_0 is much larger than the ground state energy. Second, the gradients should be small enough to avoid interfering with the optimization of other parts of overall loss when E_0 is sufficiently close to the true value. Last but not least, this loss term must monotonously increase with E_0 throughout the definition domain. In principle, E_0 can be smaller than the analytical ground state energy; however, in that case, the residual of the Schrodinger equation increases faster than this term because of the coefficient 0.001. After trained, the value of E_0 approaches the exact value of the ground state energy.

We tested the performance of the DNN Schrodinger equation solver on three classical quantum mechanical problems. The first problem is the harmonic oscillator problem [35]. The harmonic oscillator is used to approximate, for example, the molecular vibration, lattice vibration, and radiation field vibration around a steady point. All of these problems can be regarded as many independent harmonic oscillators whose potential in the Hamiltonian can be written as

$$V = \frac{1}{2}m\omega^2 x^2, \tag{5}$$

where m is the mass of the oscillator, ω is its angular frequency, and x is its deviation from equilibrium position.

The second problem is to solve the Schrodinger equation with the Woods-Saxon potential [36], which is widely used in nuclear physics to represent the charge distributions of the nucleus, as follows:

$$V = \frac{-1}{1 + e^{\frac{|x| - R_0}{a_0}}}, \tag{6}$$

where a_0 is related to the thickness of the surface layer, in

which the potential drops from the outside to the inside of the nucleus, and R_0 is the average radius of the nucleus at which the average interaction occurs.

The third potential is an infinitely high potential well with a width of $2l$,

$$V = \begin{cases} \infty, & |x| > l \\ 0, & |x| \leq l \end{cases} \tag{7}$$

For the sake of brevity, the parameters in Hamiltonian use the following values:

$$\hbar = m = \omega = 1, \ R_0 = 6.2, \ a_0 = 0.1, \ l = 4. \tag{8}$$

Different from previous studies that solve Schrodinger equations using supervised learning, our method is close to unsupervised learning, where both E_0 and ψ_0 are learned through optimization. The input to the neural network is a list of shuffled coordinates sampled from the domain. Using these coordinates, we compute the loss functions and minimize the violation of E_0 and ψ_0 to the Schrodinger equation, as well as $e^{0.001E_0}$. In principle, we can use the Markov chain Monte Carlo (MCMC) method [37] to sample coordinates with the learned wave function or use active learning to generate coordinates that violate Schrodinger equations more with the currently learned network to speed up the training process. In practice, for these simple problems, the wave function is usually very close to the exact wave function after training the DNN for 2000 iterations. We generally train 10000 iterations with a very small learning rate for the last 1000 iterations.

We use tensorflow [38] to construct the DNN, to compute the auto-diff dF/dx as well as $\nabla^2\psi$, and to update the network parameters. We use the Adam algorithm [39], which adds the momentum mechanism and adaptive learning rate to the simple stochastic gradient descent $\theta_{n+1} = \theta_n - lr\frac{1}{m}\sum_{i=1}^{m}\frac{\partial l_i}{\partial\theta}$. The relevant parameters in the Adam algorithm are set to $\beta_1 = 0.9, \beta_2 = 0.999$, $\epsilon = 10^{-7}$. To speed up the training process, we use a learning rate scheduler to adjust the learning rate and make it vary in the $10^{-2} - 10^{-5}$ range. A large learning rate at an early stage makes the function converge faster at the beginning, and a small learning rate at a late time makes the training process smooth.

To quantify the difference between the true wave function ψ_{true} and the wave function learned by the DNN ψ_{DNN}, we introduce the partial-wave fidelity K as follows:

$$K = \frac{< \psi_{\text{true}}|\psi_{\text{DNN}} >^2}{< \psi_{\text{true}}|\psi_{\text{true}} >< \psi_{\text{DNN}}|\psi_{\text{DNN}} >}. \tag{9}$$

The closer K is to one, the closer is the result of the DNN to the exact wave function.

III. RESULTS

For the harmonic oscillator potential used in the present work, the analytical ground state energy is $0.5\hbar\omega$. After 1500 iterations of training, the ground state energy from DNN is $E_0 = 0.50038\hbar\omega$, whose relative error is within 0.06%. In the last stage of 10000 iterations, the error can be controlled below 0.002%.

The partial-wave fidelity K is approximately 0.997993 using three hidden layers with 32 units (neurons) per layer. To study the influence of the number of variational parameters on the training results, we computed K using different numbers of hidden layers and numbers of units per layer. The results are shown below.

