A semigroup method for high dimensional elliptic PDEs and eigenvalue problems based on neural networks

Haoya Li [a,*], Lexing Ying [b,1]

[a] *Department of Mathematics, Stanford University, Stanford, CA 94305, USA*
[b] *Department of Mathematics and ICME, Stanford University, Stanford, CA 94305, USA*

ARTICLE INFO

ABSTRACT

In this paper, we propose a semigroup method for solving high-dimensional elliptic partial differential equations (PDEs) and the associated eigenvalue problems based on neural networks. For the PDE problems, we reformulate the original equations as variational problems with the help of semigroup operators and then solve the variational problems with neural network (NN) parameterization. The main advantages are that no mixed second-order derivative computation is needed during the stochastic gradient descent training and that the boundary conditions are taken into account automatically by the semigroup operator. Unlike popular methods like PINN [16] and Deep Ritz [5] where the Dirichlet boundary condition is enforced solely through penalty functions and thus changes the true solution, the proposed method is able to address the boundary conditions without penalty functions and it gives the correct true solution even when penalty functions are added, thanks to the semigroup operator. For eigenvalue problems, a primal-dual method is proposed, efficiently resolving the constraint with a simple scalar dual variable and resulting in a faster algorithm compared with the BSDE solver [8] in certain problems such as the eigenvalue problem associated with the linear Schrödinger operator. Numerical results are provided to demonstrate the performance of the proposed methods.

1. Introduction

Central to the discipline of applied mathematics is the problem of numerically solving partial differential equations (PDEs), among which the high dimensional problems are particularly challenging due to the "curse of dimensionality", a phenomenon that the computational complexity of certain algorithms increases exponentially with the dimension. A more challenging problem is the eigenvalue problem, which is closely related with the PDE problem and suffers from the curse of dimensionality as well. In this paper we limit the discussion to second-order linear PDEs and related eigenvalue problems.

In recent years, deep learning methods have experienced great success across a wide range of domains such as image recognition [9,14], natural language processing [3,7], molecular dynamics simulation [10,17], and protein structure prediction [1]. One reason behind this success is that neural network models are good approximators for high-dimensional functions

---

that can be trained efficiently in most cases. Leveraging on this property of the neural network, a myriad of data-driven methods have been proposed for solving high-dimensional PDEs and eigenvalue problems, for example, see [4,5,15,16].

In a recent paper [15], the semigroup operator of the differential operator is used to rewrite the variational form, which frees the algorithm from calculations of any mixed second-order derivative and automatically handles the boundary conditions. In this paper, we extend this method to more general elliptic PDEs and also derive a primal-dual method for solving the corresponding eigenvalue problems.

1.1. Background and related work

For high-dimensional PDEs, Monte-Carlo methods using Feynman-Kac formulas can be applied to obtain the value of the approximate solution at a given location. However, satisfactory solutions should provide information of not only the values on a finite number of points, but also of the entire landscape.

Monte Carlo methods have also been widely applied to the eigenvalue problems. Among various approaches, the variational Monte Carlo method (VMC) and the diffusion Monte Carlo method (DMC) are two most well-known examples that have been thoroughly investigated in the context of quantum mechanics, see for example [6]. The idea of VMC is to parameterize the wave function and minimize the energy of the system with respect to the parameters, where the energy is expressed as an expectation with respect to the probability distribution given by the squared modulus of the wave function, and is numerically computed via the Monte Carlo method. DMC utilizes the imaginary-time Schrödinger equation, whose solution can be represented by a convolution with respect to the Green's function and can thus be evaluated by Monte Carlo simulations. Since the imaginary-time Schrödinger equation is a linear differential equation, the component of the lowest energy eigenfunction remains and other components vanish as the time goes to infinity, and the wave function of the ground state can be obtained.

For neural network based approaches, the general idea is to approximate the solution with a neural network, and then train the neural network to minimize a loss built either from a variational form of the PDE or from a norm of the residue of an equivalent equation of the original PDE. Mostly related to the current work, [11,15] are concerned with the high dimensional PDEs describing the committor function in the transition path theory, which is a second-order elliptic equation with a specific kind of Dirichlet boundary conditions.

In [4], the backward stochastic differential equation (BSDE) method forms the equivalent equation using a BSDE, and the residue norm of the equivalent fixed-point equation is minimized. In some cases the equivalent variational problem has been established, and we only need to directly apply the neural network parameterization. For example, in [2], a neural network with one hidden layer is used to give the trial functions for the VMC method, and the gradient function needed in the optimization is also evaluated by Monte Carlo method. In [5], a ResNet structure is used to parameterize the approximate solution, and the optimization problem is obtained from the variational formulation of elliptic PDEs, which is then solved by stochastic gradient descent (SGD) methods.

In [8], the authors extend the BSDE approach to solve the eigenvalue problem with a second-order elliptic operator. The differential equation is rewritten as a fixed-point equation with the help of the corresponding semi-group operator as in the BSDE method. By Itô's formula, the semi-group operator is represented by a stochastic integral. After that, the numerical solution is obtained by minimizing the loss defining as the L^2 norm of the fixed-point equation residue. The L^2 constraint on the eigenfunction is implemented by dividing the L^2 norm in each batch during training.

Since both the proposed method and the BSDE method involves semigroups of diffusion processes, we remark that there are several major difference between the proposed method and the BSDE method. Since the BSDE method has many variants, here we take the version in [8] to avoid ambiguity. Firstly, the BSDE method usually requires two neural networks, one for the approximate solution and another for its gradient, while our method only needs a single neural network for the approximate solution. Secondly, the semigroup used in [8] is not the one that corresponds to the second-order differential parameter. In particular, it depends on λ, while in our method the semigroup does not depend on λ. Finally, in the method of [8], the eigenvalue is also a parameter that needs to be optimized, while in our method, it does not appear as a variable of the optimization problem but can be computed easily after solving for the eigenfunction.

1.2. Contributions and contents

The two major contributions of our approach are

- The semigroup formulation removes the need of calculating any mixed second-order derivatives, and it treats the Dirichlet boundary conditions naturally. Compared with the method proposed in [15], the method proposed in this paper applies to problems with non-zero right-hand-side term. Compared with other popular neural network based PDE solvers such as the Deep Ritz method [5] and PINN method [16] where the Dirichlet boundary condition is treated solely through additive penalty functions, the proposed method is capable of addressing the Dirichlet boundary condition without penalty functions, and the ground truth remains unchanged even when penalty functions are added.
- We extend the semigroup method also to the eigenvalue problem with a primal dual algorithm, which is able to efficiently enforce the constraint on the L^2 norm of the approximate solution. In the numerical comparison with the BSDE

method proposed in [8], the proposed method is shown to have better performance in certain aspects. For example, less running time is needed to achieve the same precision in the linear Schrödinger problem.

The rest of the paper is organized as follows. Section 2 describes the semigroup approach for the second order elliptic PDEs. Section 3 discusses the primal-dual approach for the eigenvalue problems. Finally, numerical results are reported in Section 4.

2. Elliptic PDEs

Consider the following second order elliptic equation:

$$-\nabla \cdot (a(x)\nabla u(x)) = f(x), \quad x \in \Omega \tag{1}$$

where the coefficient $a(x)$ is uniformly bounded above zero. Here, we consider two types of boundary conditions: the Dirichlet boundary condition:

$$u(x) = r(x), \quad x \in \partial\Omega, \tag{2}$$

and the periodic boundary condition:

$$u(x) = u(x + e_i), \quad x \in \Omega = [0,1)^d, \tag{3}$$

where e_i is the i-th standard basis vector in $\mathbb{R}^d$. When the periodic boundary condition is used, we assume that $a(x)$ and $f(x)$ are both periodic, and we add a further constraint that $\int_\Omega u(x)dx = 0$, so that the solution is unique. Otherwise for any solution u, $u + C$ is also a solution for an arbitrary constant C.

