


Neural-network-based multistate solver for a static Schrödinger equation

Hong Li

*Department of Computer Science, Wenzhou University, Wenzhou 325035, China
and Department of Physics and Astronomy, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*

Qilong Zhai

School of Mathematics, Jilin University, Changchun, Jilin 130012, China

Jeff Z. Y. Chen 

Department of Physics and Astronomy, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1



(Received 29 September 2020; accepted 12 February 2021; published 8 March 2021)

Solving a multivariable static Schrödinger equation for a quantum system, to produce multiple excited-state energy eigenvalues and wave functions, is one of the basic tasks in mathematical and computational physics. Here we propose a neural-network-based solver, which enables us to cover the high-dimensional variable space for this purpose. The efficiency of the solver is analyzed by examples aimed at demonstrating the concept and various aspects of the task: the simultaneous finding of multiple excited states of lowest energies, the computation of energy-degenerate states with orthogonalized wave functions, the scalability to handle a multivariable problem, and the self-consistent determination and automatic adjustment of the imbedded Monte Carlo procedure. The solver adheres to the computational techniques developed in machine learning and is vastly different from traditional numerical methods.

DOI: [10.1103/PhysRevA.103.032405](https://doi.org/10.1103/PhysRevA.103.032405)

I. INTRODUCTION

The expansion of machine-learning research [1–5] into computational physics has created an arena in which innovative computational methods have started to emerge. The ability of an artificial neural network (NN) to perform a complex numerical task opens the door to numerical maneuvers on which traditional methods are unable to perform. For example, solving differential equations can now be approached by using machine-learning algorithms designed to train a deep neural network (see, e.g., [6–22]). In another example, in statistical physics, the use of artificial neural networks has also inspired fundamentally different ways to calculate the partition function [23–25] and to conduct Monte Carlo (MC) simulations [26–31]. Solving the Schrödinger equation, the topic covered here, can now be performed with neural network approximations [32–39]. In this paper, we propose a universal solver for obtaining the eigenvalues and eigenfunctions of the static Schrödinger equation without the *a priori* knowledge of the actual or guessed solutions, when the potential energy is given for a D -dimensional problem. To overcome the difficulty in efficient sampling of the multiple-variable space, importance Monte Carlo sampling is incorporated into the algorithm, taking advantage of the continuous wave-function representation by a neural network.

The importance of the Schrödinger equation in describing a quantum-mechanical system needs no introduction. A typical

eigenproblem is to solve the differential equation,

$$\hat{H}\Psi_n(\mathbf{r}) = E_n\Psi_n(\mathbf{r}), \quad (1)$$

where \hat{H} is the Hamiltonian operator acting on the wave function $\Psi_n(\mathbf{r})$, which is a function of a D -dimensional variable \mathbf{r} . The dimensionality D depends on the problem at hand, and could actually be, for example, the dimensionality of the variable space of two coherent particles moving in three-dimensional space, which makes $D = 6$. The eigenvalues, E_n , once found, are arranged from the ground-state energy E_0 and up, where the symbolic n represents a set of quantum numbers in a multidimensional problem. A large body of literature has documented the anatomy of traditional numerical methods. To deal with correlated many-body problems, a well-developed field in computational physics is quantum Monte Carlo (QMC) methods [40,41].

In broad strokes, the numerical approach proposed in this paper shares the same general idea of the classical variational method. The eigenvalues are expressed as $E_n = \int \Psi_n^* \hat{H} \Psi_n d\mathbf{r} / \int \Psi_n^* \Psi_n d\mathbf{r}$. Known trial functions, which contain a number of variational parameters, are used to replace the unknown wave functions Ψ_n . The integrals in the above are then carried out. The resultant eigenvalues E_n become functions of these variational parameters. Minimization of the eigenvalues leads to determination of the variational parameters, and hence produces the approximation of the wave functions and eigenvalues. Special care must be taken to ensure that the wave functions are orthogonal to each other. As the number of variational parameters increases, the exact solution is numerically approached.

*jeffchen@uwaterloo.ca

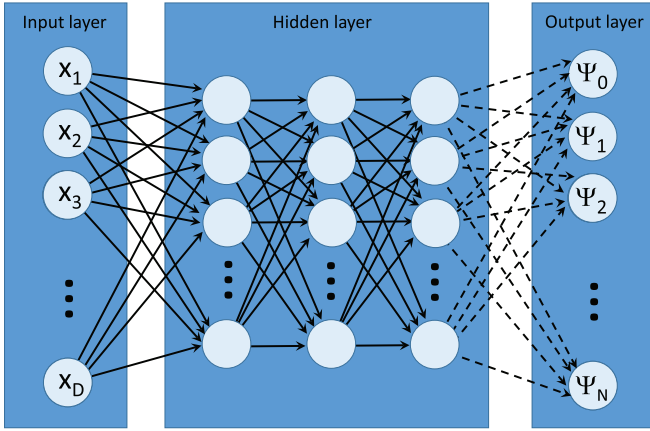


FIG. 1. Schematics of the deep NN used in the current paper. An input layer contains nodes for the spatial variables $x_1, x_2, x_3, \dots, x_D$, which are fully connected to the first hidden layer. An output layer gives the $N + 1$ wave functions $\Psi_0, \Psi_1, \Psi_2, \dots, \Psi_N$. The solid black arrow represents a typical sigmoid function and the dashed line represents a weighted linear mixing function; all are defined in Sec. II B. The number of hidden layers is L (depth) and the number of nodes per hidden layer is N_h (width).

In detail, there are three basic components in the artificial NN solver presented here.

(a) The ability of a deep NN to model complex functions is exploited to analytically represent the trial functions $\Psi_n(\mathbf{r}; \Theta)$, which contain a large number of network parameters (i.e., variational parameters), symbolically represented by the set Θ . In principle, as the network parameter space expands, any functions can be accurately approximated by the nonlinear accumulation of the intrinsic NN functions (mostly sigmoid functions), which contain these network parameters [42–44]. This is conceptually illustrated in Fig. 1 and explained in Sec. II B. In this paper a single deep NN is used for all different eigenfunctions of interest. This can be contrasted with specially designed wave functions, e.g., the Bijl-Dingle-Jastrow trial function, used in a typical QMC approach [45].

(b) The second component is minimization of a target function, which contains the sum of eigenvalues, with respect to the network parameters, Θ . A number of constraints, for example, those used to make all eigenfunctions orthogonal, are considered together at this level. This step, known in machine learning as a supervised learning session, is used here simply to minimize the eigenvalues. Once the minimization is done, a set of optimized Θ is obtained, which produces the approximate eigenvalues and eigenfunctions for the Schrödinger equation. The machine (i.e., the deep NN) then makes a “deep learning” [46] of (i.e., finds) the approximate solutions. The structure of the differential equation is embedded into the operator \hat{H} . No initial guess of the wave functions or eigenvalues is required. Hence, the task of solving the Schrödinger equation is mapped onto a nonlinear statistical regression problem of reduced complexity [47]. The form of the target function and the auxiliary conditions are described in Secs. II A and II C.

(c) The third component is to design an efficient algorithm to carry out the integrals over the variable space \mathbf{r} , required in

the target function. When the dimensionality D of the \mathbf{r} space becomes large, dividing the \mathbf{r} space into computational grids requires a computational resource (storage space and computational time) that grows exponentially in D , producing a crisis known as the curse of dimensionality [19,20,48]. To combat this deficiency, the main idea of QMC is followed here. A large but finite number of MC sampling points are adopted in the \mathbf{r} space. In concert with recent NN-based solvers [37–39], the importance-sampling technique is introduced to convert an integration into a summation to find MC means. Our implementation has a single MC weight function that is written in terms of the wave functions themselves, here for multiple states. This crucial step bypasses the need to divide the \mathbf{r} space into computational grids. The self-consistent procedure introduced here determines the weight function and the wave functions simultaneously, and controls the distribution of the sampling coordinate points in the variable space \mathbf{r} automatically. No manual decision making on the range of the variable space to be sampled is required. The implementation of the *analytic* NN representation of the wave functions is critical to the success of this component. As the Monte Carlo sampling points move in the \mathbf{r} space, the differential operations such as $\nabla^2 \Psi_n(\mathbf{r}; \Theta)$ are carried out analytically, which do not require finite-difference schemes typically used in, e.g., the finite-difference method. Details can be found in Secs. II D and II E.

Using NN to solve the Schrödinger equation is not a new topic. The representation of wave functions analytically by NN [(a) above] has been proposed in recent years and has become a common feature in NN-based solvers [32–39,49]. Some recent progress is summarized in Table I, where the main features of each work are compared. The original paper by Lagaris *et al.* recommended treating $||\hat{H}\Psi_n(\mathbf{r}) - E_n\Psi_n(\mathbf{r})||^2$ (i.e., the entire differential equation) as the main component of the cost function [32], for an excited state n . The Schrödinger equation is solved by driving the module to zero. This is similar to the treatment taken in solving other types of differential equations [7]. Similar cost functions are used in [33–35] as well. Here our approach differs from this idea by directly treating a summed collection of $E_n = \int \Psi_n^* \hat{H} \Psi_n d\mathbf{r} / \int \Psi_n^* \Psi_n d\mathbf{r}$ as the cost function, in a spirit closer to the variational method. The use of a variational E_n itself as the target function is an idea that has recently surfaced in the literature, either for the ground state only [37–39] or for an individual excited state (see [50], though it uses Slater-Jastrow parametrization instead of NN.) Here we propose to consider the sum of all E_n as a single target function, up to the $n = N$ excited level.

Beyond the ground state, one challenge is to push for a calculation of the eigenproblems of the N excited states of low-energy values. The schemes used in finding the excited states have been quite different. Lagaris *et al.* suggested to construct an excited-state wave function, $\Psi_n(\mathbf{r})$, with embedded orthogonality to previously calculated wave functions, and then to take the target function $||\hat{H}\Psi_n(\mathbf{r}) - E_n\Psi_n(\mathbf{r})||^2$ for the calculation of state n . Each $\Psi_n(\mathbf{r})$ requires a new NN separately from the previous ones [32]. Shirvany *et al.* used a numerically expensive method of moving E incrementally to minimize the target function, in an effort to numerically find the solution of an excited state; each move would require

TABLE I. A summary of representative neural network (NN) approaches in solving the Schrödinger equation, either for a given potential or for an *ab initio* electron Hamiltonian (*abH*). In some, coupled harmonic oscillators (HOs) were used as examples. The column with a “Spin” heading refers to whether the wave functions are constructed to follow antisymmetry properties of half-integer-spin particles (fermions). Most work produces energy levels which are then compared to known results (CKR). For the ground and excited states, wave functions are required to be orthogonal (OR) to each other.

