J Syst Sci Complex (2023) 36: 1–22

A Deep Learning Method for Computing Eigenvalues of the Fractional Schrödinger Operator*

GUO Yixiao · MING Pingbing

DOI:
Received: 30 June 2023 / Revised: 15 August 2023

Abstract The authors present a novel deep learning method for computing eigenvalues of the fractional Schrödinger operator. The proposed approach combines a newly developed loss function with an innovative neural network architecture that incorporates prior knowledge of the problem. These improvements enable the proposed method to handle both high-dimensional problems and problems posed on irregular bounded domains. The authors successfully compute up to the first 30 eigenvalues for various fractional Schrödinger operators. As an application, the authors share a conjecture to the fractional order isospectral problem that has not yet been studied.

Keywords Eigenvalue problem, deep learning, fractional Schrödinger operator, isospectral problem.

1 Introduction

Fractional partial differential equations have proven effective in modeling anomalous diffusion phenomena[1], turbulent flows[2], porous media flows[3], and many others. In the field of quantum mechanics, Laskin introduced the fractional Schrödinger equation[4, 5] by using Levy flights to replace the classical Brownian motion. This equation has been used to reveal some novel phenomena[6, 7], which could not be explained by the standard Schrödinger equation.

Because of the non-locality and the singularity, most of the numerical methods for solving fractional partial differential equations focus on one-dimensional problems. For two-dimensional problems, the most popular method is an adaptive finite element method proposed by Ainsworth and Glusa[8, 9], while [10] also presented some modified finite difference methods. In addition, the authors in [11, 12] studied the spectral methods to solve the fractional partial differential equations, but applying them in irregular domains presents significant challenges. Another

GUO Yixiao
LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, AMSS, Chinese Academy of Sciences, Beijing 100190*, China*; Email: guoyixiao@lsec.cc.ac.cn.
MING Pingbing (Corresponding author)
School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049*, China.*
Email: mpb@lsec.cc.ac.cn.
*This work was supported by the National Natural Science Foundation of China under Grant No. 11971467.
◇ *This paper was recommended for publication by Editor YAN Zhenya.*

innovative method is the walk-on-spheres (WOS) method[13, 14] and its extension[15]. Those methods use the Feynman-Kac formula to calculate the solution at any given point by simulating a large number of paths of the $2s$-stable Lévy process. We refer to [16–18] for a comprehensive review of the numerical methods for fractional partial differential equations.

In this paper, we study the eigenvalue problem of the fractional Schrödinger operator in a bounded domain.

$$\begin{cases} (-\Delta)^s u(x) + V(x)u(x) = \lambda u(x), & x \in \Omega \subset \mathbb{R}^d, \\ u(x) = 0, & x \in \Omega^c \triangleq \mathbb{R}^d \backslash \Omega. \end{cases} \tag{1}$$

Here, $V(x)$ is a real-valued potential energy function. The precise definition of the operator $(-\Delta)^s$ with $s \in (0, 1)$ will be presented in the subsequent section. When $V(x) = 0$, the problem simplifies to the eigenvalue problem of the fractional Laplace operator:

$$\begin{cases} (-\Delta)^s u(x) = \lambda u(x), & x \in \Omega, \\ u(x) = 0, & x \in \Omega^c. \end{cases} \tag{2}$$

To the best of our knowledge, the exact eigenvalue for any of the aforementioned problems is still unknown, and no analytical formulas even exist for calculating them under any specific circumstance. However, several numerical methods have been developed to tackle this challenge.

In [19, 20], the author presented a spectral method capable of computing very tight bounds for the eigenvalues of the fractional Laplacian within a unit ball in any dimension. Additionally, [21] introduced a spectral method that approximates the eigenvalues of the one-dimensional fractional Schrödinger operator. This method employs the same basis functions as [11] and can be extended to handle problems in hypercubes. Furthermore, a finite element method is displayed in [22] for approximating the eigenvalues within arbitrary bounded domains in one and two dimensions.

The use of neural networks for computing eigenvalues and eigenvectors dates back to the 1990s[23, 24]. While the recent efforts focus on solving the eigenvalue problem of many-body quantum systems[25–27]. These methods use neural networks to represent the underlying functions and incorporate techniques such as the variational Monte Carlo method and stochastic optimizers to approach the ground state of the quantum system. [28–33] also exploited neural networks to solve eigenvalue problems. However, there is little literature on the numerical analysis of solving eigenvalue problems with neural networks except[34].

In this work, we propose a deep learning method to solve the eigenvalue problem associated with the fractional Schrödinger operator. We convert the original eigenvalue problem into a sequence of minimization problems with constraints. By solving these minimization problems sequentially, we can determine the eigenmodes in ascending order of their eigenvalues. In addition, we introduce a novel loss function for solving these minimization problems. This loss function incorporates penalty terms to efficiently handle the orthogonal constraints. Moreover, we design a new neural network architecture that incorporates various feature functions. These feature functions are derived from prior knowledge of the underlying problem, particularly focusing on aspects such as singularity and boundary conditions.