2.1. Semigroup formulation

We define $V(x) = -\log(a(x))$ and let X_t be the solution to the stochastic differential equation (SDE)

$$dX_t = -\nabla V(X_t)dt + \sqrt{2}dW_t, \quad X_0 = x, \tag{4}$$

where W_t is the standard d-dimensional Brownian motion. For a fixed small time step $\delta > 0$, we define the operator P as follows:

$$(Pu)(x) := \mathbb{E}^x \left(u\left(X_{\tau \wedge \delta}\right)\right), \tag{5}$$

where $\mathbb{E}^x$ is the expectation taken with respect to the law of the process (4), and $\tau = \infty$ if the periodic boundary condition is used, and if the Dirichlet boundary condition is used, $\tau = \tau_{\partial\Omega}$ is defined as the hitting time of $\partial\Omega$. By Dynkin's formula, for the solution u of the equation (1) we have

$$Pu(x) = u(x) + \mathbb{E}^x \int_0^{\tau \wedge \delta} \mathcal{A}u\left(X_s\right) ds = u(x) - \mathbb{E}^x \int_0^{\tau \wedge \delta} \frac{f}{a}\left(X_s\right) ds, \quad \forall x \in \Omega, \tag{6}$$

where $\mathcal{A} = \Delta - \nabla V \cdot \nabla$ is the infinitesimal generator, and thus for the solution u of the PDE (1), we have $\mathcal{A}u(x) = -f(x)/a(x)$. Following [15], Pu can be decomposed into two parts as follows:

$$(Pu)(x) = \mathbb{E}^x \left(u\left(X_{\tau \wedge \delta}\right)\right) = \mathbb{E}^x \left(u\left(X_\delta\right)\mathbf{1}_{\{\delta < \tau\}}\right) + \mathbb{E}^x \left(r\left(X_\tau\right)\mathbf{1}_{\{\delta \geq \tau\}}\right). \tag{7}$$

We denote the first part of (7) as

$$(P^i u)(x) \equiv \mathbb{E}^x \left(u(X_{\tau \wedge \delta})\mathbf{1}_{\{\delta < \tau\}}\right) = \mathbb{E}^x \left(u(X_\delta)\mathbf{1}_{\{\delta < \tau\}}\right), \tag{8}$$

where the superscript i stands for the interior contribution and the second part of (7) as

$$(P^b r)(x) \equiv \mathbb{E}^x \left(r(X_{\tau \wedge \delta})\mathbf{1}_{\{\delta \geq \tau\}}\right) = \mathbb{E}^x \left(r(X_\tau)\mathbf{1}_{\{\delta \geq \tau\}}\right), \tag{9}$$

where the superscript b stands for the boundary contribution. As mentioned earlier, when the periodic boundary condition is used, $\tau = \infty$, so $\mathbf{1}_{\{\delta \geq \tau\}} = 0$, and thus P^b becomes a zero operator. In this case, $P^b r$ is a zero function for any function r. In order to give a uniform formulation for both types of boundary conditions, we set $r(x) = 0$ when the periodic boundary condition is used, which has no effect other than making $P^b r$ well defined, and any other function can be used. With these operators, (6) can be rewritten succinctly as

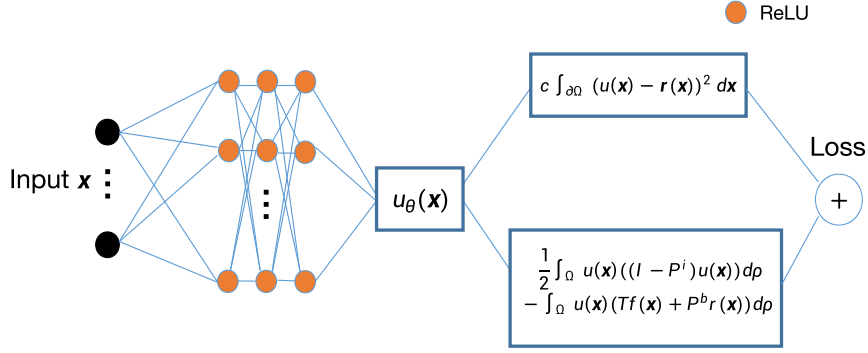$$(I - P^i)u(x) - (P^b r)(x) - (Tf)(x) = 0, \tag{10}$$

3

Fig. 1. An example of the neural network architecture and the corresponding loss when using the Dirichlet boundary condition.

where $(Tf)(x) = \mathbb{E}^x \int_0^{\tau \wedge \delta} \frac{f}{a}(X_s)\, ds$. This equation can be reformulated as the following variational problem

$$\min_u \frac{1}{2} \int_\Omega u(x) \left((I - P^i)u(x) \right) \rho(x) dx - \int_\Omega u(x)(P^b r(x) + Tf(x))\rho(x) dx, \tag{11}$$

where $\rho(x) = a(x)/\left(\int_\Omega a(x)dx \right)$. In order to do this, we need a result from [15].

Theorem 1 ([15]). *P^i is a symmetric operator on $L^2_\rho(\Omega)$, in other words, $\langle u, P^i v \rangle_\rho = \langle P^i u, v \rangle_\rho$, where $\langle \cdot, \cdot \rangle_\rho$ denotes the inner product of the Hilbert space $L^2_\rho(\Omega)$.*

One can show using Theorem 1 that the solution to (11) is the same as the solution to (10) in the following way. Assume that u^* is the solution to (11) and η is continuous in Ω with compact support. By plugging $u(x, \epsilon) = u^*(x) + \epsilon \eta(x)$ into (11) and taking derivative with respect to ϵ, we obtain

$$
\begin{aligned}
0 &= \frac{\partial}{\partial \epsilon} \left(\frac{1}{2} \int_\Omega u(x, \epsilon) \left((I - P^i)u(x, \epsilon) \right) \rho(x) dx - \int_\Omega u(x, \epsilon)(P^b r(x) + Tf(x))\rho(x) dx \right) \bigg|_{\epsilon=0} \\
&= \int_\Omega \eta(x) \left((I - P^i)u^*(x) \right) \rho(x) dx - \int_\Omega \eta(x)(P^b r(x) + Tf(x))\rho(x) dx \\
&= \int_\Omega \eta(x) \left((I - P^i)u^*(x) - P^b r(x) - Tf(x) \right) \rho(x) dx,
\end{aligned}
\tag{12}
$$

and thus $(I - P^i)u^*(x) - P^b r(x) - Tf(x) = 0$ since this is true for any η that is continuous in Ω with compact support.

2.2. Neural network approximation

In order to address the curse of dimensionality, the function u in (11) is parameterized with a neural network u_θ, and the optimization problem becomes

$$\min_\theta \frac{1}{2} \int_\Omega u_\theta(x) \left((I - P^i)u_\theta(x) \right) \rho(x) dx - \int_\Omega u_\theta(x)(P^b r(x) + Tf(x))\rho(x) dx. \tag{13}$$

When the Dirichlet boundary condition (2) is used, we solve the following penalized problem to better address the boundary condition, which is not necessary but is shown to be able to improve the performance of the algorithm.

$$\min_\theta \int_\Omega u_\theta(x) \left(\frac{1}{2}(I - P^i)u_\theta(x) - P^b r(x) - Tf(x) \right) \rho(x) dx + c \int_{\partial\Omega} (u_\theta(x) - r(x))^2 d\mu(x), \tag{14}$$

where $\mu(x)$ is a probability measure supported on $\partial\Omega$.

The architecture of the neural network used in this situation is depicted in Fig. 1. We adopt a three layer fully connected neural network with ReLU activation. Compared with [15], we do not have the singularity layer here since we do not have a temperature parameter and the singularities that appear in the situation of extremely high and extremely low temperatures no longer exist.
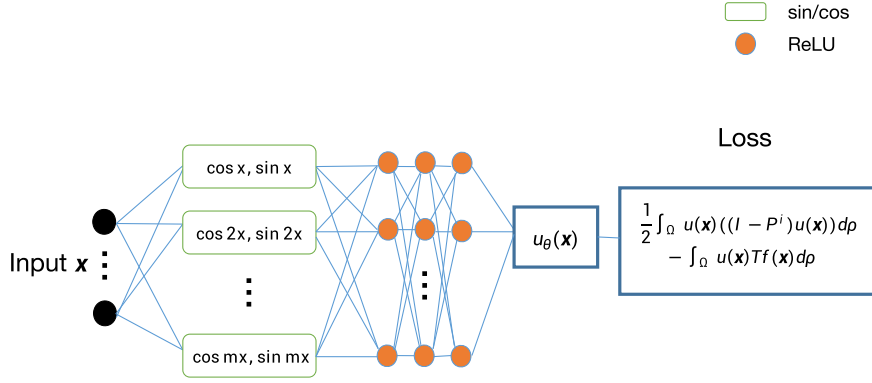
Fig. 2. An example of the neural network architecture and the corresponding loss when using the periodic boundary condition.