As presented in Table 1, the highest fidelity is achieved using 4 hidden layers with 16 hidden neurons per layer, with $K = 0.9999967$. For this simple problem, the DNN achieves a very low error in fidelity even with only two hidden layers and four units per layer. The performance increases with the number of variational parameters, approaching the exact wave function. However, the performance of the DNN saturates or even decreases if there are too many variational parameters.

To further visualize the difference between the result of the DNN and the exact solution, we compare the CDF in Fig. 1.

Table 1. Fidelities of VANN results; the first row represents the number of hidden layers, and the first column represents the number of units in each layer.

$N_{\text{unit}}N_{\text{layer}}$	1	2	3	4
4	0.9995717	0.9999767	0.9999705	0.9999618
8	0.9999416	0.9999797	0.9999910	0.9999932
16	0.9999861	0.9999923	0.9999936	**0.9999967**
32	0.9999789	0.9999909	0.9999896	0.9999922
64	0.9999744	0.9999746	0.9999903	0.9999941

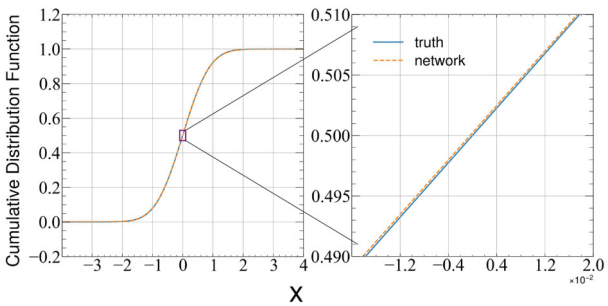

Fig. 1. (color online) Cumulative distribution equation as a function of position in the harmonic oscillator problem.

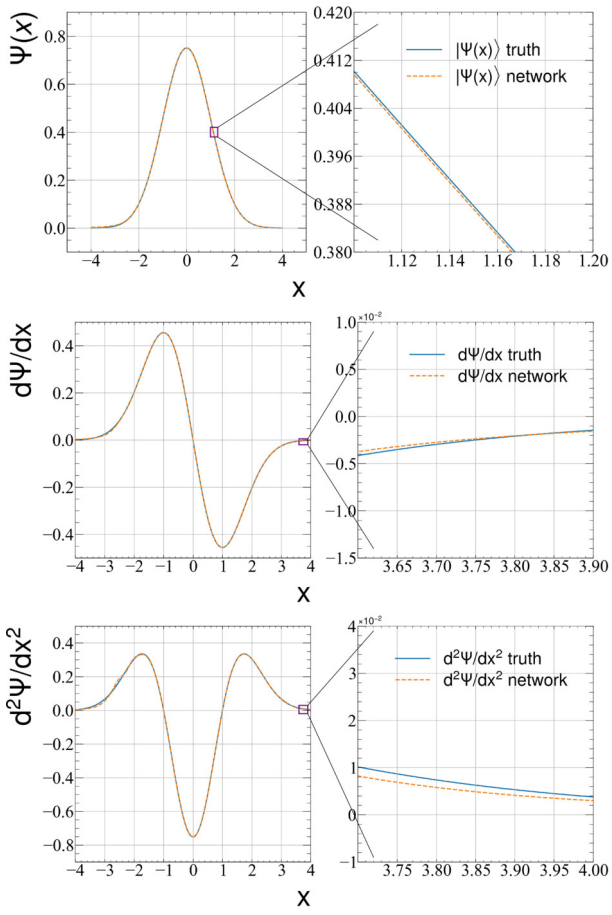

Fig. 2. (color online) Ground state wave function (top panel), first derivative (central), and second derivative (bottom) as a function of position in the harmonic oscillator problem.

Using Eq. (2), we compute $\psi(x) = \sqrt{\mathrm{d}F/\mathrm{d}x}$ and compare the wave function $\psi(x)$ and its first and second derivatives $\frac{\mathrm{d}\psi}{\mathrm{d}x}$ and $\frac{\mathrm{d}^2\psi}{\mathrm{d}x^2}$ with the analytical results. This provides a detailed comparison that shows the power of the variational function represented by the DNN.

As shown in Fig. 1, the difference between the DNN results and the ground truth is within an error range of 0.0001. The error range of the ground state wave function can be controlled within 0.0002, as shown in Fig. 2. In addition, the accuracies of the learned first and second order derivatives obtained through variational optimization are also very high, which means that this method is not only accurate, but also captures the true physics instead of finding an approximation function for the ground state wave function.