First author	Main target function	Sampling technique	Spin	Excited states	Example potentials	Scale-up analysis	Error analysis	Network type
Lagaris [32]	$\int d\mathbf{r}[(\hat{H} - E_n)\Psi_n]^2$; Fixed grids Ψ_n are OR by construction	Fixed grids	No	Up to $N = 2$	Morse; three-dimensional anharmonic oscillators		CKR	Separate, three-layer forward NNs for each n
Nakanishi [33]	$\int d\mathbf{r}[(\hat{H} - E)\Psi_n]^2$; no OR required	Fixed grids	No	Up to $N = 13$	One-dimensional HO; double-well		CKR	Three-layer forward NNs for microgenerative algorithm
Shirvany [35]	$\int d\mathbf{r}[(\hat{H} - E)\Psi_n]^2$; no OR required	Fixed grids	No	Up to $N = 5$	Square-well		CKR	Three-layer NNs for moving E
Mills [36]	Mapping of solutions	Fixed grids	No	Ground state only	Two-dimensional HO, etc.		CKR	Convolution NN
Teng [37]	$\int d\mathbf{r}\langle\Psi_0 \hat{H} \Psi_0\rangle$	MC weight Ψ_0^2	No	Ground state only	HOs in electric field		CKR	Three-layer NN (radial basis)
Pfau [38]	$\int d\mathbf{r}\langle\Psi_0 \hat{H} \Psi_0\rangle$	MC weight $\sim\Psi_0^2$	Yes	Ground state only	<i>abH</i> for LiH, etc.		CKR	Deep NN (FermiNet)
Hermann [39]	$\int d\mathbf{r}\langle\Psi_0 \hat{H} \Psi_0\rangle$	MC weight Ψ_0^2	Yes	Ground state only	<i>abH</i> for H10, etc.	Scaling on e numbers	CKR	Deep NN (PauliNet)
This paper	$\int d\mathbf{r}[\sum_n \langle\Psi_n \hat{H} \Psi_n\rangle + \sum_{nm} \langle\Psi_n \Psi_m\rangle^2]$	Single MC weight $\sum_n \Psi_n^2$	No	Up to $N = 31$	D -coupled HOs	Scalings on both D and N	Self-assessed; CKR	Single five-layer deep NN for all n

a new NN solution [35]. Here we suggest to use a single cost function, augmented by the summed constraints to ensure orthogonality of the wave functions [(b) above], all within the same NN. This approach avoids separate parametrization and new optimization for each new excited state [50]. The advantage of our approach is the systematic numerical discovery of the low-energy states up to a truncated level, avoiding accidental trapping in a high-energy eigensolution. This is demonstrated in Sec. III by examples where up to 32 energy levels are simultaneously found in an ascending order. A study of the computational cost for finding high excited states is also documented in that section.

A recent perspective paper captures the progress in using machine learning to solve the Schrödinger equation [51]. Most previous studies used one- or two-dimensional examples, for which the variable space \mathbf{r} can be efficiently handled by finite-element or finite-difference methods. A common feature in QMC approaches is to incorporate the weighted MC method into the algorithm, avoiding variable-space division that would otherwise encounter the difficulty of the curse of dimensionality. Recently developed NN schemes to handle high-dimensional problems contain MC method in the NN design [37–39]. The particular version of the MC weight that we use here is the algebraic sum of *all* squared wave functions of the ground and excited states, which enables the self-consistent determination of the entire eigenproblem under one *single* NN design and one *single* MC weight. This can be contrasted to the usage of separate parametrization and separate MC weights suggested in [50]. The idea of using the squared wave functions coincides with the use of squared projection of the ground-state wave function in [37] and squared

ground-state wave functions in [38,39], where, however, their main concern is the ground state.

Two important issues are also addressed here. The first is that with the use of the MC sampling and the NN learning procedure one needs to establish a convergence criterion to self-assess the errors of the numerical results, and hence determine when to truncate the calculation. This is particularly important when the NN solver is used for an unknown problem. A general procedure of the convergence criterion is recommended in Sec. III B.

The second is, when a unique convergence criterion is established, how the computational cost scales as a function of the total number (N) of the excited states calculated and the dimension (D) of the variable space. Intertwined with the computational complexity of the problem is the dependence on the network size. The scalability of the current approach to high N and high D is explored in Sec. VI, up to $N = 31$ and $D = 128$. Unfortunately, previously these two issues were seldom addressed carefully for NN-based solvers of the Schrödinger equation, as far as we know.

II. BASIC FORMALISM

A. NN representation of the eigenvalue problem

The general mathematical problem is to solve the partial differential equation in (1), where, in Cartesian coordinates, the position vector is expressed by its component $\mathbf{r} = (x_1, x_2, \dots, x_D)$. The Hamiltonian operator

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf{r}), \quad (2)$$

where ∇^2 is a Laplacian operator

$$\nabla^2 = \frac{\partial^2}{\partial x_i^2} \quad (3)$$

and $V(\mathbf{r})$ is the potential energy (known from a given problem), a scalar function of \mathbf{r} . For simplicity, we use the same m for the overall coefficient of ∇^2 ; the above form is for one particle of mass m moving in a D -dimensional space, strictly. The procedure described below can be easily adopted for solving the Schrödinger equation of multiple, interacting particles where the coefficients of the second-order partial derivatives in ∇^2 can differ from each other, generally. The subscript n in (1) represents a set of quantum numbers that describe the quantum states. The question is whether one can solve the above eigenproblem to obtain the eigenfunctions $\Psi_n(\mathbf{r})$ and eigenvalues E_n , arranged such that $E_0 < E_1 \leq \dots \leq E_N$. Here N specifies the maximum number of excited states of interest (the complication of multiple quantum numbers and energy degeneracy is discussed in Sec. IV).

To represent the unknown functions $\Psi_n(\mathbf{r})$ with the vector \mathbf{r} as the input, a feedforward NN is used, which has D nodes in the first layer and $(N+1)$ nodes in the final layer as the output for $\Psi_0, \Psi_1, \dots, \Psi_N$ (see Fig. 1). The deep NN is made of a number of layers so that it can model the complicity of $\Psi_n(\mathbf{r})$, which is needed for modeling the wave functions of the excited states; the deep NN contains a network-parameter set Θ ; different values of the set Θ give rise to different outputting Ψ_n , hence we write the output $\Psi_n(\mathbf{r}; \Theta)$. The set Θ needs to be adjusted during the numerical procedure to arrive at appropriate values for a specific physical problem; once this is done, an approximation for the eigenproblem is then numerically found.

How do we numerically obtain the set Θ ? Instead of directly solving the differential equation (1), we minimize the total cost function

$$J(\Theta) = \sum_{n=0}^N E_n(\Theta) + P(\Theta) + R(\Theta) \quad (4)$$

with respect to Θ to obtain a solution to the problem. The first term is the sum of all eigenvalues to be estimated, now written in terms of integrals:

$$E_n(\Theta) = \frac{\langle \Psi_n | \hat{H} | \Psi_n \rangle}{\langle \Psi_n | \Psi_n \rangle}, \quad (5)$$

where the standard bra-ket notation is used for integration over \mathbf{r} of the wave functions $\Psi_n(\mathbf{r}; \Theta)$. The second term

$$P(\Theta) = \beta \sum_{n=0}^N A_n(\Theta) + \gamma \sum_{n=1}^N \sum_{m=0}^{n-1} B_{nm}(\Theta) \quad (6)$$

contains all squared normalization conditions (A_n) that the wave functions must follow and all squared orthogonal conditions (B_{nm}) between the wave functions of eigenstates n and m where $n \neq m$. Expressions for A_n and B_{nm} are explicitly laid out in Sec. II C. The third term in (4) represents the penalty used for network parameter regularization (known as the L2 regularization in machine learning) to be specified in the next section. The penalty coefficients β and γ are selected by trial and error, to achieve the desired tolerance level of

each condition. The basic assumption here is that if $J(\Theta)$ is minimized all orthogonal conditions are then satisfied and J settles for the sum of the lowest $N+1$ eigenvalues. Computationally, the minimization procedure is well prescribed in machine learning; the back-propagation method [52], for example, is used in this paper, following a set of subroutines in TENSORFLOW [53].

In brief summary, using the above representation, we turn the eigenproblem into a conventional machine-learning problem where the eigenfunctions and eigenvalues are all expressed in terms of the network parameters. The minimization of the cost function can then be handled by existing numerical algorithms. The phrase “machine learning” normally refers to a set of features to be learned by a designated NN. We stress that no *a priori* knowledge of the eigenproblem is needed here. What the machine is actually learning (supervised by the above principles) is that the cost function is minimized when the sampled points are in the variable space \mathbf{r} . In fact, this general numerical procedure is *unsupervised*—no solution features are given at the first place.

B. Network design

In this section, we define the function $\Psi_n(\mathbf{r}; \Theta)$ based on a feedforward deep NN sketched in Fig. 1. The main function is to read the data of the spatial variable \mathbf{r} through the input layer, process the information in the hidden layers, and then generate the wave function Ψ_n in the output layer. The arrows (“edges”) represent function calls that calculate a value for the next node based on the values on the previous nodes. The function call contains its own network parameters, which are commonly referred to as the weights and biases. By varying network parameters the final output Ψ is consequently affected by these parameters. The deep NN is characterized by its depth L (the number of hidden layers) and width N_h (the number of nodes in each hidden layer).

To adhere to the common usage in machine learning [5], the sigmoid function is used here as the activation function. Assume for now that the previous layer of nodes has the values z_1, z_2, z_3, \dots (or the \mathbf{r} vector if the previous layer is the input layer). The value of node j in the next layer, z'_j , is evaluated by a sigmoid function:

$$z'_j = \left[1 + \exp \left(\sum_i w_{ji} z_i + b_j \right) \right]^{-1}, \quad (7)$$

where the matrix element w_{ji} and vector element b_j are the network parameters. The total number of these for a hidden layer is $D \times N_h + N_h$ on the first hidden layer and $N_h^2 + N_h$ thereafter. In the sketch presented in Fig. 1, a total of $(D+1)N_h + (N_h^2 + N_h)(L-1)$ parameters are needed when the final hidden layer is reached.

The final dashed edges are weighted mixing functions that take the output from the final hidden nodes, assumed to have values $z_1, z_2, z_3, \dots, z_{N_h}$:

$$\Psi_n = \sum_{j=1}^{N_h} a_{nj} \left[\exp \left(- \sum_{i=1}^D v_{ji}^2 x_i^2 \right) \right] z_j. \quad (8)$$

The network parameters are the mixing coefficients a_{nj} [with a total of $(N + 1) \times N_h$ elements] and weights v_{ji} [with a total of $N_h \times D$ elements]. The exact form of the weights in the square brackets is designed to handle wave functions that decay in far variable regions, assumed here to be Gaussian. The function form in the square bracket can be modified for adoption to a specific potential function $V(\mathbf{r})$. Our simple functional form is different from the much more complicated radial-basis functions used in [37].

In brief summary, the wave functions are represented by building up multilayers of sigmoid functions. At the final stage, weights are added to ensure that the wave functions decay in a bounded potential problem. In total, the parameter set Θ includes the matrix $[w_{ji}]$ and vector $[b_j]$ at different hidden layers, and the matrices $[v_{ji}]$ and $[a_{nj}]$ leading to the output layer.