To evaluate the accuracy of our method, we firstly solve a range of examples that could be addressed by the spectral methods also. The dimension of these problems varies from 1 to 9 and our network employs approximately 3,500 parameters. The relative error of our method is less than 0.1% for the first 5 eigenvalues. We further calculate up to 30 eigenvalues while maintaining an error of less than 1%. Subsequently, we test our method in general domains where the spectral methods could not be used and compare the results with those obtained by the finite element method. The results demonstrate that our method produces more accurate results than the finite element method over the finest mesh.

We also implement our method for a fractional Schrödinger operator with an inverse square potential function in three dimensions. By computing the first 30 eigenvalues, we observe that the order of the eigenvalues exchanges as the fractional order varies in this example. Additionally, we provide our estimations of eigenvalues for problems with different potential functions that have never been tested and exhibit eigenfunctions for various problems. All these examples collectively demonstrate the accuracy and efficiency of our method.

As an application, we apply the method to the fractional version of the isospectral problem. Based on our numerical result, we conjecture that even if the spectra of two domains are identical for the Laplacian, they would not be the same for the fractional Laplacian.

The rest of the paper is as follows. In Section 2, we begin with a discussion about the fractional operator and present the newly devised loss function. In Section 3, we display a deep learning scheme for solving the eigenvalue problem with a novel neural network architecture. We show that the results of numerous numerical experiments in Section 4 and compare them with other existing methods. We apply our method to the fractional version of the isospectral problem in Section 5 and draw the conclusions in Section 6.

2 Formulation

The fractional Laplace operator $(-\Delta)^s$ has many different definitions[17, 35] and these definitions are not equivalent in all circumstances. In this paper, we adopt the Ritz definition as follows:

$$(-\Delta)^s u(x) = C_{d,s} \int_{\mathbb{R}^d} \frac{u(x) - u(y)}{\|y - x\|^{d+2s}} dy, \quad \text{for all } x \in \mathbb{R}^d, \tag{3}$$

where

$$C_{d,s} := \frac{2^{2s} s \Gamma(s + d/2)}{\pi^{d/2} \Gamma(1 - s)} \tag{4}$$

and $\|y-x\|$ represents the distance between x and y, i.e., $\|y-x\| = \|y-x\|_{\ell_2}$. The integral in (3) should be interpreted in the principal value sense. This definition is also known as the integral definition in the literature and it is equivalent to the definition via the Fourier transform[36]

$$(-\Delta)^s u(x) = \mathcal{F}^{-1}\{|\xi|^{2s} \mathcal{F}\{u\}(\xi)\}(x), \tag{5}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent the Fourier transform and the inverse Fourier transform, respectively.

To discuss the variational form of the eigenvalue problem (1), we first define the standard fractional order Sobolev space as

$$H^s(\mathbb{R}^d) := \left\{ u \in L^2(\mathbb{R}^d) : \int_{\mathbb{R}^d} (1+|\xi|^{2s}) |\mathcal{F}u(\xi)|^2 d\xi < \infty \right\}. \tag{6}$$

With the equivalent definition of the fractional Laplacian, it can be expressed as

$$H^s(\mathbb{R}^d) := \left\{ u \in L^2(\mathbb{R}^d) : \|u\|_{H^s(\mathbb{R}^d)} < \infty \right\}, \tag{7}$$

where the norm is

$$\|u\|^2_{H^s(\mathbb{R}^d)} = \|u\|^2_{L^2(\mathbb{R}^d)} + |u|^2_{H^s(\mathbb{R}^d)} \tag{8}$$

with the seminorm is

$$|u|^2_{H^s(\mathbb{R}^d)} = \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \frac{[u(y)-u(x)]^2}{\|x-y\|^{d+2s}} dy dx. \tag{9}$$

For the eigenvalue problem with homogeneous Dirichlet boundary condition, we find a solution in

$$H^s_c(\Omega) := \left\{ u \in H^s(\mathbb{R}^d) : u(x) = 0, \quad \text{for all } x \in \Omega^c \right\}. \tag{10}$$

A bilinear form associated with the fractional Schrödinger operator is derived from the nonlocal Green's first identity (see [16, (1.22)])

$$\begin{aligned} a(u,v) :=& \int_\Omega ((-\Delta)^s u(x) + V(x)u(x))v(x)dx \\ =& \frac{C_{d,s}}{2} \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \frac{(u(x)-u(y))(v(x)-v(y))}{\|x-y\|^{d+2s}} dy dx + \int_\Omega V(x)u(x)v(x)dx. \end{aligned} \tag{11}$$

Using this bilinear form and the variational principle, the kth smallest eigenvalue is given by

$$\lambda_k = \min_E \max_{u \in E \setminus \{0\}} \frac{a(u,u)}{\|u\|^2_{L^2(\Omega)}}, \tag{12}$$

where E is a k-dimensional subspace of $H^s_c(\Omega)$. [37–39] displayed some analytical results for the eigenvalue of the fractional Schrödinger operator.