The same network is used to solve Schrodinger equations with the Woods-Saxon potential and the infinitely high potential well. No modification is made to the hyperparameters other than the potential part in the Hamiltonian. The comparisons between the learned ground state wave functions and the true values are

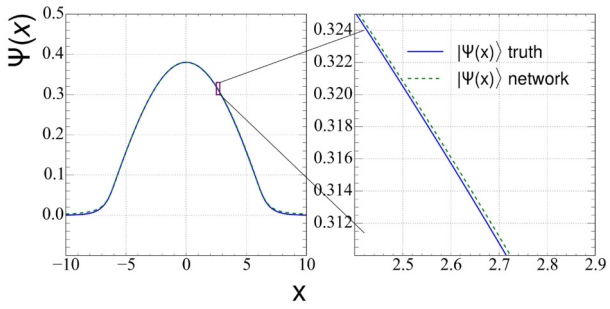

Fig. 3. (color online) Ground state wave function in the Woods-Saxon potential energy.

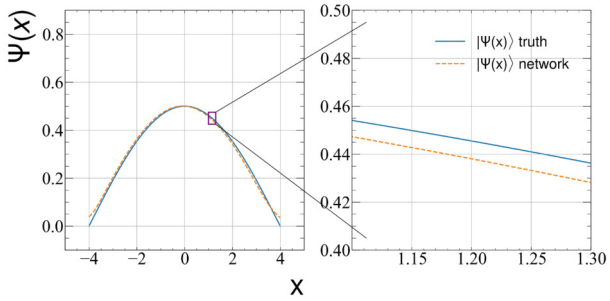

Fig. 4. (color online) Ground state wave function in the infinitely high potential well.

shown in Fig. 3 and Fig. 4. The result shows that the ground state wave functions obtained from the DNN CDF are also in excellent agreement with the exact solution. The error range of the Woods-Saxon potential's ground state wave function can be controlled within 0.0002. As shown in Fig. 4, the performance of the network on the infinitely high potential well is relatively poor, and the range of error is expanded to 0.02, which is much worse than that in the harmonic oscillator potential and Woods-Saxon potential. This is also reflected in the calculation of the ground state energy and partial-wave fidelity K. The ground state energy calculated by the DNN for the Woods-Saxon potential problem is –0.97382, also within an error of 0.002% to the exact result –0.97385, and $K = 0.999964$, similar to that for the harmonic oscillator potential. However, for the potential well problem, the DNN $E_0 = 0.07787$, whose relative error to the accurate result of 0.07710 is approximately 1%, and $K = 0.9977475$, also less than the average level of the first two potentials. It reminds us that this DNN might not perform good at dealing with potentials with discontinuities, like the infinitely high potential well, whose potential energy discontinuously changes from zero to infinity at the boundary. We think that this is due to the fact that the soft constraint cannot handle the infinite potential energy at the boundary well, so the probability density at the boundary is not 0.

IV. CONCLUSIONS

In the present study, we used a physics-based neural network to solve Schrodinger equations numerically. We designed a monotonic neural network to represent the CDF of the ground state wave function. In this way, the wave function represented by the DNN satisfies the normalization condition by design. The variational optimization is reduced to an optimization problem by minimizing the violation of the trial wave function and trial ground state energy E_0 to Schrodinger equations. The method is used to solve Schrodinger equations with three different potentials, the harmonic oscillator, the Woods-Saxon potential, and the infinitely high potential well, all with a small relative error.

Compared to traditional variational methods in solving quantum mechanical problems, the trial wave function represented by the DNN does not have fixed function forms before training. The training objective is different from the traditional $E_0 = \frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle}$, where numerical integration is required for both the numerator and denominator. In our case, the objective is to minimize the violation to the Schrodinger equation on sampled spatial coordinates. As the neural network is constrained, the trial wave function is normalized by construction. Our method is also different from the previous Schrodinger equation solver using supervised learning, where ground state energies from numerical solutions are needed to train the neural network. In another DNN Schrodinger solver [30, 31], the initial values of the network parameters greatly affect the optimization results. To avoid strong fluctuations, they provide a trial wave function whose form is close to the exact solution. The disadvantage of the previous algorithms is that they can only solve problems in which the form of the exact solution of the equation is known. Our algorithm can directly ignore the pretraining process, so we do not need to know any information of the exact solution before training. This is more universal and provides the possibility to solve problems that have never been dealt with before. In addition, we observe that our DNN can approximate the ground state wave function with fewer trainable parameters. Moreover, the physical constraints constructed in the neural network make the current method quite data efficient. Thus, we can achieve higher accuracy with less computation.

The current method can be improved in several ways. First, the CDF works for wave functions in high dimensional space as long as the n-dim spatial coordinates are flattened. Second, the spatial coordinates used for training can be sampled using the learned wave function or through active learning, to increase the training efficiency. Third, the anti-symmetric constraints of the wave function should be considered for many fermion systems. Although further efforts have to be done to improve the current method, it shows good properties in solving classical quantum mechanical problems. The next step is to solve the ground state energy and wave functions of the deuteron. It also paves a new way in solving many nucleon problems.