The large number of network parameters may overwhelm a typical machine-learning session. To prevent overfitting, the L2 regularization method [5,54] is used by adding to the cost function J in (4) a penalty term:

$$R(\Theta) = \frac{\Lambda}{2} \sum_j \sum_i w_{ji}^2. \quad (9)$$

All computations conducted in this paper are produced by letting $\Lambda = 0.001$, for moderate control of the approximate magnitudes of w_{ji} . No dropouts [55] are used.

C. Orthonormal conditions

Having built the trial wave functions by an NN, we are now ready to express other terms required in the cost function, (4). Any wave function, as the solution to the Schrödinger equation, can have an overall, undetermined coefficient which is fixed by the normalization function. This is reflected in the linear mixing of the hidden layer output to the wave functions in (8), with an overall undetermined coefficient for each n . To remedy this uncertainty, we let

$$A_n = \left(\sum_{j=1}^{N_h} a_{nj}^2 - 1 \right)^2, \quad (10)$$

and place A_n in the penalty function (6) to enforce a unique solution for the n th wave function. The enforcement does not directly yield the conventional integrated normalization condition, which is not needed in the actual computation. For example, the wave functions are directly renormalized through the denominator of (5) to ensure the correct evaluation of all E_n .

The orthogonality between the wave functions of the n th and m th states is dealt with by defining

$$B_{nm} = \langle \Psi_n | \Psi_m \rangle^2, \quad (n > m). \quad (11)$$

The expectation is that once the penalty function, (6), is minimized to yield zero, the orthogonality conditions are satisfied by having all $B_{nm} \rightarrow 0$, numerically. Note that the computation of B_{nm} does not require the normalization of the wave functions, Ψ_n and Ψ_m .

The above formalism, Eqs. (4)–(11), is now written in machine-learning languages which can be used as the basic templet for coding. The Laplacian derivative in $\hat{H}|\Psi_n\rangle$, in

particular, can be taken analytically on the \mathbf{r} dependence with the implementation of TENSORFLOW functional calls [53]. The evaluation of J is done by sampling through the \mathbf{r} space, with selected points that enable the calculation of the integrals needed in (5) and (11). The back propagation technique, coupled with the deepest descent minimization scheme, can then be used to drive the Θ set to the desired solution.

D. Importance MC sampling

Both (5) and (11) require the evaluation of integrals of the type

$$\langle \Psi_n | \hat{O} | \Psi_m \rangle = \int dx_1 dx_2 dx_3 \dots dx_D \Psi_n^*(\mathbf{r}) \hat{O} \Psi_m(\mathbf{r}) \quad (12)$$

in the variable space. The operator \hat{O} can be \hat{H} or can just be an identity operator. One could divide the interested variable space (evenly or preweighted), covering the main variation of the integrand by K^D sampling points. The number of grid points in each coordinate component of \mathbf{r} is K and is usually large. Then, the computational time would scale as K^D to some power. At a low D , this is not a problem. At a high D , however, the computational time explodes exponentially in D , which forms a roadblock in treating high- D problems; this undesirable behavior is known as the curse of dimensionality. The computation on an “epoch,” a step taken to minimize J with respect to all Θ , requires the calculation of these integrals in D dimensions in order to produce the next improved Θ set.

The MC method overcomes this difficulty. We replace the integration required in (12) by importance sampling, in reference to a weight function $W(\mathbf{r})$. If we can design a weight $W(\mathbf{r})$ in the variable space where the wave functions are most significant, then

$$\langle \Psi_n | \hat{O} | \Psi_m \rangle = \frac{1}{S} \sum_{l=1}^S \Psi_n^*(\mathbf{r}_l) \hat{O} \Psi_m(\mathbf{r}_l) / W(\mathbf{r}_l), \quad (13)$$

where S is the total number of sampling points. The set $\{\mathbf{r}_l\}$, where $l = 1, 2, \dots, S$, is selected according the weight function $W(\mathbf{r})$. From one epoch to another epoch, a different $\{\mathbf{r}_l\}$ set can be created according to the Metropolis rule to move $\{\mathbf{r}_l\}$ by a random increment following the distribution of the weight function [56,57].

What $W(\mathbf{r})$ should one use? The wave functions of the bounded states themselves can be used for this purpose. Here we propose to use

$$W(\mathbf{r}) = \frac{1}{N} \sum_{n=0}^N |\Psi_n(\mathbf{r})|^2. \quad (14)$$

The summation takes into account the peaks of the wave functions in the excited states considered. Of course, this is not the only choice of the MC weight $W(\mathbf{r})$.

E. Summary: Self-consistent numerical procedure

This section contains various numerical recipes that are summarized here. For a given $V(\mathbf{r})$, the self-consistent numerical procedure consists of the following steps.

(1) An initial, reasonable, and simple guess of $W(\mathbf{r})$ is made and S sampling data points are created in D -dimensional

space. These points could follow the distribution of $W(\mathbf{r})$, or could be, at this stage, randomly distributed in a reasonable range. An initial guess of the Θ set is also made.

(2) The integrals in (5) and (11) are evaluated according to the MC sum in (13). Hence the cost function in (4) is evaluated, together with the gradient $\nabla_{\Theta}J$. An improved Θ set is obtained from

$$\Theta - \eta \nabla_{\Theta}J \rightarrow \Theta. \quad (15)$$

The parameter η is the so-called learning rate and is adjusted according to the ADAM algorithm [58].

(3) Based on the improved Θ and through the deep NN, an updated set of functions $\Psi_n(\mathbf{r})$ is available. In terms of the definition in (14), a new $W(\mathbf{r})$ is also available.

(4) The sampled points $\{\mathbf{r}_l\}$ are moved by adding to the existing values small, random displacements, to form the set $\{\mathbf{r}'_l\}$. The Metropolis rule is used here, in reference to the updated probability function $W(\mathbf{r})$, to determine whether or not the new set $\{\mathbf{r}'_l\}$ is accepted. The overall magnitude of the displacement is adjusted in later epochs to yield a 50% acceptance rate, approximately.

(5) We go back to step 2, now taking the updated Θ and $\{\mathbf{r}_l\}$ to calculate various quantities.

Steps 2–5 constitute a single learning epoch. The self-consistent loop from 2 to 5 is considered convergent, if the estimated error of eigenvalues is within a prescribed tolerance level ϵ .

III. EXAMPLE 1: SIMPLE HARMONIC OSCILLATOR

A. Numerical results

We first examine the classical eigenproblem of a single particle in a one-dimensional simple-harmonic-oscillator (SHO) potential, $V(x) = (\kappa/2)x^2$, to illustrate the basic procedure. For simplicity, we take the reduced units

$$\hbar^2/m = 1, \quad (16)$$

and

$$\kappa = 1, \quad (17)$$

as they can be rescaled and absorbed into redefinition of \mathbf{r} and E_n .

The network sketched in Fig. 1 is used in this paper, where we adopt a three-layer deep NN,

$$L = 3, \quad (18)$$

and solve the eigenproblem for various N up to $N + 1 = 32$, in a study of scalability of the algorithm. Several selections of the number of nodes per hidden layer, N_h , in the range

$$N_h = [36, 300] \quad (19)$$

are used in order to assess the effects of network size on the final results. All Θ parameters were randomly selected from a normal distribution of mean zero and variance 0.1, initially. The following results are produced by initially selecting a learning rate:

$$\eta = 0.001. \quad (20)$$

The penalty coefficients,

$$\beta = 1 \text{ and } \gamma = 100, \quad (21)$$

are used in (6).

To enable the MC sampling, an initial weight $W = 1$ for the range of $x = [-5, +5]$ was selected. A total of

$$S = 2 \times 10^3 \quad (22)$$

points were used between $[-5, 5]$ for initialization. This number is kept unchanged throughout the entire calculation, as the weight function evolves according to (14).

The overall performance of the solver can be monitored by how the calculated E_n and penalty function P vary as the computation progresses. As functions of epochs, they are shown in Figs. 2(a) and 2(b). The example plotted is from a single run to calculate $N + 1 = 16$ energy levels, where $N_h = 50$ hidden nodes per layer are used in the network. Initially the numerical procedure experiences an unstable period in searching for the solution. This is reflected by the uncertainty in E_n and high value of the penalty P , within the first 2×10^4 epochs. After that, these computation trajectories show that the solver begins to produce a reasonable estimate of the eigenvalues. During the computation, the NN output layer yields $N + 1$ wave functions in no particular order. The convergence in Fig. 2(a) gives rise to an energy sequence in a later stage, although rare events of switching two adjacent energy levels are also observed. As there is no prespecification of the appearance order of these energy levels, the final converged E_n ($n = 0, 1, 2, \dots, N$) in Fig. 2(a) are used to determine the order of the wave functions shown in Fig. 2(c).

The self-consistent procedure prescribed above numerically produces the excited energy states and wave functions. For comparison, the SHO eigenproblem has an analytic solution. In reduced units the eigenvalues are

$$E_n^{\text{SHO}} = (n + 1/2), \quad n = 0, 1, 2, \dots \quad (23)$$

and eigenfunctions are the physicists' Hermite functions [59,60]:

$$\Psi_n^{\text{SHO}}(x) = \left(\frac{1}{\pi}\right)^{1/4} \frac{1}{\sqrt{2^n n!}} H_n(x) e^{-x^2/2}, \quad (24)$$

where

$$H_n(x) = (-1)^n e^{x^2} \left(\frac{d}{dx}\right)^n e^{-x^2}. \quad (25)$$

The exact solution can be used to benchmark the performance of our numerical solver, but were not used in machine learning. All plots in Figs. 2(a) and 2(c) agree with the known exact solution, (23) and (24).

B. Error estimates and convergence criterion

The present example problem has an exact solution. In general, however, there is no *a priori* knowledge of the solution, hence a general criterion for convergence must be constructed to facilitate the termination of the machine-learning calculation.