Numerically solving the min-max problem to derive the eigenvalues is challenging and requires significant effort. Therefore, we propose a new formulation that provides a more convenient approach to solve the problem by solely minimizing a loss function. Given the first k eigenmodes $(\lambda_1, u_1), \cdots, (\lambda_k, u_k)$, the subsequent eigenvalue can be characterized as

$$\lambda_{k+1} = \min_{u \in E^{(k)}} \frac{a(u,u)}{\|u\|^2_{L^2(\Omega)}}, \tag{13}$$

where

$$E^{(k)} = \{ u \in H^s_c(\Omega) \setminus \{0\} : u \perp u_i, 1 \le i \le k \}. \tag{14}$$

We use a neural network to approximate the next eigenfunction and construct a loss function by incorporating penalty terms to handle the orthogonal constraints. The loss function for computing the kth eigenvalue is

$$L_k(u) = \frac{a(u,u)}{\|u\|^2_{L^2(\Omega)}} + \beta \sum_{j=1}^{k-1} \frac{(u,u_j)^2}{\|u\|^2_{L^2(\Omega)}\|u_j\|^2_{L^2(\Omega)}}, \tag{15}$$

where β is a penalty parameter and its value should be greater than $\lambda_k - \lambda_1$. When $k = 1$, the loss function reduces to

$$L_1(u) = \frac{a(u,u)}{\|u\|^2_{L^2(\Omega)}}, \tag{16}$$

and the penalty term becomes unnecessary. We remark that the loss function (15) originates from the variational principle and does not rely on any specific properties of the fractional differential operators. Thus, this loss function can be employed for the standard Schrödinger operator and other ordinary differential operators that admit Dirichlet forms.

We use the deep learning scheme introduced in Section 3 to minimize the loss. After minimizing the loss, the neural network provides an approximation of the eigenfunction $\widehat{u}_k$, while the eigenvalue λ_k is approximated as

$$\widehat{\lambda}_k = \frac{a(\widehat{u}_k, \widehat{u}_k)}{\|\widehat{u}_k\|^2_{L^2(\Omega)}}. \tag{17}$$

We iterate this process to obtain the eigenmodes one by one. Since the exact eigenfunction is unknown, the approximating function $\widehat{u}_k$ will be used as a substitute in (15). Consequently, any numerical errors from previous calculations can impact the accuracy of eigenmodes with higher eigenvalues. As the training progresses, the numerical error becomes larger and larger. However, subsequent experiments demonstrate that our method can calculate the first dozens of eigenmodes with great precision.

3 Deep Learning Scheme

In this section, we present our deep learning scheme for solving the eigenvalue problem (1). We introduce a new architecture and describe the Monte Carlo sampling method used to calculate the loss.

When applying deep learning to solve PDE problems, the fully connected neural network (FCNN) and the residual network (ResNet) are commonly employed to approximate functions. However, these architectures have some weaknesses. They could not ensure the boundary conditions, and penalty terms are often added to the loss function to constrain the functions in general. Moreover, these architectures tend to underperform when the solutions contain singular terms. According to [40], the solutions of fractional partial differential equations exhibit an s-order singularity near the boundary, i.e.,

$$u(x) \approx \mathrm{dist}(x, \partial\Omega)^s + v(x), \quad x \in \overline{\Omega}, \tag{18}$$

where $v(x)$ is a smooth function over $\overline{\Omega}$. Furthermore, [21] also conjectured that the eigenfunctions of the fractional Schrödinger operator exhibit an s-order singularity near the boundary. To overcome these drawbacks, we design a new architecture that incorporates prior knowledge about the singular term of the functions and the boundary conditions.

Our architecture is based on FCNN, and a similar one is shown in [41]. It consists of several hidden layers and an output layer, while we mainly change the formulation of the output layer. Each hidden layer is a fully connected layer with the same width. We denote the number of hidden layers as l and the width of each layer as m. The activation function we used is $\sigma = \tanh$ and we denote the vectorized function of $\sigma(x)$ as $\phi(x)$, i.e.,