When the periodic boundary condition (3) is used, we adopt the following neural network architecture to address the boundary condition with the help of the trigonometric basis, where m is a hyperparameter of the neural network. Since the boundary condition is already treated by the neural network architecture, we do not need the penalty term in this situation.

In the implementation of the optimization algorithm, we also need the derivative of the integrals in (13). By the symmetry of P^i in $L^2_\rho(\Omega)$, the derivative is

$$\int_\Omega \nabla_\theta u_\theta(x) \left((I - P^i) u_\theta(x) \right) \rho(x) \mathrm{d}x - \int_\Omega \nabla_\theta u_\theta(x) (P^b r(x) + Tf(x)) \rho(x) \mathrm{d}x. \tag{15}$$

With the help of a random variable $X \sim \rho$, the derivative can be further transformed into:

$$\mathbb{E}_{X \sim \rho} \nabla_\theta u_\theta(X) \left((I - P^i) u_\theta(X) - (P^b r(X) + Tf(X)) \right). \tag{16}$$

An unbiased estimator for (16) is thus

$$\nabla_\theta u_\theta(X) \left(u_\theta(X) - u_\theta(X_\delta) \mathbf{1}_{\{\delta < \tau\}} - r(X_\tau) \mathbf{1}_{\{\delta \geq \tau\}} - \int_0^{\tau \wedge \delta} \frac{f}{a}(X_s) \mathrm{d}s \right), \tag{17}$$

where X_s is the solution of (4) at time s with the initial condition given by $X_0 = X$. X_δ and X_τ are obtained by evaluating X_s at δ and the stopping time τ, respectively. The derivative of the penalty term is

$$2c \int_{\partial\Omega} \nabla_\theta u_\theta(x)(u_\theta(x) - r(x)) \mathrm{d}\mu(x) \tag{18}$$

and an unbiased estimator for the derivative is

$$2c \nabla u_\theta(X)(u_\theta(X) - r(X)), \tag{19}$$

where $X \sim \mu$. Notice that there is no mixed second-order derivative in (17) and (19), unlike [11] or [5], where mixed second-order derivatives $\frac{\partial^2}{\partial\theta\partial x_i}$, $1 \leq i \leq d$ are needed in the training process.

The optimization problem (13) and the penalized problem (14) can be solved by applying SGD-type optimization, for example the Adam method in [12]. In the numerical implementation, the integral $\int_0^{\tau \wedge \delta} \frac{f}{a}(X_s) \mathrm{d}s$ can be approximated for example by the Euler-Maruyama scheme (see for example [13]). For other implementation details such as the determination of $\mathbf{1}_{\{\delta \geq \tau\}}$, we follow the method in [15]. The complete algorithm is summarized in Algorithm 1.

3. Primal-dual formulation for eigenvalue problems

In this section we extend the semigroup method described in Section 2 to a primal-dual algorithm for eigenvalue problems. For the eigenvalue problems, a major difference is that the corresponding variational problem has a constraint $\|u\| = 1$. In [8], the authors divide the neural network approximate solution by a normalization factor when calculating the training loss, which is similar with the Batch Normalization technique, except that the normalization factor is computed with an auxiliary batch rather than the original batch. Here we propose a primal-dual method that handles the constraint via a scalar Lagrange multiplier. As a result, the time complexity of solving the eigenvalue problem is only marginally higher than that of solving the corresponding PDE.

Algorithm 1 Semigroup method for the elliptic PDE (1).

Require: batch size B and $\tilde{B}$, total number of iterations T_{iter}, time step δ, learning rate η_t, the penalty coefficient c used in the Dirichlet case, the level m of trigonometric bases used in the periodic case.

1: Initialize the neural network u_θ.
2: **for** $t = 1, .., T_{\text{iter}}$ **do**
3:     Sample a batch of data from the distribution ρ with size B:

$$X_1, X_2, \ldots, X_B \sim \rho.$$

4:     For each X_k ($1 \le k \le B$), sample $X_{k,\delta}$ according to the SDE (4):

$$X_{k,\delta} = X_k - \nabla V(X_k)\delta + \sqrt{2}W_\delta,$$

and when the periodic boundary condition is used, move $X_{k,\delta}$ into Ω by translating an integer multiple of the period.
5:     For each X_k, decide the value of $\mathbf{1}_{\{\delta<\tau\}}$ by

$$\mathbf{1}_{\{\delta<\tau\}} = 1 \text{ if } X_{k,\delta} \in \Omega.$$

6:     For each X_k such that $\mathbf{1}_{\{\delta<\tau\}} = 0$, let X_τ be the intersection of $\partial\Omega$ and the line segment $X_k X_{k,\delta}$.
7:     Compute the gradient in (17).
8:     If the Dirichlet boundary condition is used, sample a batch of data $\{\tilde{X}_j\}_{j=1}^{\tilde{B}}$ from the distribution μ, and compute the gradient of the penalty term by (18).
9:     Update the neural network parameters θ via the Adam method with learning rate η_t (other hyper-parameters in the Adam method are set as the default values).
10: **end for**
11: If the periodic boundary condition is used, then sample a batch of data $\{\tilde{X}_j\}_{j=1}^{\tilde{B}}$ from the uniform distribution on Ω, and subtract $\frac{1}{\tilde{B}}\sum_{j=1}^{\tilde{B}} u_\theta(\tilde{X}_j)$ from u_θ.

Consider $\Omega = [0, 1)^d$ and define C_{per}^∞ as the space of smooth functions on Ω satisfying the periodic boundary condition (3). Consider a symmetric elliptic operator $L = -\sum_{i,j=1}^d \frac{\partial}{\partial x_j}\left(A_{ij}\frac{\partial}{\partial x_i}\right) + V$, where $V \in C_{\text{per}}^\infty(\Omega)$, $A_{ij} \in C_{\text{per}}^\infty(\Omega)$, $1 \le i, j \le d$ and $A_{ij} = A_{ji}$, $1 \le i, j \le d$. We also assume that A is uniformly elliptic, i.e., $\sum_{i,j=1}^d A_{ij}(x)\xi_i\xi_j \ge \alpha\|\xi\|^2$ for some constant $\alpha > 0$ and any $x, \xi \in \Omega$. By variational principle we know that for the first eigenpair (u, λ), the eigenvalue problem

$$Lu = \lambda u, \quad \|u\|_{L^2} = 1, \tag{20}$$

with the periodic boundary condition (3) is equivalent to the variational form:

$$u = \operatorname*{argmin}_{u \in H_{\text{per}}^1(\Omega), \|u\|_{L^2}=1} \frac{1}{2}\int_\Omega u(x)Lu(x)\mathrm{d}x, \tag{21}$$

where $H_{\text{per}}^1(\Omega)$ consists of the functions satisfying the periodic boundary condition (3) in $H^1(\Omega)$. For simplicity of the notations, we omit the function space when there is no ambiguity.

3.1. Resolving the constraint

The main obstacle in implementing the formulation (21) lies in the constraint $\|u\| = 1$. We propose to handle the constraint with a multiplier term, and reconstruct (21) as the minimax formulation:

$$\min_u \max_g \frac{1}{2}\int_\Omega u(x)Lu(x)\mathrm{d}x + \frac{g}{2}(\|u\|_{L^2}^2 - 1), \tag{22}$$

where g is the Lagrange multiplier. The equivalence between (21) and (22) can be seen by maximizing over g in (22) and thus we have removed the explicit constraint in (21). We adopt the neural network parameterization as in Section 2 and obtain the following optimization problem:

$$\min_\theta \max_g \frac{1}{2}\int_\Omega u_\theta(x)Lu_\theta(x)\mathrm{d}x + \frac{g}{2}(\|u_\theta\|_{L^2}^2 - 1). \tag{23}$$

Let us define

$$E(\theta, g) = \frac{1}{2}\int_\Omega u_\theta(x)Lu_\theta(x)\mathrm{d}x + \frac{g}{2}(\|u_\theta\|_{L^2}^2 - 1). \tag{24}$$

Taking derivatives with respect to θ and g gives

$$\frac{\partial E}{\partial \theta} = \int_\Omega \nabla_\theta u_\theta(x)(L + g)u_\theta(x)\mathrm{d}x,$$

$$\frac{\partial E}{\partial g} = \frac{1}{2}(\|u_\theta\|^2_{L^2} - 1), \tag{25}$$

where we have utilized the symmetry of L.