The eigenvalues E_n fluctuate epoch by epoch, even near the final, converging period of the calculation, due to the nature of MC sampling [see Fig. 2(a)]. The E_n variations are more

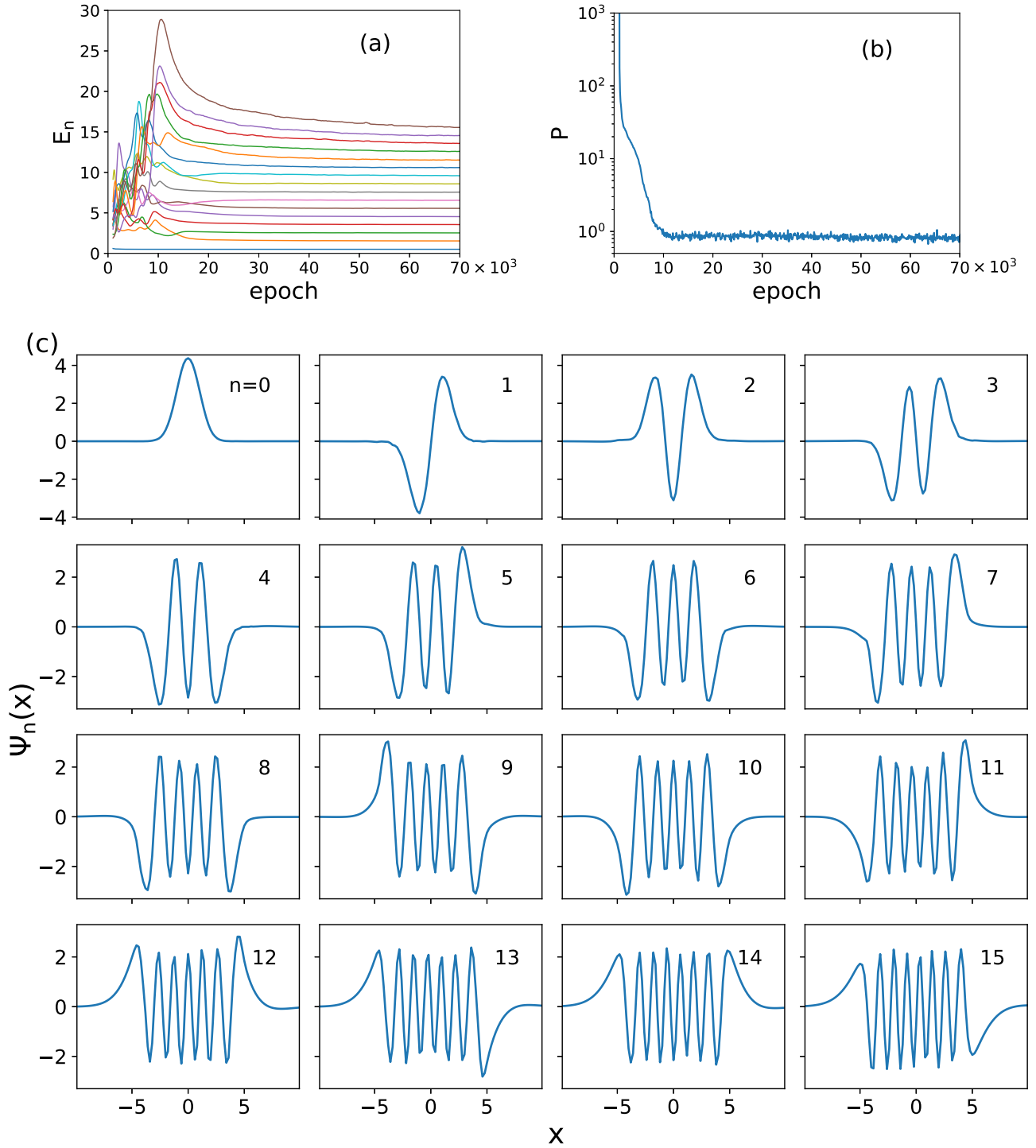


FIG. 2. NN solution for the example SHO problem, for excited states up to $N = 15$. The convergence of the eigenvalues E_n as functions of epoch [defined in (5)] is demonstrated in (a). The declining of the penalty function in (6) is displayed in (b). The final wave functions, for all $\Psi_n(x)$ where $n = 0, 1, 2, \dots, N$, are displayed in (c), arranged according to the magnitude of E_n , from low to high. To produce these particular trajectories of E_n and P , $L = 3$ hidden layers and $N_h = 50$ hidden nodes per layer are used.

drastic at the beginning and tend to smooth out at the later stage. The main idea is to take two adjacent blocks of epochs and examine the two mean values of E_n , each evaluated within their own blocks. If the difference between the two is comparable to the independent estimate of the numerical error, the machine-learning procedure terminates.

Here, to estimate the MC error, we use the successive m epochs as a statistical window. The variance of the n th computed energy level is calculated from

$$\sigma_n^2 = \overline{E_n^2} - \bar{E}_n^2. \quad (26)$$

The overbars represent the average taken from all measurements within the m epochs of the calculation. Note that each epoch itself already contains S MC sampling events. Then,

$$\bar{E}_n = \frac{1}{m} \sum_{j=1}^m \frac{1}{SA_j} \sum_{l=1}^S [\Psi_n^*(\mathbf{r}_l) \hat{H} \Psi_n^*(\mathbf{r}_l) / W(\mathbf{r}_l)]_j \quad (27)$$

and

$$\bar{E}_n^2 = \frac{1}{m} \sum_{j=1}^m \frac{1}{SA_j^2} \sum_{l=1}^S [\Psi_n^*(\mathbf{r}_l) \hat{H} \Psi_n^*(\mathbf{r}_l) / W(\mathbf{r}_l)]_j^2. \quad (28)$$

The subscript $j = 1, 2, 3, \dots, m$ labels the m epochs taken into consideration. The coefficient A_j is a normalization factor evaluated at the j th epoch:

$$A_j = \frac{1}{S} \sum_{l=1}^S [\Psi_n^*(\mathbf{r}_l) \Psi_n^*(\mathbf{r}_l) / W(\mathbf{r}_l)]_j. \quad (29)$$

One can refer back to Sec. II D for the definition of the weighted average. If all data points are statistically independent, then the standard deviation, calculated by taking the m epochs of S events, is

$$\delta_n = \sigma_n / \sqrt{mS}. \quad (30)$$

On average,

$$\delta = \frac{1}{N+1} \sum_{n=0}^N \delta_n = \sigma / \sqrt{mS}, \quad (31)$$

where

$$\sigma = \frac{1}{N+1} \sum_{n=0}^N \sigma_n \quad (32)$$

is the mean square-root variance, averaged over the $N+1$ energy levels.

The termination of a machine-learning session is determined by whether or not the E_n trajectories in Fig. 2(a) are flattened out. The mean-energy difference, δE , is taken from

$$\delta E = \left[\frac{1}{N+1} \sum_{n=0}^N (\bar{E}_n - \bar{E}'_n)^2 \right]^{1/2}, \quad (33)$$

where \bar{E}_n and \bar{E}'_n are, respectively, calculated from two successive blocks. The session is considered convergent if

$$\delta E \leq \epsilon, \quad (34)$$

where ϵ is a tolerance level. The selection of ϵ , however, cannot be arbitrarily small. The statistical average in (27) carries with it an uncertainty level, δ_n in (30). We adopted ϵ such that

$$\epsilon = \alpha \delta \quad (35)$$

where α is a numerical coefficient, selected here to be $\alpha = 2$. Hence the machine-learning procedure terminates at

$$\delta E / \sigma \leq \alpha / \sqrt{mS}. \quad (36)$$

The results presented in this section are based on a statistic window of

$$m = 1 \times 10^3. \quad (37)$$

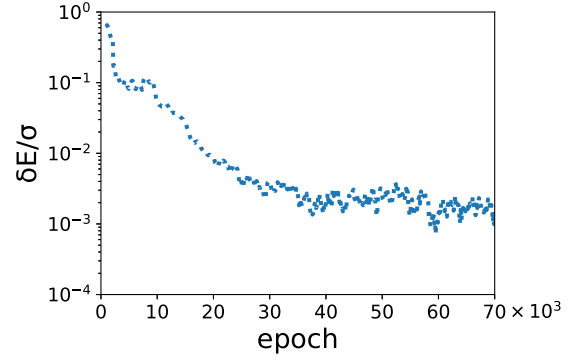


FIG. 3. Typical trajectory of the calculated $\delta E / \sigma$ [Eqs. (32) and (33)] as a function of epoch, of a run that determines the lowest $N+1 = 16$ energy states of a SHO. $\delta E / \sigma$ is used for identifying convergence of the self-consistent numerical procedure (see Sec. III B).

An example trajectory of the computed $\delta E / \sigma$ is plotted in Fig. 3.

How precise are the energy levels determined from the machine-learning procedure? There are two main error sources in the current NN approach. As in any procedures based on MC sampling, the first originates from the stochastic errors and it can be controlled by increasing the sample size, S , as reflected in the estimate, (30). The second column in Table II shows the standard deviation δ_n estimated this way, calculated at the termination of the calculation, for energy level n . The second has its roots in a typical trial-function approach; as such the calculated eigenvalues are the upper bounds of the actual values. The comparisons between \bar{E}_n and the benchmarking E_n^{SHO} are given in the third column, from which one always has $\bar{E}_n > E_n^{\text{SHO}}$. In principle, this can

TABLE II. The relative errors of the energy levels of the SHO example, at the convergence determined by the criterion in (34). The second column is δ_n / \bar{E}_n , calculated according to (5) and (30), numerically. The third column is a comparison with the exact solution in (23).

Level n	δ_n / \bar{E}_n	$\bar{E}_n / E_n^{\text{SHO}} - 1$
$n = 0$	8.9×10^{-4}	2.1×10^{-3}
$n = 1$	6.4×10^{-4}	3.0×10^{-3}
$n = 2$	5.8×10^{-4}	5.2×10^{-3}
$n = 3$	5.3×10^{-4}	6.3×10^{-3}
$n = 4$	5.1×10^{-4}	2.0×10^{-3}
$n = 5$	4.9×10^{-4}	7.8×10^{-3}
$n = 6$	4.9×10^{-4}	2.8×10^{-3}
$n = 7$	4.9×10^{-4}	3.9×10^{-3}
$n = 8$	4.8×10^{-4}	4.3×10^{-3}
$n = 9$	4.6×10^{-4}	3.6×10^{-3}
$n = 10$	4.6×10^{-4}	2.0×10^{-3}
$n = 11$	4.4×10^{-4}	3.3×10^{-3}
$n = 12$	6.5×10^{-4}	2.8×10^{-3}
$n = 13$	7.5×10^{-4}	2.5×10^{-3}
$n = 14$	6.3×10^{-4}	4.5×10^{-3}
$n = 15$	5.5×10^{-4}	5.5×10^{-3}
Average	5.7×10^{-4}	4.0×10^{-3}

be controlled by the accuracy between the NN represented wave functions and the presumably unknown, actual wave functions. Beyond SHO potentials, how the complicity of a Hamiltonian of a real system affects these two types of errors remains to be further studied.

C. Network performance

The performance of the machine-learning solver used here depends on a few factors. This includes the complexity of the network (represented by the size of depth L and width of the hidden layers N_h), the difficulty level of the wave functions that the neural network attempts to model (represented by N , the number of excited states), the final precision required, and the efficiency of an optimization method used in optimizing J . Here we explore the dependence on the first two, by fixing the precision requirement through (36) and ADAM optimization method in (15).

The maximal number of epochs needed to minimize the cost function, M , and the actual computational time, T , are used as indicators. Taking into account the stochastic nature of the problem, in randomly setting up the initial network parameters Θ and moving the sampling set $\{\mathbf{r}_l\}$ according to the Metropolis procedure, each data point presented in Figs. 4(a) and 4(b) is obtained from 20 independent machine-learning runs. By further fixing the hidden layer depth at $L = 3$, both M and T are examined as functions of N_h and N .