$$\phi(x) = (\sigma(x_1), \sigma(x_2), \cdots, \sigma(x_m)).$$

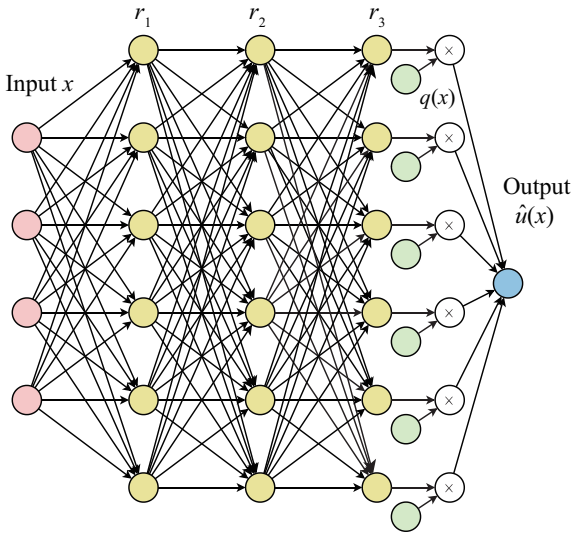


Figure 1 The neural network architecuture combined with feature funtcions

The input layer contains a linear transformation that maps input values from $\mathbb{R}^d$ to $\mathbb{R}^m$, followed by an activation introducing non-linearity. The output of the first layer is given by

$$r_1 = \phi(W_1 x + b_1), \tag{19}$$

where $W_1 \in \mathbb{R}^{m\times d}$ and $b_1 \in \mathbb{R}^m$. The other hidden layers also contain similar transformations, mapping values from $\mathbb{R}^m$ to $\mathbb{R}^m$, and the output of the ith layer is represented as

$$r_i = \phi(W_i r_{i-1} + b_i), \quad 2 \le i \le l, \tag{20}$$

where $W_i \in \mathbb{R}^{m\times m}$ and $b_i \in \mathbb{R}^m$. The output layer combines the outputs of the lth layer and a series of feature functions. The final output of the entire network is given by

$$\widehat{u}(x) = \sum_{j=1}^{m} W_{l+1}^{(j)} \cdot r_l^{(j)} \cdot q_j(x), \tag{21}$$

where $W_{l+1} \in \mathbb{R}^m$, $W_{l+1}^{(j)}$ and $r_l^{(j)}$ is the jth component of W_{l+1} and r_l, respectively. The feature functions are the key part of our method and play a critical role in ensuring the boundary conditions and capturing the singularity of the eigenfunctions. The selection of these functions is based on prior knowledge and significantly influences the accuracy and efficiency of our method. We enforce that the feature functions belong to $H_c^s(\Omega)$ to ensure the final output resides in the appropriate function space. The choice of the feature functions for different examples will be provided in the next section.

The set of all parameters is defined as

$$\theta := \{W_1, \cdots, W_{l+1}, b_1, \cdots, b_l\}. \tag{22}$$

During each epoch, we calculate the loss function $L_k[u(x;\theta)]$ and employ stochastic optimization methods, such as the adaptive moment estimation (ADAM) optimizer, to update the parameters. After multiple epochs, the loss function will diminish to a value small enough and we will derive the approximated eigenmode. To balance accuracy and efficiency, we gradually reduce the learning rate and increase the number of sampling points as the training progresses.

Next, we present the details of the sampling technique used to estimate the loss function. The calculation of (u, u_j) and $\|u\|^2_{L^2(\Omega)}$ is straightforward. We uniformly sample N points $x_1, x_2, \cdots, x_N$ in Ω and calculate these values unbiasedly by

$$(u, u_j) \approx \frac{|\Omega|}{N} \sum_{i=1}^{N} u(x_i) u_j(x_i) \tag{23}$$

and

$$\|u\|^2_{L^2(\Omega)} \approx \frac{|\Omega|}{N} \sum_{i=1}^{N} u(x_i)^2. \tag{24}$$

However, calculating the quadratic form $a(u, u)$ requires a more sophisticated approach since it is a double integral over $\mathbb{R}^d$ and the denominator becomes extremely small when the two variables are in close proximity. To address these difficulties, we pick a convex domain D that contains Ω and separate the integral into two parts.

$$\begin{aligned} a(u,u) &:= \frac{C_{d,s}}{2} \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \frac{[u(y)-u(x)]^2}{\|x-y\|^{d+2s}} dydx \\ &= \frac{C_{d,s}}{2} \int_D \int_D \frac{[u(y)-u(x)]^2}{\|x-y\|^{d+2s}} dydx + C_{d,s} \int_D \int_{D^c} \frac{[u(y)-u(x)]^2}{\|x-y\|^{d+2s}} dydx \\ &\triangleq A_1 + A_2. \end{aligned} \tag{25}$$

For the first part A_1, the integral domain is bounded, but its value approaches infinity as y approaches x. We alleviate this by applying a coordinate transformation to change the integral

formulation

$$\begin{aligned} A_1 &= \frac{C_{d,s}}{2}\int_D\int_D \frac{[u(y)-u(x)]^2}{\|x-y\|^{d+2s}}dydx \\ &= \frac{C_{d,s}}{2}\int_D\int_{S^{d-1}}\int_0^{+\infty} \mathbf{1}_{x+w\xi\in D}\frac{[u(x+w\xi)-u(x)]^2}{w^{1+2s}}dwd\xi dx \\ &= \frac{C_{d,s}}{2}\int_D\int_{S^{d-1}}\int_0^{w^+} \left[\frac{u(x+w\xi)-u(x)}{w}\right]^2\frac{dw}{w^{2s-1}}d\xi dx. \end{aligned} \tag{26}$$