3.2. Scaling the Lagrange multiplier

In the optimization problem (23), it is important that the Lagrange multiplier has an appropriate scale, since otherwise the problem would be ill-conditioned. If we introduce a scaling parameter c for the dual variable g, then the optimization problem becomes

$$\min_\theta \max_g \frac{1}{2}\int_\Omega u_\theta(x)Lu_\theta(x)\mathrm{d}x + \frac{cg}{2}(\|u_\theta\|^2_{L^2} - 1), \tag{26}$$

or equivalently,

$$E = \frac{1}{2}\int_\Omega u_\theta(x)Lu_\theta(x)\mathrm{d}x + \frac{cg}{2}(\|u_\theta\|^2_{L^2} - 1), \tag{27}$$

and the derivatives are scaled accordingly:

$$\frac{\partial E}{\partial \theta} = \int_\Omega \nabla_\theta u_\theta(x)(L + cg)u_\theta(x)\mathrm{d}x,$$

$$\frac{\partial E}{\partial g} = \frac{c}{2}(\|u_\theta\|^2_{L^2} - 1). \tag{28}$$

If c is too large, then in the optimization process the first term in E is neglected by the neural network. If c is too small, then the constraint is not well-enforced. Therefore in the implementation, c should be properly chosen such that the two terms in E are balanced.

Based on (28), we arrive at the primal-dual scheme

$$\dot{\theta} = -\int_\Omega \nabla_\theta u_\theta(x)(L + cg)u_\theta(x)\mathrm{d}x,$$

$$\dot{g} = \frac{c}{2}(\|u_\theta\|^2_{L^2} - 1). \tag{29}$$

In what follows, we focus on the Schrödinger operator $L = -\Delta + V$ and consider the problem of finding the first eigenpair (which corresponds to the ground state) with periodic boundary conditions. The minimax formulation (26) becomes

$$\min_u \max_g \frac{1}{2}\int_\Omega |\nabla u(x)|^2\mathrm{d}x + \frac{1}{2}\int_\Omega V(x)|u(x)|^2\mathrm{d}x + \frac{cg}{2}(\|u\|^2_{L^2} - 1). \tag{30}$$

Two stochastic schemes for implementing (29) are discussed below.

3.3. Scheme I

By replacing the operator $\frac{\Delta}{2}$ with the semigroup approximation, we can reformulate (30) as

$$\min_u \max_g \frac{1}{\delta}\int_\Omega u(x)(u(x) - \mathbb{E}(u(x + W_\delta)))\mathrm{d}x + \frac{1}{2}\int_\Omega V(x)|u(x)|^2\mathrm{d}x + \frac{cg}{2}(\|u\|^2 - 1), \tag{31}$$

where W_δ is the standard Brownian motion. This replacement can be justified as follows. By Dynkin's formula, we have

$$\frac{1}{\delta}(u(x) - \mathbb{E}u(x + W_\delta)) = \mathbb{E}\frac{1}{\delta}\int_0^\delta -\frac{1}{2}\Delta u(x + W_s)\mathrm{d}s.$$

Let $\varphi = -\frac{1}{2}\Delta u$. If φ is bounded and sufficiently smooth, then by Fubini's theorem,

$$\mathbb{E}\frac{1}{\delta}\int_0^\delta \varphi(x+W_s)\mathrm{d}s = \frac{1}{\delta}\int_0^\delta \mathbb{E}\varphi(x+W_s)\mathrm{d}s = \frac{1}{\delta}\int_0^\delta \mathbb{E}\Big[\varphi(x)+\nabla\varphi(x)\cdot W_s + O(\|W_s\|^2)\Big]\mathrm{d}s$$

$$= \frac{1}{\delta}\int_0^\delta \Big[\varphi(x)+O(s)\Big]\mathrm{d}s = \varphi(x)+O(\delta),$$

which shows that $\frac{1}{\delta}(u(x)-\mathbb{E}u(x+W_\delta)) = -\frac{1}{2}\Delta u(x) + O(\delta)$ and justifies the replacement in (31). Consider a uniform random variable X on Ω, the problem (31) can be written as

$$\min_u \max_g \frac{1}{\delta}\mathbb{E}u(X)(u(X)-u(X+W_\delta)) + \frac{1}{2}\mathbb{E}V(X)|u(X)|^2 + \frac{cg}{2}(\|u\|^2-1). \tag{32}$$

Since X is uniform on Ω, the process $X+W_\delta$ is reversible and

$$\langle u, \tilde{P}v\rangle = \langle \tilde{P}u, v\rangle, \tag{33}$$

where $\langle u, v\rangle = \mathbb{E}u(X)v(X)$ and $\tilde{P}u(x) = \mathbb{E}u(x+W_\delta)$. By this symmetry, a gradient descent scheme similar with (29) can be derived

$$\dot{\theta} = -\mathbb{E}\nabla_\theta u_\theta(X)\left(\frac{1}{\delta}(u_\theta(X)-u_\theta(X+W_\delta)) + cgu_\theta(X)\right),$$

$$\dot{g} = \frac{c}{2}\mathbb{E}(u_\theta(X)^2-1). \tag{34}$$

Unbiased estimators for the expectations in (34) are therefore

$$\nabla_\theta u_\theta(X)(\frac{1}{\delta}(u_\theta(X)-u_\theta(X+W_\delta)) + cgu_\theta(X)), \tag{35}$$

and

$$\frac{c}{2}(u_\theta(X)^2-1), \tag{36}$$

where X is uniform in Ω.

3.4. Scheme II

In (32), we have used $\frac{1}{\delta}(u(x)-\mathbb{E}u(x+W_\delta))$ to approximate $-\frac{\Delta}{2}u(x)$. The effectiveness of this approximation relies on the spatial symmetry of W_δ, which is only true in distribution. In other words, after using the unbiased estimators given in (35) with samples of X and W_δ, the symmetry can be impacted. This problem can be addressed by starting by approximating the first term of (30) with

$$\frac{1}{2}\int_\Omega |\nabla u(x)|^2\mathrm{d}x = \frac{1}{2\delta}\left(\int_\Omega \mathbb{E}|u(x)-u(x+W_\delta)|^2\mathrm{d}x + o(\delta)\right). \tag{37}$$

The formulation above can be obtained, for example, by plugging $u(x+W_\delta) = u(x)+\nabla u(x)\cdot W_\delta + O(\|W_\delta\|^2)$ into the expectation, which gives

$$\mathbb{E}|u(x)-u(x+W_\delta)|^2 = \mathbb{E}|\nabla u(x)\cdot W_\delta|^2 + o(\delta) = \mathbb{E}(W_\delta^\top \nabla u(x)(\nabla u(x))^\top W_\delta) + o(\delta)$$

$$= \mathbb{E}(\mathrm{tr}(W_\delta^\top \nabla u(x)(\nabla u(x))^\top W_\delta)) + o(\delta) = \mathbb{E}(\mathrm{tr}(\nabla u(x)(\nabla u(x))^\top W_\delta W_\delta^\top)) + o(\delta)$$

$$= \mathrm{tr}(\nabla u(x)(\nabla u(x))^\top \mathbb{E}(W_\delta W_\delta^\top)) + o(\delta) = \delta\,\mathrm{tr}(\nabla u(x)(\nabla u(x))^\top) + o(\delta) \tag{38}$$

$$= \delta|\nabla u(x)|^2 + o(\delta).$$

Notice that compared with the first term in (31), the right hand side of (37) has an extra factor 1/2. The appearance of this factor is natural since

$$\frac{1}{2}\mathbb{E}|u(x)-u(x+W_\delta)|^2 = u(x)(u(x)-\mathbb{E}u(x+W_\delta)) + \frac{1}{2}(\mathbb{E}u(x+W_\delta)^2-u(x)^2),$$

so the additional factor 1/2 here can be viewed as a result of replacing $\frac{1}{2}u(x)^2$ with $\frac{1}{2}\mathbb{E}u(x+W_\delta)^2$ in (31).