Figure 4(a) demonstrates how M depends on N , for various choices of N_h . At a given N , data indicate a lower M from a larger N_h , which reflects the fact that fewer epochs are needed to model the current problem, by a more complicated network. It is worth noting that a small network ($N_h = 36$) already has the ability to model a high excited state ($N + 1 = 32$), but requires a longer learning process.

Though a smaller M is achieved by a more complicated NN (larger N_h) to perform the same task for the same N , the real computation time, T , in units of seconds, can be longer. This is due to the actual time that the solver takes to evaluate J and update the parameters through the deepest decent method. Figure 4(b) demonstrates this dependence.

As N_h is adequately large, both data sets, M and T , display a linear line on the log-log plot. Numerically the power laws,

$$M \propto N^{\mu_M} \quad \text{and} \quad T \propto N^{\mu_T}, \quad (38)$$

are observed. The scaling exponents μ_M and μ_T are obtained from the log-log plots by a linear fit and shown in Fig. 4(c) for various N_h . As $N_h \gg 1$, these exponents approach the asymptotic values, $\mu_M \approx 0.7$ and $\mu_T \approx 1.5$.

D. Monte Carlo weight convergence

An important component of the current algorithm design is the implementation of the MC sampling described in Sec. IID. It controls the range of the sampling points in the variable space according to a self-consistently determined weight function, related to the $N + 1$ wave functions of an N excited-state problem.

From the SHO example, Figs. 5(a)–5(e) demonstrate the convergence of the MC weights in typical runs, for $N + 1 = 2, 4, 8, 16$, and 32. Initially the set $\{x_j\}$ ($j = 1, \dots, S$) is ran-

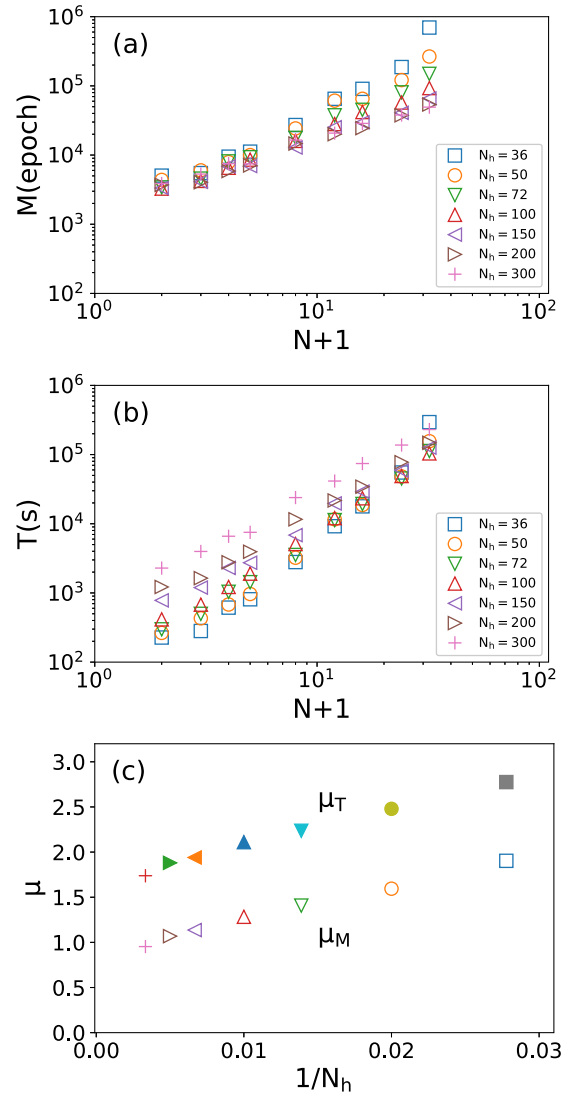


FIG. 4. Efficiency of the machine-learning solver in solving SHO excited states. The maximal number of epochs, M , and the real computation time needed for convergence, T (in seconds), are shown in (a) and (b), to calculate a problem with N excited states. Different network sizes, through the variation of the number of hidden nodes N_h , are examined. Each data point is an average of M and T , produced from 20 independent machine-learning runs, in order to produce the needed statistics. The errorbars are smaller than the size of the plotted symbols. The asymptotic power laws are empirically determined by fitting the data to (38). The numerical values, of thus determined scaling exponents, are displayed in plot (c) as a function of $1/N_h$.

domly selected within the $[-5, 5]$ range, following an initial weight $W = 1$. The almost uniform distributions of the S initial sampling points are demonstrated by the violin plot at the beginning epoch in these plots. As the computation proceeds, summarized in Sec. IIE, the set evolves (shrinks or spreads) beyond the original range and finally converges to a stable range. The final range of $\{x_j\}$ ($j = 1, \dots, S$) is not prespecified, as we purposely let it float according to the probability distribution $W(x)$. The self-consistent numerical procedure enables the different converged weights for various

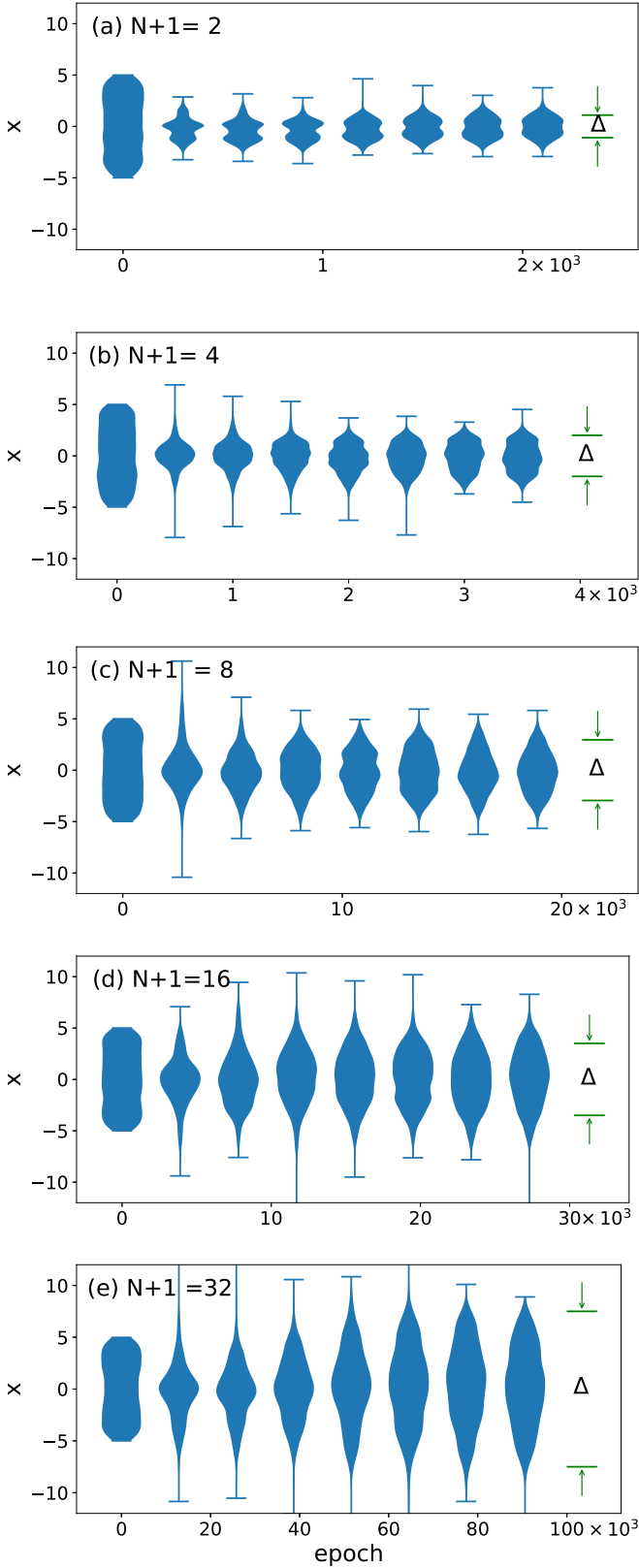


FIG. 5. (a)–(e) Violin plots of the evolution of the Monte Carlo weights as machine learning proceeds for $N + 1 = 2, 4, 8, 16$, and 32 , respectively. These weights are visualized by vertical bars at various epochs, on which the thickness corresponds to weight magnitudes. The Monte Carlo displacement magnitudes, Δ , are indicated on the far right-hand side.

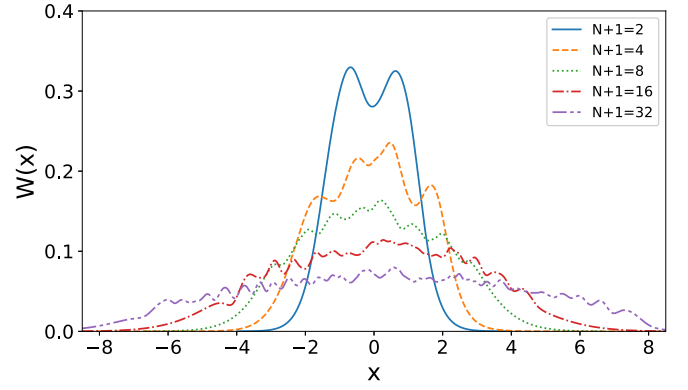


FIG. 6. Converged weights $W(x)$ for $N + 1 = 2, 4, 8, 16$, and 32 .

N , displayed in Fig. 6. The broader weight function for higher N is due to the fact that the wave functions extend to a larger x range, displayed in Fig. 2(c). The broadened $W(x)$ ultimately brings more stochastic fluctuations through (13), which yields an error that can be controlled by increasing sampling size S and sampling time m ; the computational price is higher for higher N , of course.

The selection of a new MC sampling set, $\{x'_j\}$ ($j = 1, \dots, S$), is attempted by displacing the sampling points to new positions at a random distance, from an old sampling set $\{x_j\}$ ($j = 1, \dots, S$). The magnitude of the move, Δ , controls the acceptability of the new set. This MC parameter is initially set at $\Delta = 1$ and then self-adjusted in our algorithm, to guarantee that the mean acceptance rate of the MC move is approximately 50%. The final convergent Δ values are indicated in Figs. 5(a)–5(e) for different N , by the green ranges. Different initial guesses, generated from a Gaussian distribution or taken from an existing solution, have been attempted and have no major effects on the final convergence.

The enabling factor of the MC procedure is the representation of the wave functions by the NN in a continuum space. Unlike a traditional solver where the wave functions are described by the specified representative nodes (either in real or in spectral-function space), the NN representation gives us the opportunity to sample at any arbitrary points. This property liberates us from the preselected and fixed representative nodes, and allows us to move the sampling points freely in the variable space, in reference to a continuum weight function.