Here, S^{d-1} is the unit $(d-1)$-dimensional sphere and w^+ denotes the distance from the point x to the boundary ∂D along the direction ξ. The shape of D should facilitate this calculation and its size should be as small as possible to enhance sampling efficiency. It is crucial that D is a convex domain, which guarantees that any ray starting from x intersects the boundary ∂D only once. This makes the last equal sign in (26) and simplifies the calculation.

For the numerical implementation, we first uniformly sample $x_1, x_2, \cdots, x_N$ in D, and uniformly sample $\xi_1, \xi_2, \cdots, \xi_N$ in S^{d-1}. Then, we calculate w_i^+ which depends on x_i and ξ_i, and sample w_i with the probability

$$P(w_i = w) = \mathbf{1}_{0<w<w_i^+}\frac{1}{w^{2s-1}}\frac{2-2s}{(w_i^+)^{2-2s}}. \tag{27}$$

To reduce numerical instability caused by dividing a very small amount, we set a minimum value w_c for w. In our numerical experiments, we take $w_c = 10^{-4}$ and

$$\widetilde{w}_i = \max(w_i, w_c) = \max(w_i, 10^{-4}). \tag{28}$$

Through these steps, we derive a practicable and efficient sampling method for calculating A_1,

$$A_1 \approx \frac{C_{d,s}}{2N}\,|D|\,|S^{d-1}|\,\sum_{i=1}^{N}\left[\left[\frac{u(x_i+\widetilde{w}_i\xi_i)-u(x_i)}{\widetilde{w}_i}\right]^2\frac{(w_i^+)^{2-2s}}{2-2s}\right]. \tag{29}$$

The calculation of A_2 is comparatively simpler, as $u(y) = 0$ when $y \in \Omega^c$. We simplify the equation by calculating part of the integral in advance and this prevents sampling in an infinite domain.

$$\begin{aligned} A_2 &= C_{d,s}\int_D\int_{D^c}\frac{[u(y)-u(x)]^2}{\|x-y\|^{d+2s}}dydx \\ &= C_{d,s}\int_D\int_{S^{d-1}}\int_{w^+}^{+\infty}\frac{u(x)^2}{w^{1+2s}}dwd\xi dx \\ &= C_{d,s}\int_D\int_{S^{d-1}}u(x)^2\frac{1}{(w^+)^{2s}}\frac{1}{2s}d\xi dx. \end{aligned} \tag{30}$$

By employing the same method to uniformly sample x_i and ξ_i in D and S^{d-1}, we can approximate A_2 unbiasedly by

$$A_2 \approx \frac{C_{d,s}}{N}\,|D|\,|S^{d-1}|\,\sum_{i=1}^{N}\left[u(x_i)^2\frac{1}{(w_i^+)^{2s}}\frac{1}{2s}\right]. \tag{31}$$

4 Numerical Results

In this section, we present the numerical results for various problems. Firstly, we introduce the training configuration. We employ the networks introduced in Section 3. Each of them contains 3 hidden layers with a width of 40, unless otherwise stated. The total number of parameters is approximately $3,500$. For each eigenvalue, we conduct $120,000$ epochs. Initially, the learning rate is set to 5×10^{-3}, and $1,000$ points are sampled. After every $20,000$ epochs, we reduce the learning rate to one-fourth of its current value, and double the number of sampling points. To identify the kth eigenvalue ($k \geq 2$), we set the penalty parameter β to 4 times the maximum eigenvalue we have found. This selection is sufficiently large to discover the subsequent correct eigenvalue. Using a too large penalty parameter would cause the loss function to become sharp, leading to inefficient training. All experiments are conducted on a single NVIDIA A100 GPU.

We remark that there are several potential enhancements for the aforementioned setup, such as using larger and deeper networks, training more epochs, and so on. However, implementing these enhancements would require much more effort and resources. The current configuration is chosen by balancing efficiency and accuracy. Generally, it takes approximately 5 minutes to discover a new eigenvalue in most cases.

4.1 Fractional Laplacian in the d-Dimensional Unit Ball

We first calculate the eigenvalues of the fractional Laplace operator in d-dimensional unit balls, i.e., $\Omega = B(0,1)$. As we mentioned before, the exact eigenvalues of this problem are currently unknown. However, Dyda, et al. showed a method to calculate tight lower and upper bounds of the eigenvalues for this specific case[19]. We reproduce their method and obtain several bounds. By utilizing these bounds, we can confirm the first few digits of the exact eigenvalues. These inferred values are then used to test the accuracy of our method. In the following experiments, we calculate the relative errors by

$$e := \frac{|\widehat{\lambda} - \lambda^*|}{\lambda^*},$$

where $\widehat{\lambda}$ is our numerical result and λ^* is the inferred value.