With this approximation and the uniform random variable X on Ω, (30) can be written as

$$\min_u \max_g \mathbb{E}\left(\frac{1}{2\delta}|u(X) - u(X + W_\delta)|^2 + \frac{1}{2}V(X)|u(X)|^2 + \frac{cg}{2}(u(X)^2 - 1)\right). \tag{39}$$

For this problem, the gradient descent scheme is

$$\dot{\theta} = -\mathbb{E}\left[\frac{1}{\delta}(\nabla_\theta u_\theta(X) - \nabla_\theta u_\theta(X + W_\delta))(u_\theta(X) - u_\theta(X + W_\delta))\right.$$

$$\left. + \nabla_\theta u_\theta(X)(V(X)u_\theta(X) + cgu_\theta(X))\right], \tag{40}$$

$$\dot{g} = \frac{c}{2}\,\mathbb{E}(u_\theta(X)^2 - 1).$$

Unbiased estimators for the expectations are therefore

$$\frac{1}{\delta}(\nabla_\theta u_\theta(X) - \nabla_\theta u_\theta(X + W_\delta))(u_\theta(X) - u_\theta(X + W_\delta)) + \nabla_\theta u_\theta(X)(V(X)u_\theta(X) + cgu_\theta(X)) \tag{41}$$

and

$$\frac{c}{2}(u_\theta(X)^2 - 1). \tag{42}$$

It is clear that even after replacing the expectation with the unbiased estimators, the problem is still symmetric.

Algorithm 2 Semigroup methods for the eigenvalue problem.

Require: default value of the dual variable g_{default}, batch size B and $\tilde{B}$, total number of iterations T_{iter}, time step δ, learning rate $\eta_t, \tilde{\eta}_t$, scaling factor c, the level m of trigonometric bases used.
1: Initialize the neural network u_θ.
2: **for** $t = 1, .., T_{\text{iter}}$ **do**
3:     Sample a batch of data $\{X_k\}_{k=1}^B$ uniformly in Ω.
4:     Compute the gradient in (35) or (41).
5:     Sample a batch of data $\{\tilde{X}_j\}_{j=1}^{\tilde{B}}$ uniformly in Ω.
6:     Compute $\epsilon_t = \max\left(\frac{1}{\tilde{B}}\sum_j u_\theta(\tilde{X}_j)^2 - 1, 1\right)$.
7:     **if** $t = 1$ **or** $\epsilon_t \epsilon_{t-1} < 0$ **then**
8:         Set $g = \text{sign}(\epsilon_t)g_{\text{default}}$.
9:     **else**
10:         Update the dual variable by $g \leftarrow g + \tilde{\eta}_t \frac{c\epsilon_t}{2}$.
11:     **end if**
12:     Update the neural network parameters by the Adam method with learning rate η_t (other hyper-parameters in the Adam method are set as the default values).
13: **end for**
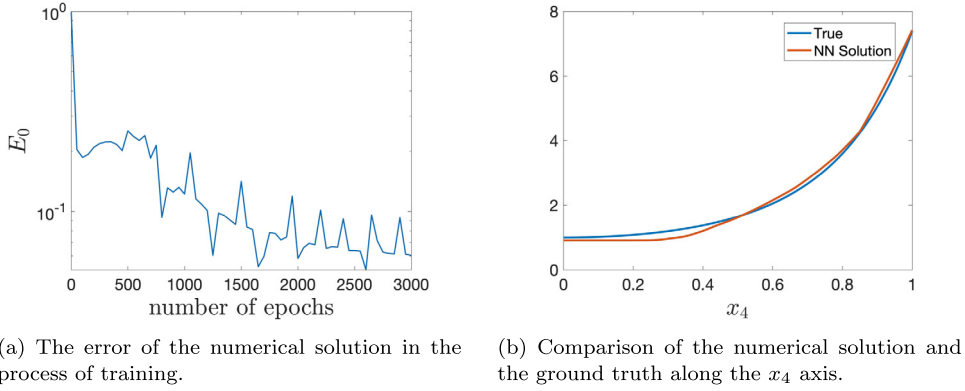

3.5. Implementation of the multiplier term

Intuitively, the dual variable should be able to give the correct preference for u_θ. Specifically, when $\|u_\theta\| > 1$, g should be positive, and when $\|u_\theta\| < 1$, g should be negative. In the implementation, this is enforced in each step of the (stochastic) gradient update. More specifically, in step t we estimate $\|u_\theta\| - 1$ using a batch of data with size $\tilde{B}$, and denote the estimator as ϵ_t. If $\epsilon_t \epsilon_{t-1} < 0$, then we reset the multiplier g as $\text{sign}(\epsilon_t)g_{\text{default}}$, where $g_{\text{default}} > 0$ is a hyperparameter. In this way, g will be positive when $\|u_\theta\| - 1$ changes from negative to positive, and negative when $\|u_\theta\| - 1$ changes from positive to negative, which leads $\|u_\theta\|$ towards the correct update direction.

We also set the estimation of the gradient of g to be $c/2$ if it is larger than $c/2$. This is because $\frac{c}{2}\,\mathbb{E}(u_\theta(X)^2 - 1)$ is lower bounded by $-c/2$ but it has no upper bound. In practice setting a symmetric upper bound usually gives a better performance. We summarize the implementation details in Algorithm 2, where we adopt the neural network architecture in Fig. 2.

4. Numerical experiments

4.1. Elliptic PDEs

In this section, we verify the effectiveness of Algorithm 1. We measure the error of the numerical solutions by $E_0 = \|u_\theta - u^*\|_{L^2_\rho(\Omega)}/\|u^*\|_{L^2_\rho(\Omega)}$, where u^* is the ground truth. In the numerical examples, E_0 is estimated on a test set of size 1×10^4.

(a) The error of the numerical solution in the process of training.

(b) Comparison of the numerical solution and the ground truth along the x_4 axis.

Fig. 3. The convergence of the error E_0 and the comparison of the NN solution with the true solution. (a): The error $\|u_\theta - u^*\|_{L^2_\rho(\Omega)}/\|u^*\|_{L^2_\rho(\Omega)}$ every 50 epochs during the training process. (b): Comparison of the NN solution with the true solution on a randomly chosen coordinate axis. Blue: The true solution; Red: The NN solution. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

In the first numerical example, we set $a(x) = e^{-2\|x\|^2}$ and $f(x) = -4d$ in (1), and assume the Dirichlet boundary condition (2) with $r(x) = e^2$ on the boundary of the domain $\Omega = B(0, 1)$, the unit ball in $\mathbb{R}^d$. In other words, we consider the following problem:

$$\Delta u + 4x \cdot \nabla u = -4de^{2\|x\|^2}, \quad x \in B(0, 1),$$
$$u = e^2, \quad x \in \partial B(0, 1).$$

In this problem, the unique solution is $u^*(x) = e^{2\|x\|^2}$, which can be obtained by direct calculations. Numerical experiment is implemented for the 10-dimensional case. We use 4.0×10^6 samples to train the model, which is a three-layer fully-connected network with 120 neurons in each layer. We set $\delta = 0.0001$, $c = 0.8$, and train the model for 3000 epochs using the Adam algorithm in [12] with learning rate 0.001 and 70000 samples in each batch. The final error is $E_0 = 0.060$, which is computed on a test set with size 1.0×10^5. The result is visualized in Fig. 3. In Fig. 3(a), the error is recorded every 50 epochs, from which we can see that the error converges through the training process. Since the true solution is a radial function, we can visualize the difference between the NN solution and the true solution on a randomly chosen coordinate axis. In Fig. 3(b), it can be seen that the NN solution shows good accordance with the ground truth on the positive x_4 axis. From this numerical test, we have verified that the proposed method is effective when the right-hand-side term is non-zero, which is a case not covered in [15].