IV. EXAMPLE 2: TWO-DIMENSIONAL HARMONIC OSCILLATOR

A. Decoupled oscillators

We now examine the solution of the Schrödinger equation for a two-dimensional harmonic oscillator, where the variable space is $\mathbf{r} = (x_1, x_2)$. For a decoupled potential energy (in reduced units)

$$V(\mathbf{r}) = \frac{1}{2}(x_1^2 + x_2^2), \quad (39)$$

the exact solution is known and contains energy degeneracy. The eigenvalues, for example, are related to two quantum numbers, n_1 and n_2 , associated with x_1 and x_2 directions, respectively.

TABLE III. Numerical solution from the deep NN of the energy eigenvalues (in reduced units), for two decoupled harmonic oscillators (column 2), two coupled harmonic oscillators (column 4), and five coupled harmonic oscillators (column 6), described in Secs. IV A, IV B, and V. The first column is the unilateral label n , produced from numerically sorting the energy levels. The third, fifth, and seventh columns contain identifications of these states in reference to the exact solution of the oscillator problems, in terms of two quantum numbers (n_1, n_2) for the two-dimensional problem, and five quantum numbers (n_1, n_2, \dots, n_5) for the five-dimensional problem.

n	\bar{E}_n	(n_1, n_2)	\bar{E}_n	(n_1, n_2)	\bar{E}_n	$(n_1, n_2, n_3, n_4, n_5)$
0	1.01	(0,0)	1.02	(0,0)	2.93	(0,0,0,0,0)
1	2.01	(0,1)	1.95	(1,0)	3.36	(1,0,0,0,0)
2	2.02	(1,0)	2.15	(0,1)	3.71	(0,1,0,0,0)
3	3.01	(0,2)	2.88	(2,0)	3.85	(2,0,0,0,0)
4	3.02	(1,1)	3.07	(1,1)	4.11	(0,0,1,0,0)
5	3.02	(2,0)	3.29	(0,2)	4.13	(1,1,0,0,0)
6	4.01	(0,3)	3.80	(3,0)	4.51	(0,2,0,0,0)
7	4.02	(1,2)	3.99	(2,1)	4.55	(1,0,1,0,0)
8	4.03	(2,1)	4.20	(1,2)	4.57	(0,0,0,1,0)
9	4.03	(3,0)	4.41	(0,3)	4.83	(0,0,0,0,1)
10	5.02	(0,4)	4.73	(4,0)	4.93	(0,1,1,0,0)
11	5.02	(1,3)	4.94	(3,1)	5.01	(1,0,0,1,0)
12	5.02	(2,2)	5.14	(2,2)	5.29	(1,0,0,0,1)
13	5.03	(3,1)	5.32	(1,3)	5.33	(0,0,2,0,0)
14	5.03	(4,0)	5.56	(0,4)	5.37	(0,1,0,1,0)
15	6.04	(5,0)	5.67	(5,0)	5.60	(0,1,0,0,1)

The self-consistent numerical procedure described in Sec. II, on the other hand, uses a single, generalized quantum number n for the output from the deep NN. Using the same parameters specified in (18)–(22) and the same convergence criterion in Sec. III B, we numerically solved this problem up to $N = 15$ excited states, using $N_h = 100$. The NN solutions for the eigenvalues are listed in the second column of Table III. The assignment of n to these energy levels is done according to numerical sorting $\bar{E}_0 < \bar{E}_1 \leq \bar{E}_2 \dots \leq \bar{E}_N$.

One can observe, from the numerical values, that the energy levels of each group— $n = 1, 2$; $n = 3, 4, 5$; $n = 6, 7, 8, 9$; etc.—are energetically degenerate within the numerical uncertainty. This is done without the knowledge of the exact solution of two SHOs, separately in x_1 and x_2 directions:

$$E^{\text{SHO}}(n_1, n_2) = \left(\frac{1}{2} + n_1\right) + \left(\frac{1}{2} + n_2\right). \quad (40)$$

The third column lists the degenerate quantum states in (n_1, n_2) from the exact solution, identified *a posteriori*, for comparison.

As the requirement of wave-function orthogonality is embedded in our algorithm design, the wave functions of each group of these energetically degenerated states are made mutually orthogonal. Figure 7 displays the wave functions of the $N + 1 = 16$ states listed in the second column of Table III. These three-dimensional plots, blindly obtained without knowing the exact solution, agree with similar plots based on the exact solutions.

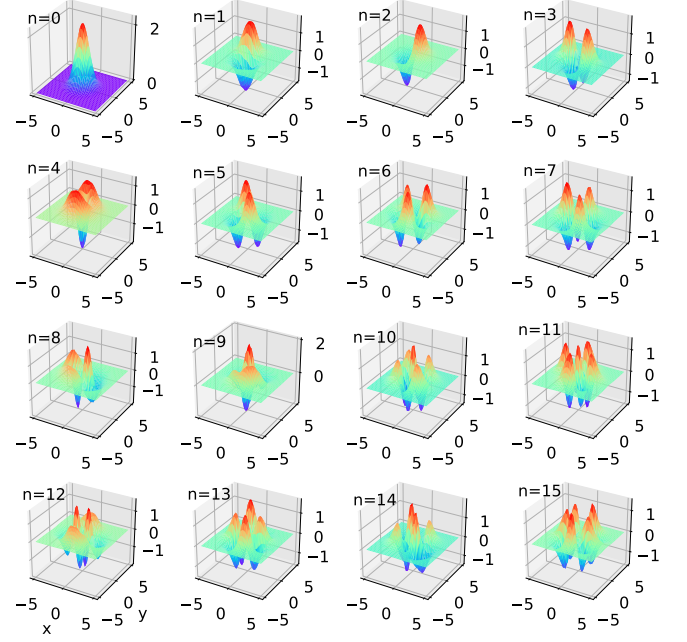


FIG. 7. Three-dimensional plots of the wave functions $\Psi_n(x, y)$, obtained from the NN-based solver, of the $N + 1 = 16$ energy states. The n labels are identified numerically, following the sequence of \bar{E}_n .

B. Coupled harmonic oscillators

It comes as no surprise that the algorithm works equally well, in solving the eigenproblem of the coupled oscillators, where

$$V(\mathbf{r}) = \frac{1}{2}(a_{11}x_1^2 + 2a_{12}x_1x_2 + a_{22}x_2^2). \quad (41)$$

As an example, the arbitrarily selected coefficients used here are $a_{11} = 0.8851$, $a_{12} = -0.1382$, and $a_{22} = 1.1933$. The same parameter values of L , N_h , η , β , γ , S , m , and ϵ stated in the last subsection, are used here.

The numerical results for \bar{E}_n are displayed in the fourth column of Table III. Because of the mismatch of the coefficients of the diagonalized form [see below, (42)], the energy levels are no longer degenerate. Their corresponding wave functions $\Psi_n(x_1, x_2)$ are plotted by two illustration methods in Fig. 8. The three-dimensional plots can be compared with those in Fig. 7. The footprint plots further illustrate the locations of the eigenfunction peaks in the (x_1, x_2) plane.

The coupled oscillators, given by the potential function in (41), can be decoupled, in terms of a diagonalized potential form:

$$V(\mathbf{r}) = \frac{1}{2}(\lambda_1 q_1^2 + \lambda_2 q_2^2). \quad (42)$$

The coefficients $\lambda_1 = 0.835$ and $\lambda_2 = 1.243$ are the eigenvalues of the $\{a_{ij}\}$ matrix. We ensured that these eigenvalues are positive when $\{a_{ij}\}$ are selected, to form a bound potential problem. The two canonical variables are $q_1 = -0.9315x_1 + 0.3637x_2$ and $q_2 = -0.3637x_1 - 0.9315x_2$. Then, based on the decoupled version, there exists an exact solution:

$$E^{\text{SHO}}(n_1, n_2) = \left(\frac{1}{2} + n_1\right)\lambda_1^{1/2} + \left(\frac{1}{2} + n_2\right)\lambda_2^{1/2}, \quad (43)$$

where the two quantum numbers $n_1 = 0, 1, 2, \dots$ and $n_2 = 0, 1, 2, \dots$. The fifth column in Table III contains the identi-

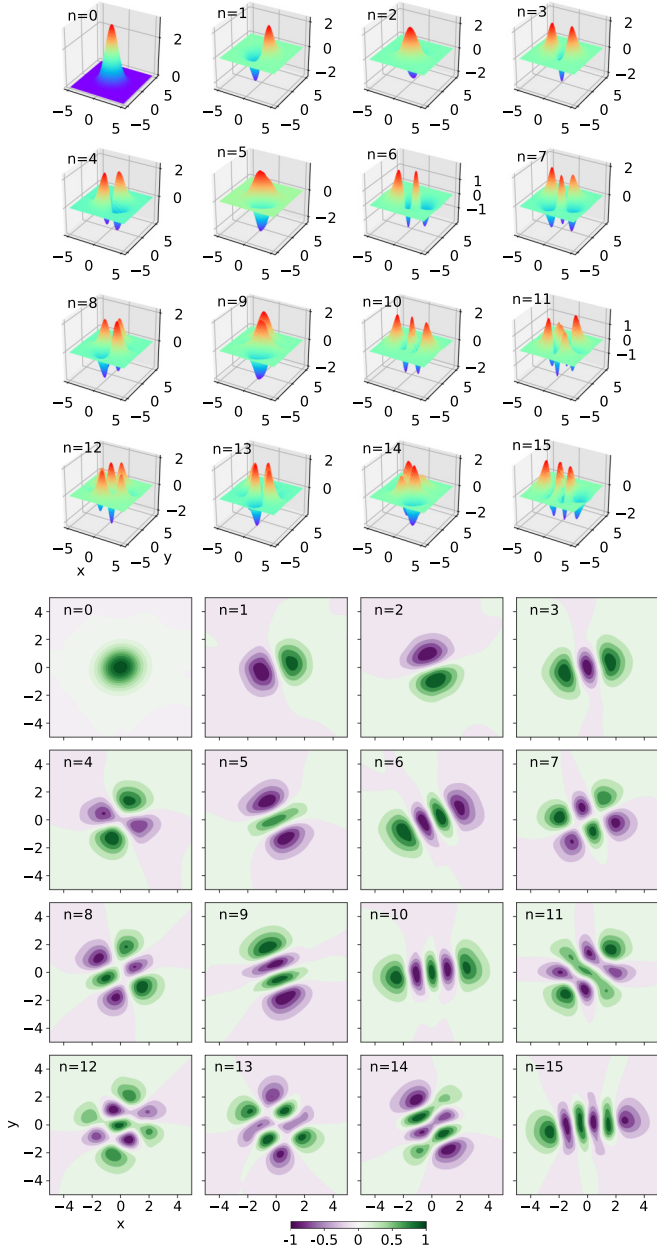


FIG. 8. Illustrations of the numerical solution of the eigenfunctions for the coupled harmonic problem in Sec. IV B. The first 16 plots are three-dimensional plots of the function $\Psi_n(x, y)$. The next 16 plots are footprint plots of the same functions, but now projected onto the (x, y) plane, where the intensity of the color represents the function values.

cation of these quantum states with the numerically produced states in terms of n .