We test our method for $d = 1, 3$, and 9. For simplicity, we let the sampling domain $D = \Omega$ and define the feature functions as

$$q_j(x) := \mathrm{ReLU}\big(1 - \|x\|_2^2\big)^{p_j}. \tag{32}$$

Here, $\mathrm{ReLU}(x)=\max(0,x)$. These feature functions ensure that the output of the neural networks vanishes in Ω^c. The exponents p_j are evenly spaced over the interval $[s,3]$ to capture both the sharp and the smooth behaviors near the boundary.

For the case $d = 1$, the numerical results are summarized in Table 1. Our method successfully provides accurate values for the first 10 eigenvalues, and their relative error are less than 0.2%. For larger s, the results can be better, allowing us to calculate more eigenvalues while maintaining this level of precision. Furthermore, we display the eigenfunctions in Figure 2

to demonstrate the singularity near the boundary. As s decreases, the eigenfunctions become sharper near the boundary.

Table 1 Estimates of the eigenvalues of (2) in $\Omega = (-1, 1)$

s		$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=10$
	Exact	0.97259	1.09219	1.14732	1.18684	1.21655	1.31070
0.05	Our	0.97261	1.09217	1.14735	1.18689	1.21665	1.31325
	Rel. error	2.06×10^{-5}	1.83×10^{-5}	2.61×10^{-5}	4.38×10^{-5}	8.22×10^{-5}	1.95×10^{-3}
	Exact	0.97017	1.60154	2.02882	2.38716	2.69474	3.88845
0.25	Our	0.97020	1.60148	2.02878	2.38761	2.69540	3.89149
	Rel. error	3.09×10^{-5}	3.75×10^{-5}	1.97×10^{-5}	1.88×10^{-4}	2.45×10^{-4}	7.82×10^{-4}
	Exact	1.15777	2.75476	4.31680	5.89215	7.46018	15.3155
0.5	Our	1.15780	2.75496	4.31666	5.89386	7.46028	15.3224
	Rel. error	2.59×10^{-5}	7.26×10^{-5}	3.24×10^{-5}	2.90×10^{-4}	1.34×10^{-5}	4.51×10^{-4}
	Exact	1.59750	5.05976	9.59431	15.0188	21.1894	61.0924
0.75	Our	1.59747	5.05971	9.59273	15.0225	21.1944	61.0977
	Rel. error	1.88×10^{-5}	9.88×10^{-6}	1.65×10^{-4}	2.46×10^{-4}	2.36×10^{-4}	8.68×10^{-5}
	Exact	2.24406	8.59575	18.7168	32.4620	49.7200	186.450
0.95	Our	2.24379	8.59504	18.7175	32.4626	49.7308	186.461
	Rel. error	1.20×10^{-4}	8.26×10^{-5}	3.74×10^{-5}	1.94×10^{-5}	2.17×10^{-4}	5.90×10^{-5}

Note: Exact shows the first few digits of the exact eigenvalue.

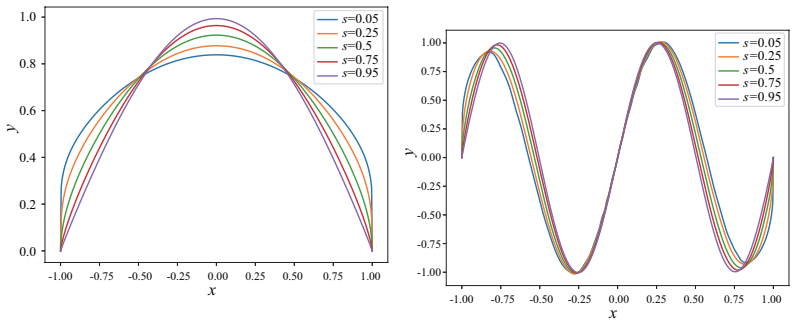


Figure 2 The first and the fourth eigenfunction of (2) in $\Omega = (-1, 1)$

When $d = 3$, we calculate the first 30 eigenvalues. The relative error is less than 0.2% for $s \geq 0.25$. However, the accuracy for $s = 0.05$ is not so satisfactory, compared to the higher order cases. Their errors grow faster so that the results become unreliable earlier. The numerical results are displayed in Table 2. We also calculate the result for $s = 0.9999$. Our numerical

result is very close to the exact value while the eigenvalue of the fractional Laplacian converges to the eigenvalue of the common Laplacian as s approaches 1.