In the second numerical example, we consider a problem with the periodic boundary condition (3). The parameters are

$$a(x) = \exp\left(-\sum_{i=1}^{d} \cos(2\pi x_i)\right),$$

$$f(x) = 2\pi^2 \exp\left(-\sum_{i=1}^{d} \cos(2\pi x_i)\right)\left(\sum_{i=1}^{d} (2\sin(2\pi x_i) - \sin(4\pi x_i))\right). \tag{43}$$
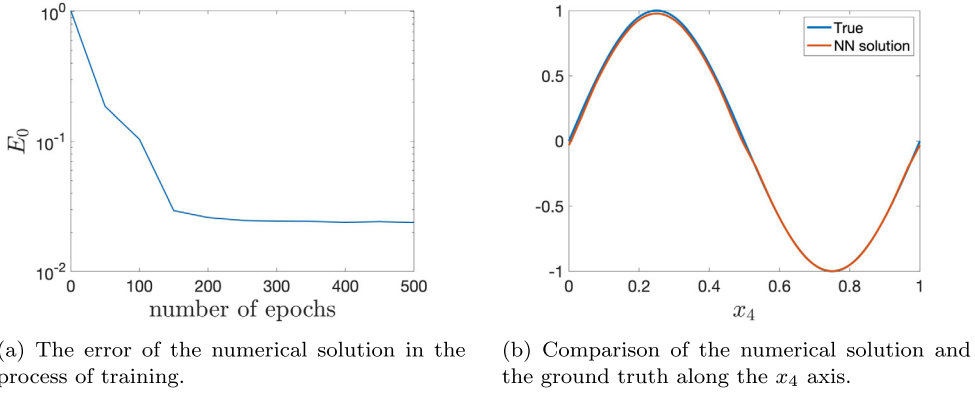
By direct calculation, we see that the exact solution is

$$u(x) = \sum_{i=1}^{d} \sin(2\pi x_i). \tag{44}$$

Numerical experiment is carried out for $d = 10$. The neural network structure is described in Fig. 2, in which we take $m = 1$ and set the width of the network as 12. We take $\delta = 0.0001$, $\tilde{B} = 2 \times 10^5$. In the implementation of the Adam algorithm, the batchsize is set as $B = 70000$ and the learning rate is set as $\eta = 0.001$. A training set with 1.0×10^7 samples to train the model. After a training process of 500 epochs, the final error $\|u_\theta - u^*\|_{L^2_\rho(\Omega)}/\|u^*\|_{L^2_\rho(\Omega)}$ is 0.024. The evolution of the precision of the approximate solution is demonstrated in Fig. 4(a), which shows a rapid convergence of the numerical solution. In order to compare the numerical solution with the ground truth, we plot the two functions along a randomly chosen axis in Fig. 4(b), from which we can see that the NN solution shows good accordance with the true solution.

4.2. Eigenvalue problem - Schrödinger operator

In this section, we consider the eigenvalue problem associated with the Schrödinger operator

$$L = -\Delta + V, \tag{45}$$

(a) The error of the numerical solution in the
process of training.

(b) Comparison of the numerical solution and
the ground truth along the x_4 axis.

Fig. 4. The convergence of the error E_0 and the comparison of the NN solution with the true solution. (a): The error $\|u_\theta - u^*\|_{L^2_\rho(\Omega)}/\|u^*\|_{L^2_\rho(\Omega)}$ every 50 epochs during the training process. (b): Comparison of the NN solution with the true solution on a randomly chosen coordinate axis. Blue: The true solution; Red: The NN solution.

and the periodic boundary condition (3). Here V is a potential function

$$V(x) = 4\pi^2 \sum_{i=1}^{d} c_i \cos(2\pi x_i), \tag{46}$$

where $c_i \in [0, 0.2]$ for $1 \le i \le d$. We adopt the same parameters as in [8]. The reference solution u^* and the corresponding eigenvalue λ^* are obtained by the spectral method described in [8], and we measure the error of the numerical solutions by $E_0 = \|u_\theta - u^*\|_{L^2_\rho(\Omega)}/\|u^*\|_{L^2_\rho(\Omega)}$ and the error of the eigenvalue by $E_1 = |\lambda - \lambda^*|/|\lambda^*|$, where λ is obtained via:

$$\frac{1}{\|u^*\|^2_{L^2_\rho(\Omega)}} \left(\frac{2}{\delta} \int_\Omega u_\theta(x)(u_\theta(x) - \mathbb{E}(u_\theta(x + W_\delta)))dx + \int_\Omega V(x)|u_\theta(x)|^2 dx \right),$$

if scheme I is used, and via

$$\frac{1}{\|u^*\|^2_{L^2_\rho(\Omega)}} \left(\frac{1}{\delta} \int_\Omega |u_\theta(x) - \mathbb{E}(u_\theta(x + W_\delta))|^2 dx + \int_\Omega V(x)|u_\theta(x)|^2 dx \right).$$

Numerical tests are performed when $d = 5, 10$. In the numerical results, E_0 is estimated on a test set of size 1×10^4, and E_1 is estimated by averaging 10 estimations on test sets of size 1×10^5. The level m of trigonometric bases used is set as 5.

4.2.1. Scheme I

When adopting the semigroup formulation (31) to solve the 5-dimensional case, 1.2×10^7 samples are used to train the neural network with width 300, with 1.0×10^5 samples used in each iteration. Throughout the training process, the learning rates $\tilde{\eta}_t = \eta_t = 0.0003$. The hyper-parameters g_{default}, c and δ are set to be 1, 70 and 0.0001, respectively. After 400 iterations of training, the final errors of the eigenfunction and eigenvalue are $E_0 = 0.036$ and $E_1 = 0.11$, respectively. The decay of errors is presented in Fig. 5(a), from which we can see that the numerical solutions converge to the reference solution.

For the 10-dimensional problem, we also use 1.2×10^7 samples. A neural network with width 600 is trained learning rates $\tilde{\eta}_t = \eta_t = 0.0005$. In each iteration, 7.0×10^4 samples are randomly chosen from the training set. We set the hyper-parameters g_{default}, c and δ as 4, 40 and 0.0001, respectively. After 400 iterations, the final errors of the eigenfunction and eigenvalue are $E_0 = 0.058$ and $E_1 = 0.07$, respectively. The convergence of the numerical solutions to the reference solution is shown in Fig. 6(a). In both 5d and 10d problems, the constraint $\|u_\theta\| = 1$ is well-enforced at the end of the training process, as shown in Fig. 5(b) and Fig. 6(b).

Since it is difficult to visualize functions in high dimensions, we compare the probability density function of $u_\theta(Z)$ and $u(Z)$, where Z is a uniform random variable on Ω. The probability density function is obtained by performing kernel density estimation on a sample set of size 10000. The comparison of the probability density functions is given in Fig. 7, from which we can conclude that the numerical solutions obtained are in good accordance with the ground truths.

4.2.2. Scheme II

When adopting the semigroup formulation (39), 4.0×10^6 samples are used to train the neural network with width 300 for the 5-dimensional case and 600 for the 10-dimensional case, and 2000 iterations of training are implemented. In each
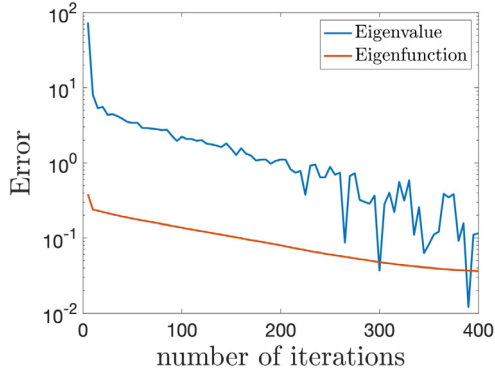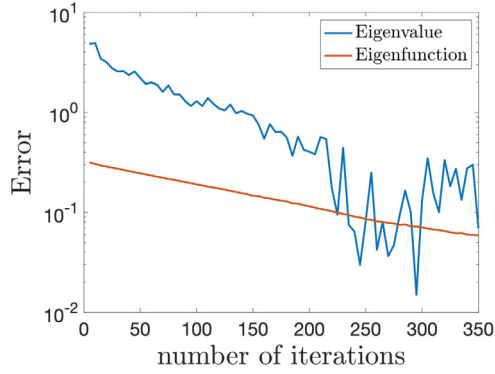
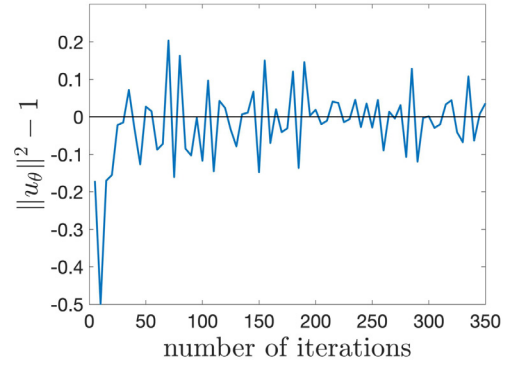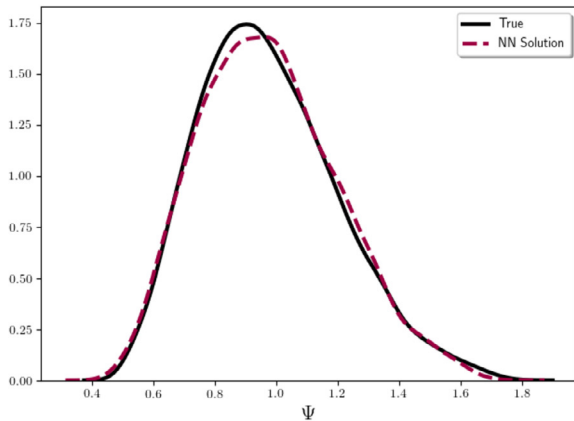(a) The error of the numerical solution in the process of training.
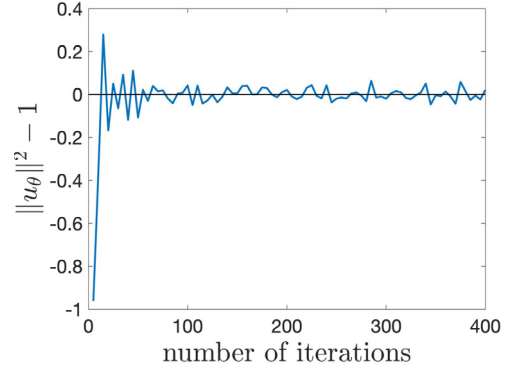
(b) The residue $\|u_\theta\|^2 - 1$ of the constraint in the process of training.