ACKNOWLEDGMENTS

LG Pang and KF Pu acknowledge the support provided by Huawei Technologies Co., Ltd. The contributions of Dr. Hong-Liang Lü are non-Huawei achievements. The computations were performed at the NSC3 super cluster at CCNU and High-Performance Computing Center of Wuhan University of Science and Technology.

References

[1] G. V. Cybenko, Mathematics of Control, Signals and Systems **2**, 303 (1989)
[2] A. Boehnlein *et al.*, Rev. Mod. Phys. **94**, 031003 (2022)
[3] D. Saad, American Scientist **92**, 578 (2004)
[4] P. Mehta, M. Bukov, C.-H. Wang *et al.*, Physics reports **810**, 1 (2019)
[5] E. M. Nordhagen, J. M. Kim, B. Fore *et al.*, arXiv: 2210.00365
[6] B. R. Barrett, P. Navrátil, and J. P. Vary, Progress in Particle and Nuclear Physics **69**, 131 (2013)
[7] G. Torlai, G. Mazzola, J. Carrasquilla *et al.*, Nature Physics **14**, 447 (2018)
[8] C. Adams, G. Carleo, A. Lovato *et al.*, Phys. Rev. Lett. **127**, 022502 (2021)
[9] D. Pfau, J. S. Spencer, A. G. D. G. Matthews, and W. M. C. Foulkes, Phys. Rev. Res. **2**, 033429 (2020)
[10] M. Ruggeri, S. Moroni, and M. Holzmann, Phys. Rev. Lett. **120**, 205302 (2018)
[11] J. Han, L. Zhang, and E. Weinan, Journal of Computational Physics **399**, 108929 (2019)
[12] S. Shi, K. Zhou, J. Zhao, S. Mukherjee, and P. Zhuang, Phys. Rev. D **105**, 014017 (2022)
[13] K. Choo, A. Mezzacapo, and G. Carleo, Nature communications **11**, 2368 (2020)
[14] M. Scherbela, R. Reisenhofer, L. Gerard *et al.*, Nature Computational Science **2**, 331 (2022)
[15] Y. Yang and P. Zhao, arXiv: 2211.13998
[16] R. P. Feynman, Rev. Mod. Phys. **20**, 367 (1948)
[17] S. Chen, O. Savchuk, S. Zheng *et al.*, Phys. Rev. D **107**, 056001 (2023)
[18] Y. Che, C. Gneiting, and F. Nori, Phys. Rev. B **105**, 214205 (2022)
[19] M. Raissi, P. Perdikaris, and G. E. Karniadakis, arXiv: 1711.10561
[20] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Journal of Computational physics **378**, 686 (2019)
[21] E. Haghighat, M. Raissi, A. Moure *et al.*, Computer Methods in Applied Mechanics and Engineering **379**, 113741 (2021)
[22] J. Hendriks, C. Jidling, A. Wills *et al.*, arXiv: 2002.01600

[23] J. Hermann, Z. Schätzle, and F. Nóe, Nature Chemistry **12**, 891 (2020)
[24] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964)
[25] W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965)
[26] M. S. Badar, S. Shamsi, J. Ahmed *et al*., *Molecular dynamics simulations: concept, methods, and applications, in Transdisciplinarity* (Springer, 2022), p. 131
[27] D. Luo, G. Carleo, B. K. Clark *et al*., Phys. Rev. Lett. **127**, 276402 (2021)
[28] H. J. Rothe, *Lattice gauge theories: an introduction* (Singapore: World Scientific Publishing Company, 2012), p. 628
[29] R. Abbott, M. S. Albergo, A. Botev *et al*., arXiv: 2208.03832
[30] J. Keeble and A. Rios, Phys. Lett. B **809**, 135743 (2020)
[31] H. Saito, Journal of the Physical Society of Japan **87**, 074002 (2018)
[32] C. Giuseppe and M. Troyer, Science **355**, 602 (2017)
[33] A. Paszke, S. Gross, S. Chintala *et al*., Automatic differentiation in pytorch (2017)
[34] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (JMLR Workshop and Conference Proceedings, 2010), p. 249
[35] I. Senitzky, Phys. Rev. **124**, 642 (1961)
[36] M. Capak and B. Gönül, Modern Physics Letters A **31**, 1650134 (2016)
[37] R. L. Karandikar, Sadhana **31**, 81 (2006)
[38] M. Abadi, A. Agarwal, P. Barham *et al*., arXiv: 1603.04467
[39] D. P. Kingma and J. Ba, arXiv: 1412.6980