Table 2 Estimates of the eigenvalues of (2) in the unit ball ($d = 3$)

s		$k=1$	$k=2$	$k=3$	$k=5$	$k=10$	$k=15$	$k=30$
	Exact	1.092197	1.14300	1.14300	1.17687	1.18684	1.20274	1.23712
0.05	Our	1.092194	1.14303	1.14304	1.17757	1.18714	1.20518	1.26123*
	Rel. error	2.75×10^{-6}	2.62×10^{-5}	3.50×10^{-5}	5.95×10^{-4}	2.53×10^{-4}	2.03×10^{-3}	1.95e-02
	Exact	1.601538	1.98571	1.98571	2.28647	2.38716	2.54207	2.92181
0.25	Our	1.601535	1.98569	1.98580	2.28750	2.38852	2.54400	2.92759
	Rel. error	1.87×10^{-6}	1.01×10^{-5}	4.53×10^{-5}	4.50×10^{-4}	5.70×10^{-4}	7.59×10^{-4}	1.98×10^{-3}
	Exact	2.75476	4.12130	4.12130	5.40002	5.89215	6.63029	8.71829
0.5	Our	2.75498	4.12087	4.12114	5.40141	5.89405	6.63462	8.72610
	Rel. error	7.99×10^{-5}	1.04×10^{-4}	3.88×10^{-5}	2.57×10^{-4}	3.22×10^{-4}	6.53×10^{-4}	8.96×10^{-4}
	Exact	5.05976	8.93319	8.93319	13.1781	15.0187	17.7566	26.5730
0.75	Our	5.06078	8.93035	8.93205	13.1780	15.0284	17.7742	26.5872
	Rel. error	2.02×10^{-4}	3.18×10^{-4}	1.28×10^{-4}	3.04×10^{-6}	6.40×10^{-4}	9.91×10^{-4}	5.33×10^{-4}
	Exact	8.59575	17.0965	17.0965	27.5394	32.4619	39.8028	65.8034
0.95	Our	8.59548	17.0967	17.1011	27.5366	32.4666	39.8463	65.8256
	Rel. error	3.14×10^{-5}	1.05×10^{-5}	2.67×10^{-4}	1.01×10^{-4}	1.44×10^{-4}	1.09×10^{-3}	3.37×10^{-4}
	Exact	9.86685	20.1840	20.1840	33.2050	39.4629	48.8111	82.6813
0.9999	Our	9.86767	20.1819	20.1830	33.2124	39.4798	48.8591	82.7274
	Rel. error	8.26×10^{-5}	1.02×10^{-4}	4.73×10^{-5}	2.23×10^{-4}	4.17×10^{-4}	9.82×10^{-4}	5.57×10^{-4}
1	Exact	9.86960	20.1907	20.1907	33.2175	39.4784	48.8312	82.7192

Note: Exact show the first few digits of the exact eigenvalue. * represents the error of this solution is too large and it should not be convinced.

In this example, some eigenvalues have a multiplicity greater than 1. For these repeated eigenvalues, our method can identify all the mutually orthogonal eigenfunctions. However, it is inefficient to find the next new eigenvalue with a different value if the multiplicity is too large. For example, in the case of the 9-dimensional ball, the smallest eigenvalue is simple. From the second to the tenth eigenvalue, they share the same value, while the next eigenvalue has a multiplicity of 44. Consequently, it is very time-consuming for our method to find a new eigenvalue beyond these three eigenvalues. The numerical results for the 9-dimensional ball can be found in Table 3.

All these results indicate that solving the eigenvalue problem with a small fractional order is more challenging and the accuracy of the solution becomes worse earlier than the large fractional order cases. Additionally, it should be noted that the standard error of calculating (17) caused by using the Monte Carlo method varies from 3×10^{-5} to 3×10^{-4} in all our experiments, depending on the complexity of the eigenmodes. Hence, the accuracy of these results has approached the limit of our method with the current configuration.