Fig. 5. The convergence of the error of the eigenfunction and eigenvalue and the residue of the constraint $\|u_\theta\|^2 - 1$ in the training process of the 5-dimensional problem with scheme I. Blue curve in (a): Convergence of the error of the approximate eigenvalue; Orange curve in (a): Convergence of the error of the NN eigenfunction.

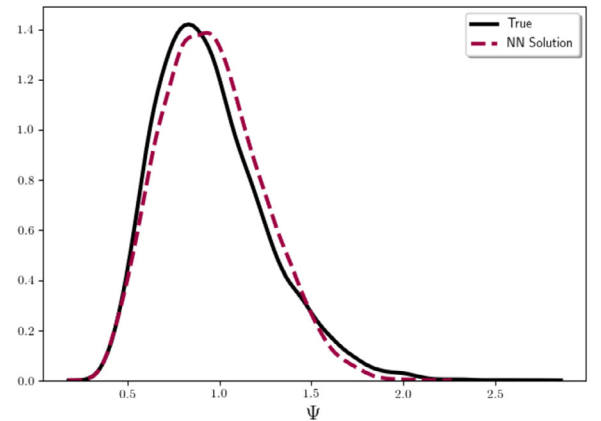(a) The error of the numerical solution in the process of training.

(b) The residue $\|u_\theta\|^2 - 1$ of the constraint in the process of training.

Fig. 6. The convergence of the error of the eigenfunction and eigenvalue and the residue of the constraint $\|u_\theta\|^2 - 1$ in the training process of the 10-dimensional problem with scheme I. Blue curve in (a): Convergence of the error of the approximate eigenvalue; Orange curve in (a): Convergence of the error of the NN eigenfunction.
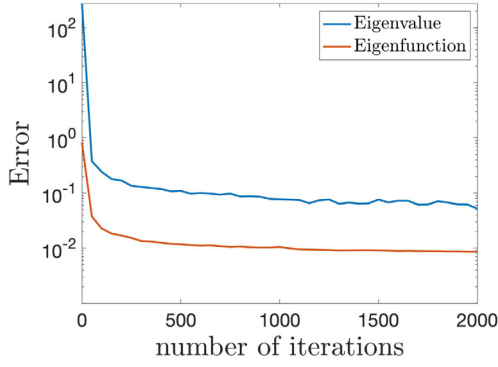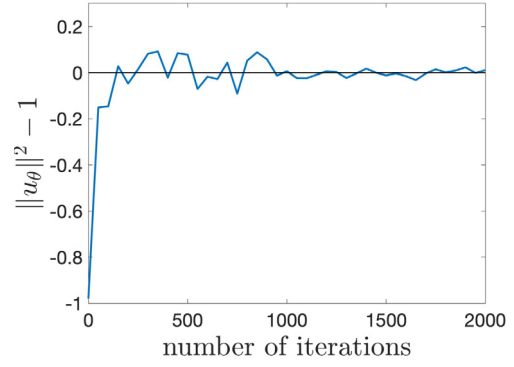

(a) Comparison of the 5d result.

(b) Comparison of the 10d result.

Fig. 7. Comparisons between the NN represented solutions and the ground truths using semigroup scheme I. (a): The 5-dimensional case. (b): The 10-dimensional case.
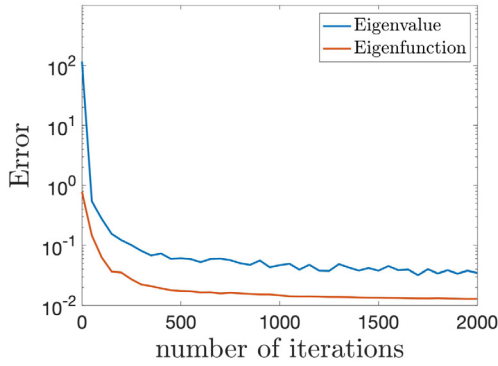
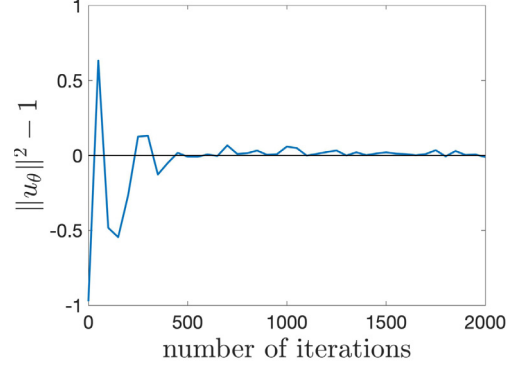(a) The error of the numerical solution in the process of training.

(b) The residue $\|u_\theta\|^2 - 1$ of the constraint in the process of training.

Fig. 8. The convergence of the error of the eigenfunction and eigenvalue and the residue of the constraint $\|u_\theta\|^2 - 1$ in the training process of the 5-dimensional problem with scheme II. Blue curve in (a): Convergence of the error of the approximate eigenvalue; Orange curve in (a): Convergence of the error of the NN eigenfunction.

(a) The error of the numerical solution in the process of training.

(b) The residue $\|u_\theta\|^2 - 1$ of the constraint in the process of training.

Fig. 9. The convergence of the error of the eigenfunction and eigenvalue and the residue of the constraint $\|u_\theta\|^2 - 1$ in the training process of the 10-dimensional problem with scheme II. Blue curve in (a): Convergence of the error of the approximate eigenvalue; Orange curve in (a): Convergence of the error of the NN eigenfunction.

iteration, 1.0×10^4 samples are chosen randomly from the training set. In this problem, the hyper-parameter δ and c are set to be 0.001 and 10 respectively, and g_{default} is set to be 4 for the 5-dimensional case and 1 for the 10-dimensional case. The learning rate for the dual variable $\tilde{\eta}_t$ is set as 0.1 while the learning rate for the primal variable η_t is set as 0.0008 for the first half of the training process and 0.0003 for the second half of the training process.

For the 5-dimensional case and the 10-dimensional case, the final errors of the eigenfunction are 0.0086 and 0.013, respectively, and the final errors of the eigenvalue are 0.052 and 0.035, respectively. The training processes are depicted in Fig. 8 and Fig. 9, respectively. As illustrated in Fig. 8(a) and Fig. 9(a), the numerical solutions of the eigenfunction and eigenvalue converge to the corresponding reference solutions. Compared with the training process using Scheme I, the final errors are much lower and the size of the training set is much smaller, although it takes more iterations to reach the final precision. This can also be verified by comparing Fig. 7 and Fig. 10, since the estimated probability density functions for the numerical solutions are closer to the probability density functions for the reference solutions in Fig. 10. We check that the constraint $\|u_\theta\| = 1$ is well-enforced at the end of the training process in Fig. 8(b) and Fig. 9(b).