V. EXAMPLE 3: FIVE COUPLED HARMONIC OSCILLATORS

Next, we examine the example of a $D = 5$ problem. The potential energy is written as

$$V(x_1, x_2, x_3, \dots, x_D) = \frac{1}{2} \sum_{i=1}^D (\lambda_i q_i^2). \quad (44)$$

TABLE IV. Values of the coefficients used in Sec. V. See Eqs. (44) and (45).

i	λ_i	b_{i1}	b_{i2}	b_{i3}	b_{i4}	b_{i5}
1	0.1641	-0.2362	0.7505	-0.3577	0.4966	-0.0800
2	0.5744	-0.2231	0.2015	-0.0752	-0.3208	0.8950
3	1.3732	0.5401	0.6092	0.4518	-0.3530	-0.0911
4	2.5335	0.4098	-0.1384	0.3439	0.7197	0.4202
5	3.5373	-0.6593	0.0760	0.7375	0.0887	-0.0876

The variables x_j ($j = 1, 2, \dots, D$) are expressed in terms of q_i by

$$q_i = \sum_{j=1}^D b_{ij} x_j. \quad (45)$$

For the particular example discussed here, the values of the coefficients are listed in Table IV. Care has been taken to ensure that the vector $(b_{i1}, b_{i2}, \dots, b_{i5})$ is orthonormal to the vector $(b_{j1}, b_{j2}, \dots, b_{j5})$.

The NN-based solver treats $\mathbf{r} = (x_1, x_2, \dots, x_D)$ as input, for which the potential energy is in a coupled form. To produce the numerical results in this section, the same parameter values for $L, N_h, \eta, \beta, \gamma, S, m$, and ϵ , stated in Sec. IV B, are used. We let $N = 15$ to find the lowest 16 energy levels. The numerically produced eigenvalues are displayed in column 6 of Table III, with a single n label, according to the ascending order of the eigenvalues.

In comparison, the exact solution of the problem gives

$$E^{\text{SHO}}(n_1, n_2, \dots, n_D) = \sum_{i=1}^D \left(\frac{1}{2} + n_i \right) \lambda_i^{1/2}, \quad (46)$$

where each of the five quantum numbers n_i ($i = 1, 2, \dots, 5$) takes the values 0, 1, 2, \dots . The last column of Table III lists the quantum states according to this labeling system, after matching the numerical solution with the exact results.

It is difficult to illustrate the wave functions in a high- D space. We use the footprint approach in Fig. 9, where the wave-function plot for a given n , $\Psi_n(x_1, x_2, \dots, x_5)$, consists of a matrix of 5×5 subplots. The NN outputs of the final results are displayed to show the wave functions. The subplot at matrix “element” ij is a footprint plot for the two-variable function $\Psi_n(0, 0, \dots, x_i, \dots, x_j, \dots, 0, 0)$, where $x_{i'} = x_{j'} = 0$ when $i' \neq i$ and $j' \neq j$ are used. The exception is the diagonal element, where $i = j$. The single-variable function $\Psi_n(0, 0, \dots, x_i, 0, 0)$ is displayed by a line plot.

VI. EXAMPLE 4: GROUND STATES OF COUPLED HARMONIC OSCILLATORS

In the last example, we explore the scalability of the NN-based solver, for solving the ground-state problem of high- D systems, for a variable $\mathbf{r} = [x_1, x_2, x_3, \dots, x_D]$ that contains up to $D = 128$ components. Various sizes of the deep NN are used by adjusting the number of hidden nodes N_h , in order to explore the effects of the network size. The essential numerical procedure remains the same as described in Sec. II

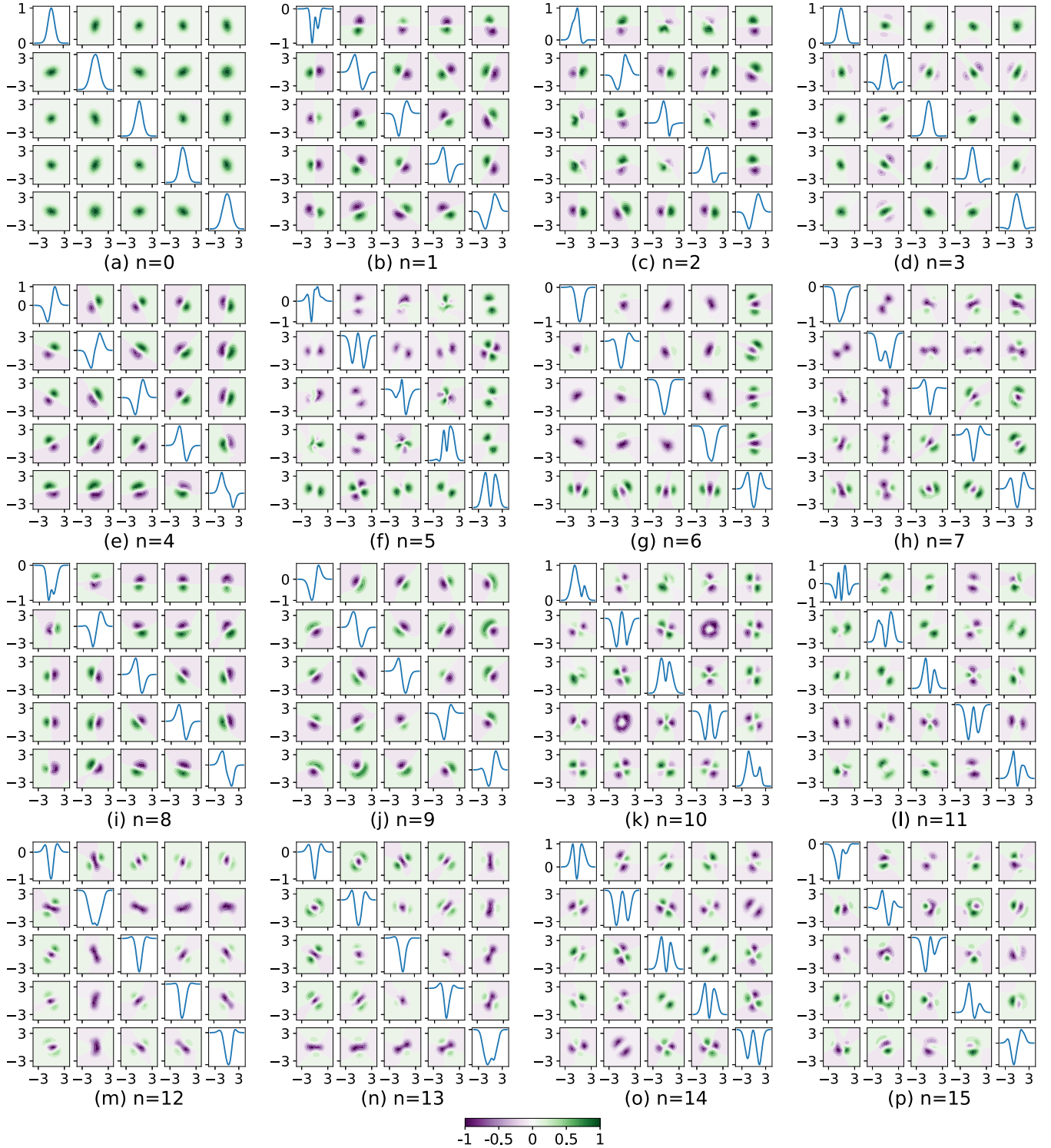


FIG. 9. Footprint plots of the wave functions obtained by the NN-based solver for the $N + 1 = 16$ quantum states of lowest-energy levels, of the $D = 5$ example discussed in Sec. V. The plots (a)–(p) correspond to energy levels E_n , arranged from $n = 0$ to $n = N + 1 = 16$, respectively. The values of E_n can be found in Table III. For a given n , each plot contains $D \times D$ subplots, forming a matrix of plots. The off-diagonal subplots at matrix element ij are two-dimensional footprint plots of a wave function in which the variables x_i and x_j are used as the plotting coordinates and all other variables are set to zero. The line plots at diagonal elements ii are simply the wave function displayed as a function of x_i when all other variables are set to zero.

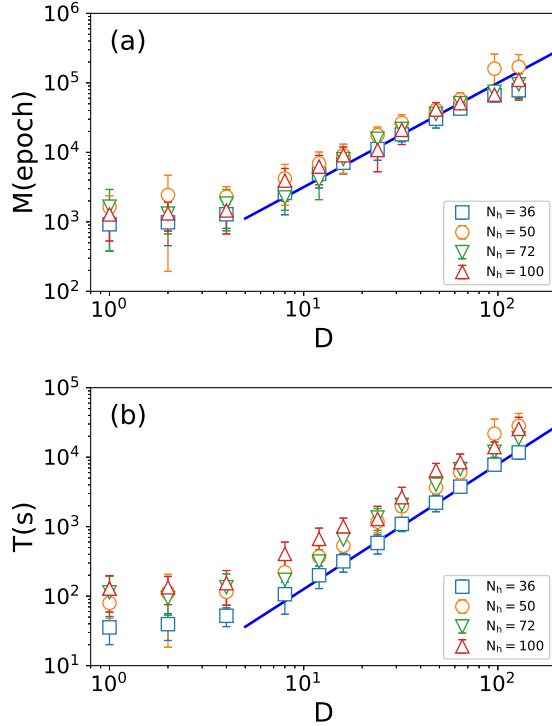


FIG. 10. Log-log plots of (a) the mean maximum epochs (M) and (b) the mean total computational time (T) that the NN-based solver takes to converge, as a function of D , the dimensionality of the variable \mathbf{r} in the Schrödinger equation, for a coupled oscillator problem. The solid blue lines represent the empirical power laws in (47).

and all parameter values, L , η , β , γ , S , m , and ϵ , are the same as those used in Sec. V.

The potential function has the same form as in (44) and (45). When a run starts, the values of λ_i ($i = 1, 2, \dots, D$) are randomly generated in the ranges $[0.1, 1]$; for each i , the coefficients $\{b_{ij}\}$ ($j = 1, 2, \dots, D$) are also randomly generated with each element taking a value in the range $[-1, 1]$. Note that for this subsection no orthonormal considerations were made to form the b_{ij} matrix, as the aim here is not to compare with the exact solution.

The maximum epoch M and the computational time T that the NN-based solver takes to converge are recorded. To collect adequate statistics, for each given D and N_h , a total of 20 independent runs, started from fresh network parameters Θ and an independent random set of λ_i and b_{ij} , are conducted. The data collected from the 20 runs, on M and T , are then averaged and plotted as a single data point in Fig. 10. The standard deviations of the plotted points are also shown by the errorbars.

Regardless of D , the same number of D -dimensional sampling points, $S = 2 \times 10^3$, are used. Described in Sec. IID, as the solver proceeds, these points move in the D -dimensional space, by taking a controlled MC displacement. The typical storage size for all sampling points is a moderate $D \times S$, and is refreshed if a new MC attempt is successful.