Comparison with the method used in [8]. In this part, we briefly compare our numerical results obtained by scheme II with those in [8]. The linear Schrödinger problem in their paper is only different with the problem here by a 2π factor. Specifically, they consider the operator

$$L = -\Delta + V,$$

and the periodic boundary condition (3), with the domain replaced by $[0, 2\pi]^d$, and V replaced by
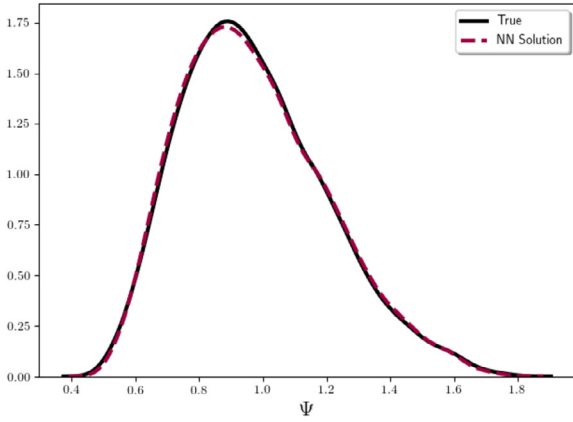
(a) Comparison of the 5d result.
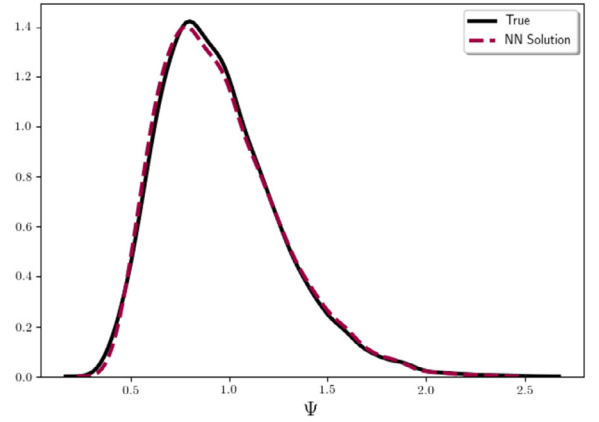
(b) Comparison of the 10d result.

Fig. 10. Comparisons between the NN represented solutions and the ground truths using Scheme II. (a): The 5-dimensional case. (b): The 10-dimensional case.

$$V(x) = \sum_{i=1}^{d} c_i \cos(x_i),$$

where the coefficients $\{c_i\}_{i=1}^{d}$ are the same with the coefficients used here. The L^2 error reported in [8] is calculated after normalizing the solutions with $\int_{\Omega} u^2(x)\mathrm{d}x = |\Omega|$, which is also satisfied by the solutions here since the volume of $[0, 1]^d$ is 1 and we assume $\|u^*\|_{L^2(\rho)} = 1$. Since the L^2 error reported in their paper is invariant under scaling, it is reasonable to compare the numerical results obtained there with the results in this paper. In the following, we divide the errors of the eigenvalue reported in their paper by the true eigenvalues in order to be consistent with the measurement used here.

We test the method in [8] using the source code provided by the authors of [8] and the hyperparameters in [8] on the same machine as we used to test our methods, which is a machine with 4 N1 virtual CPUs on the Google Cloud platform with altogether 26 GB memory and a Tesla K80 GPU. In both the 5-dimensional and 10-dimensional cases, the model is trained for 80000 iterations with 1024 samples used in each iteration. For the 5-dimensional case, the final errors for the eigenvalue and eigenfunction are 0.016 and 0.0098, respectively. For the 10-dimensional case, the final errors for the eigenvalue and eigenfunction are 0.017 and 0.012, respectively. It can be seen that the precision of the eigenfunction is comparable with the precision of eigenfunction in scheme II, while the errors of the eigenvalue are smaller but of the same magnitude as our results.

On the other hand, due to the simplicity of our method, our method enjoys a shorter computation time and requires fewer samples. For the method in [8], the computation time used in the 5-dimensional and 10-dimensional problems are 1.5×10^4 seconds (4.2 hours) and 4.3×10^4 seconds (12 hours), respectively, and altogether 8.0×10^7 samples are used, while for scheme II proposed in this paper the computation time used are 3.0×10^2 seconds (5.0 minutes) and 5.7×10^2 seconds (9.5 minutes), and 4.0×10^6 samples are used. If we consider the numerical solutions obtained by the first 11000 iterations of the method in [8], the errors of the eigenvalue and eigenfunction are 0.087 and 0.035 for the 5d case and 0.097 and 0.059 for the 10d case, which is larger than those in our method, but the time used is still 2.1×10^3 seconds (35 minutes) and 5.9×10^3 seconds (98 minutes) for the 5d and 10d case, respectively, and the number of samples used is still 1.1×10^7, which is much larger than our method. Finally, we mention that other problems such as the problem of finding the second eigenpair are addressed in [8], which is not covered in our paper.

5. Conclusion

In this paper, we present a semigroup method solving high dimensional PDE problems and eigenvalue problems effectively. We have shown numerically the efficiency of the proposed method in problems that have non-zero right-hand-side term with Dirichlet boundary conditions and periodic boundary conditions. In comparison with popular deep PDE solvers such as the Deep Ritz method [5] and PINN method [16], where penalty function is used to enforce the Dirichlet boundary condition and thus changes the solution, the proposed semigroup method addresses the Dirichlet boundary condition without penalty functions, and even when penalty functions are used, the true solution remains the same. Two semigroup schemes are proposed for the eigenvalue problems. With a scalar Lagrange multiplier, these schemes are able to handle the constraint in the eigenvalue problem and obtain accurate solutions. In comparison with established solvers such as the BSDE method [8], the proposed method uses much less computation time to achieve the same precision for certain problem such as the linear Schrödinger problem.

The numerical schemes adopted here are generally first-order schemes. For future work, higher order schemes can be applied to improve the precision. Moreover, importance sampling techniques can be integrated with the proposed method

to facilitate the generation of a training set with high quality. For the PDE problems, we have extended the method in [15] to the case with non-zero right-hand-side term and more general boundary conditions. We point out that it is possible to further generalize the semigroup method to other types of elliptic PDEs. One possible way to proceed is to replace the second-order derivatives using the semigroup operator in, for example, the PINN method ([16]). In this way, it is possible to avoid the calculation of mixed third-order derivatives needed there and to integrate the boundary conditions naturally through the semigroup operator instead of enforcing it solely through a penalty function. We can possibly accelerate the training of the model and alleviate the difficulty in tuning the penalty coefficient by this replacement.

CRediT authorship contribution statement

Haoya Li: Conceptualization, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Lexing Ying:** Conceptualization, Methodology, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Mohammed AlQuraishi, Alphafold at casp13, Bioinformatics 35 (22) (2019) 4862–4865.
[2] Giuseppe Carleo, Matthias Troyer, Solving the quantum many-body problem with artificial neural networks, Science 355 (6325) (2017) 602–606.
[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Bert: pre-training of deep bidirectional transformers for language understanding, arXiv preprint, arXiv:1810.04805, 2018.
[4] Weinan E, Jiequn Han, Arnulf Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Commun. Math. Stat. 5 (4) (2017) 349–380.
[5] Weinan E, Bing Yu, The Deep Ritz Method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[6] W.M.C. Foulkes, Lubos Mitas, R.J. Needs, Guna Rajagopal, Quantum Monte Carlo simulations of solids, Rev. Mod. Phys. 73 (1) (2001) 33.
[7] Alex Graves, Abdel-rahman Mohamed, Geoffrey Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6645–6649.
[8] Jiequn Han, Jianfeng Lu, Mo Zhou, Solving high-dimensional eigenvalue problems using deep neural networks: a diffusion Monte Carlo like approach, J. Comput. Phys. 423 (2020) 109792.
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[10] Weile Jia, Han Wang, Mohan Chen, Denghui Lu, Jiduan Liu, Lin Lin, Roberto Car, Linfeng Zhang, et al., Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning, arXiv preprint, arXiv:2005.00223, 2020.
[11] Yuehaw Khoo, Jianfeng Lu, Lexing Ying, Solving for high-dimensional committor functions using artificial neural networks, Res. Math. Sci. 6 (1) (2019) 1.
[12] Diederik P. Kingma, Jimmy Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.
[13] Peter E. Kloeden, Eckhard Platen, Numerical Solution of Stochastic Differential Equations, vol. 23, Springer Science & Business Media, 2013.
[14] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (May 2017) 84–90.
[15] Haoya Li, Yuehaw Khoo, Yinuo Ren, Lexing Ying, Solving for high dimensional committor functions using neural network with online approximation to derivatives, arXiv preprint, arXiv:2012.06727, 2020.
[16] Maziar Raissi, Paris Perdikaris, George E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[17] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, Weinan E, Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics, Phys. Rev. Lett. 120 (14) (2018) 143001.