The scalings of M and T on D , as displayed in Figs. 10(a) and 10(b), typically follow power laws at large D :

$$M \propto D^{\nu_M} \quad \text{and} \quad T \propto D^{\nu_T}. \quad (47)$$

It is noticeable that the data sets of M for various N_h collapse into approximately the same trend at large D . Hence, in terms of the maximum epoch that the solver takes to solve a D -dimensional problem, the performance of the NN-based solver is almost N_h insensitive, provided, of course, N_h is adequately large. The empirical fit of the power laws gives

$$\nu_M = 1.5 \quad \text{and} \quad \nu_T = 1.8. \quad (48)$$

Both are moderately small scaling exponents.

VII. SUMMARY

This paper proposes and explores a nontraditional numerical scheme for solving a multivariable static Schrödinger equation in a bound potential energy, from the ground state up to a significantly large number of excited states. The algorithm design exploits the basic properties of a deep neural network, which simultaneously yields an output for wave functions to a prespecified level (Secs. IIA and IIB). The complexity of a high-dimensional problem is dealt with by introducing the importance-sampling Monte Carlo technique to efficiently cover the variable space (Sec. IID).

With a given potential energy, the self-consistent learning algorithm produces a numerical solution for the eigenenergies and wave functions, arranged sequentially according to the magnitude of the eigenenergies, from the ground-state energy to an excited-state energy. The mutual orthogonality of all corresponding wave functions is enforced when the self-consistent procedure converges, regardless of the possible existence of energy degeneracy (Sec. IIC).

A number of examples are used to showcase the properties of the proposed scheme. A numerical study is carried out for the analytically known simple harmonic oscillator problem in Sec. III. The low $N + 1 = 32$ energy levels are numerically reproduced, which allows for an in-depth analysis of error estimates, algorithm performance in terms of computational time versus network size, etc. Another numerical study then follows in Sec. IV, for two decoupled and coupled oscillators (which have analytically known solutions). The numerical solver treats energetically degenerate states in the same way as in a nondegenerate problem—numerically no two energies are perfectly identical. Although a manual comparison with the exact solution, including the information on energy degeneracies, can be made after the numerical solution is found, the numerical solution itself is adequate to facilitate an analysis of the quantum properties. Two high-dimensional example problems are then explored in Secs. V and VI.

The basic ingredients in the current algorithm design are NNs as the variational functions and an importance-sampling Monte Carlo evaluator. The latter is a common theme used in quantum Monte Carlo methods, where different approaches have been taken to design the trial functions [40,41]. For example, in a classical paper, Ceperley and Alder used the squared ground-state wave function as the weight for Monte Carlo sampling, where the wave function takes the Bijl-Dingle-Jastrow form [45]; the ground state of an electron gas was successfully calculated [61]. Reference [50] is a recent realization of this method for estimating the excited states of benzene, etc. In another example, exploiting the similar-

ity between the Schrödinger equation and a typical diffusion equation, a diffusion Monte Carlo procedure has led to the finding of the ground and excited states of malonaldehyde [62].

The coupled harmonic potentials are used to demonstrate and elaborate the basic concept of our multistate solver. What we generally called a D -dimensional space could be actually broken down to the three-dimensional spaces required by a multiple-particle problem. It paves the way to developing more sophisticated NN multistate solvers for realistic quantum-chemistry systems, such as those recently examined in [38,39] by specially designed NNs (which are for the ground state only). A potential application is to solve for the

low-lying vibrational states of a well-studied, real molecule for which some experimental data as well as QMC results (for example, those in [62]) are available.

ACKNOWLEDGMENTS

H.L. and J.Z.Y.C. are grateful to Qian-Shi Wei and Marcel Nooijen for discussion on other related topics at the beginning stage of this project. We wish to acknowledge the financial support from the National Natural Science Foundation of China (Grants No. 11775161 and No. 21873009) as well as the Natural Sciences and Engineering Council of Canada.

-
- [1] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning* (Springer, New York, 2013).
 - [2] C. E. Rasmussen, in *Advanced Lectures on Machine Learning* (Springer, New York, 2004), pp. 63–71.
 - [3] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, New York, 2006).
 - [4] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine Learning: An Artificial Intelligence Approach* (Springer, New York, 2013).
 - [5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT, Cambridge, MA, 2016).
 - [6] A. Meade and A. Fernandez, *Math. Comput. Mod.* **20**, 19 (1994).
 - [7] I. E. Lagaris, A. Likas, and D. I. Fotiadis, *IEEE Trans. Neural Networks* **9**, 987 (1998).
 - [8] B. P. van Milligen, V. Tribaldos, and J. A. Jiménez, *Phys. Rev. Lett.* **75**, 3594 (1995).
 - [9] M. Quito, C. Monterola, and C. Saloma, *Phys. Rev. Lett.* **86**, 4741 (2001).
 - [10] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, *IEEE Trans. Neural Networks* **11**, 1041 (2000).
 - [11] L. P. Aarts and P. Van Der Veer, *Neural Proc. Lett.* **14**, 261 (2001).
 - [12] A. Malek and R. S. Beidokhti, *Appl. Math. Comput.* **183**, 260 (2006).
 - [13] R. S. Beidokhti and A. Malek, *J. Franklin Inst.* **346**, 898 (2009).
 - [14] M. Kumar and N. Yadav, *Comput. Math. Appl.* **62**, 3796 (2011).
 - [15] K. Rudd and S. Ferrari, *Neurocomputing* **155**, 277 (2015).
 - [16] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *J. Comput. Phys.* **378**, 686 (2019).
 - [17] M. Raissi, *J. Mach. Learning Res.* **19**, 1 (2018).
 - [18] J. Han, A. Jentzen, and W. E, *Proc. Natl. Acad. Sci.* **115**, 8505 (2018).
 - [19] W. E, J. Han, and A. Jentzen, *Commun. Math. Stat.* **5**, 349 (2017).
 - [20] Q. Wei, Y. Jiang, and J. Z. Y. Chen, *Phys. Rev. E* **98**, 053304 (2018).
 - [21] Y. Yang, M. Hou, J. Luo, and Z. Tian, *J. Intelligent Fuzzy Syst.* **38**, 3445 (2020).
 - [22] M. Magill, A. M. Nagel, and H. W. de Haan, *Phys. Rev. Research* **2**, 033110 (2020).
 - [23] G. Torlai and R. G. Melko, *Phys. Rev. B* **94**, 165134 (2016).
 - [24] C. Desgranges and J. Delhommelle, *J. Chem. Phys.* **149**, 044118 (2018).
 - [25] D. Wu, L. Wang, and P. Zhang, *Phys. Rev. Lett.* **122**, 080602 (2019).
 - [26] L. Huang and L. Wang, *Phys. Rev. B* **95**, 035105 (2017).
 - [27] L. Huang, Y.-F. Yang, and L. Wang, *Phys. Rev. E* **95**, 031301(R) (2017).
 - [28] L. Wang, *Phys. Rev. E* **96**, 051301(R) (2017).
 - [29] W. Yu, Y. Liu, Y. Chen, Y. Jiang, and J. Z. Y. Chen, *J. Chem. Phys.* **151**, 031101 (2019).
 - [30] S. Whitelam, D. Jacobson, and I. Tamblyn, *J. Chem. Phys.* **153**, 044113 (2020).
 - [31] J. Li, H. Zhang, and J. Z. Y. Chen, *Phys. Rev. Lett.* **123**, 108002 (2019).
 - [32] I. Lagaris, A. Likas, and D. Fotiadis, *Comput. Phys. Commun.* **104**, 1 (1997).
 - [33] H. Nakanishi and M. Sugawara, *Chem. Phys. Lett.* **327**, 429 (2000).
 - [34] M. Sugawara, *Comput. Phys. Commun.* **140**, 366 (2001).
 - [35] Y. Shirvany, M. Hayati, and R. Moradian, *Commun. Nonlinear Sci. Numer. Simul.* **13**, 2132 (2008).
 - [36] K. Mills, M. Spanner, and I. Tamblyn, *Phys. Rev. A* **96**, 042113 (2017).
 - [37] P. Teng, *Phys. Rev. E* **98**, 033305 (2018).
 - [38] D. Pfau, J. S. Spenser, and A. G. D. G. Mathews, *Phys. Rev. Research* **2**, 033429 (2020).
 - [39] J. Hermann, Z. Schätzle, and F. Noé, *Nat. Chem.* **12**, 891 (2020).
 - [40] J. B. Anderson, *Quantum Monte Carlo: Origins, Development, Applications* (Oxford University, New York, 2007).
 - [41] F. Becca and S. Sprella, *Quantum Monte Carlo Approaches for Correlated Systems* (Cambridge University, Cambridge, England, 2017).
 - [42] K. Hornik, M. Stinchcombe, and H. White, *Neural Networks* **2**, 359 (1989).
 - [43] G. Cybenko, *Math. Control, Signals and Systems* **2**, 303 (1989).
 - [44] K. Hornik, *Neural Networks* **4**, 251 (1991).
 - [45] D. Ceperley, *Phys. Rev. B* **18**, 3126 (1978).
 - [46] Y. LeCun, Y. Bengio, and G. Hinton, *Nature (London)* **521**, 436 (2015).
 - [47] M. Rupp, A. Tkatchenko, K. R. Müller, and O. A. von Lilienfeld, *Phys. Rev. Lett.* **108**, 058301 (2012).

- [48] R. Bellman, *Dynamic Programming* (Princeton University, Princeton, NJ, 1957).
- [49] G. Carleo and M. Troyer, *Science* **355**, 602 (2017).
- [50] S. Pathak, B. Busemeyer, J. N. B. Rodrigues, and L. K. Wagner, *J. Chem. Phys.* **154**, 034101 (2021).
- [51] S. Manzhos, *Machine Learning: Science and Technology* **1**, 013002 (2020).
- [52] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Nature (London)* **323**, 533 (1986).
- [53] Please see <https://www.tensorflow.com>.
- [54] S. Kakade, S. Shalev-Shwartz, and A. Tewari, *J. Mach. Learn. Res.* **13**, 1865 (2012).
- [55] N. S. G. Hinton, A. K. I. Sutskever, and R. Salakhutdinov, *J. Mach. Learn. Res.* **15**, 1929 (2014).
- [56] D. P. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*, 4th ed. (Cambridge University, Cambridge, England, 2014).
- [57] L. Tierney, *Ann. Stat.* **22**, 1701 (1994).
- [58] D. Kingma and J. Ba, [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014).
- [59] N. M. Atakishiev and S. K. Suslov, *Theor. Math. Phys.* **85**, 1055 (1990).
- [60] D. J. Griffiths and D. F. Schroeter, *Introduction to Quantum Mechanics*, 3rd ed. (Cambridge University, Cambridge, England, 2018).
- [61] D. M. Ceperley and B. J. Alder, *Phys. Rev. Lett.* **45**, 566 (1980).
- [62] Y. Wang, B. J. Braams, J. M. Bowman, S. Carter, and D. P. Tew, *J. Chem. Phys.* **128**, 224314 (2008).