Table 3 Estimates of the eigenvalues of (2) in the 9-dimensional unit ball

s		$k=1$	$k=2$	$k=3$	$k=5$	$k=10$	$k=11$	$k=15$
	Exact	1.20274	1.22386	1.22386	1.22386	1.22386	1.24179	1.24179
0.05	Our	1.20275	1.22392	1.22393	1.22395	1.22475	1.24361	1.24411
	Rel. error	8.31×10^{-6}	4.90×10^{-5}	5.72×10^{-5}	7.35×10^{-5}	7.27×10^{-4}	1.47×10^{-3}	1.87×10^{-3}
	Exact	2.54207	2.76833	2.76833	2.76833	2.76833	2.97357	2.97357
0.25	Our	2.54212	2.76834	2.76855	2.76884	2.78037	2.98222	2.98411
	Rel. error	1.97×10^{-5}	3.61×10^{-6}	7.95×10^{-5}	1.84×10^{-4}	4.35×10^{-3}	2.91×10^{-3}	3.54×10^{-3}
	Exact	6.63029	7.82911	7.82911	7.82911	7.82911	9.00556	9.00556
0.5	Our	6.62929	7.82897	7.82986	7.83152	7.87850	9.02421	9.03107
	Rel. error	1.51×10^{-4}	1.79×10^{-5}	9.58×10^{-5}	3.08×10^{-4}	6.31×10^{-3}	2.07×10^{-3}	2.83×10^{-3}
	Exact	17.7566	22.6391	22.6391	22.6391	22.6391	27.8025	27.8025
0.75	Our	17.7617	22.6390	22.6414	22.6581	22.7577	27.8484	27.9111
	Rel. error	2.87×10^{-4}	4.42×10^{-6}	9.81×10^{-5}	8.39×10^{-4}	5.24×10^{-3}	1.65×10^{-3}	3.91×10^{-3}
	Exact	39.8028	53.8038	53.8039	53.8038	53.8038	69.4807	69.4807
0.95	Our	39.8146	53.8138	53.8427	53.8439	54.0359	69.6889	69.7573
	Rel. error	2.96×10^{-4}	1.86×10^{-4}	7.22×10^{-4}	7.45×10^{-4}	4.31×10^{-3}	3.00×10^{-3}	3.98×10^{-3}

Note: Exact shows the first few digits of the exact eigenvalue.

4.2 Fractional Schrödinger Operator

Next, we solve the eigenvalue problem of the fractional Schrödinger operator with a potential function $V(x)$. To demonstrate the effectiveness of our method, we conduct two tests in one-dimensional intervals and solve a three-dimensional problem with an inverse square potential in a unit ball.

The problem domains for the first two examples are $\Omega = (-1, 1)$. We also let $D = \Omega$ and define the feature functions as

$$q_j(x) := \mathrm{ReLU}\big(1 - x_1^2\big)^{p_j}. \tag{33}$$

The exponents p_j are also evenly spaced over the interval $[s, 3]$. In the first example, the potential function we employed is $V(x) = x^2/2$. We test our method for different s and the numerical results are presented in Table 4. By comparing our results with those obtained from [21], we observe that they are very close to each other. The relative differences are less than 0.06% for the first 10 eigenvalues in all circumstances.

The second example involves a distinct potential function $V(x) = 50x^2 + \sin(2\pi x)$. We calculate the first few eigenvalues and show our estimates in Table 5. Additionally, we plot the first six eigenfunctions in Figure 3, revealing that the shapes and singularity of the eigenfunctions of the fractional Schrödinger operator differ from those of the fractional Laplacian. It is clear that some eigenfunctions do not exhibit singularity near the boundary with this potential. But, our method still successfully identifies them.

Table 4 Estimates of the eigenvalues of (1) with $V(x) = x^2/2$ in $\Omega = (-1, 1)$

s		$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 10$
	Our	1.05992	1.76847	2.19033	2.55183	2.85805	4.05264
0.25	Ref.	1.05995	1.76850	2.19047	2.55226	2.85848	4.05477
	Diff.	2.83×10^{-5}	1.70×10^{-5}	6.39×10^{-5}	1.69×10^{-4}	1.50×10^{-4}	5.26×10^{-4}
	Our	1.24024	2.91807	4.48137	6.05866	7.62650	15.4822
0.5	Ref.	1.24036	2.91792	4.48124	6.05836	7.62828	15.4813
	Diff.	9.68×10^{-5}	5.14×10^{-5}	2.90×10^{-5}	4.95×10^{-5}	2.33×10^{-4}	5.81×10^{-5}
	Our	1.67073	5.21206	9.75501	15.1826	21.3543	61.2587
0.75	Ref.	1.67054	5.21184	9.75495	15.1818	21.3573	61.2629
	Diff.	1.14×10^{-4}	4.22×10^{-5}	6.15×10^{-6}	5.27×10^{-5}	1.40×10^{-4}	6.86×10^{-5}
	Our	2.31064	8.73900	18.8735	32.6231	49.8832	186.616
0.95	Ref.	2.31063	8.73878	18.8749	32.6228	49.8845	186.667
	Diff.	4.33×10^{-6}	2.52×10^{-5}	7.42×10^{-5}	9.20×10^{-6}	2.61×10^{-5}	2.74×10^{-4}

Note: Ref. represents the reference values given by [21]. Diff. represents the relative difference between these two results.

Table 5 Estimates of the eigenvalues of (1) with $V(x) = 50x^2 + \sin(2\pi x)$ in $\Omega = (-1, 1)$

s	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 10$
0.25	2.17977	3.90875	4.74208	5.50170	6.06804	-*
0.5	3.67234	8.61490	11.9706	15.0614	17.7531	29.2724
0.75	5.31594	14.5127	22.3722	30.0024	37.4224	78.2848
0.95	6.71889	20.0334	33.6763	48.9012	66.5992	203.348

Note: * represents it fails to generate a solution due to the accumulation of the compuataional error.

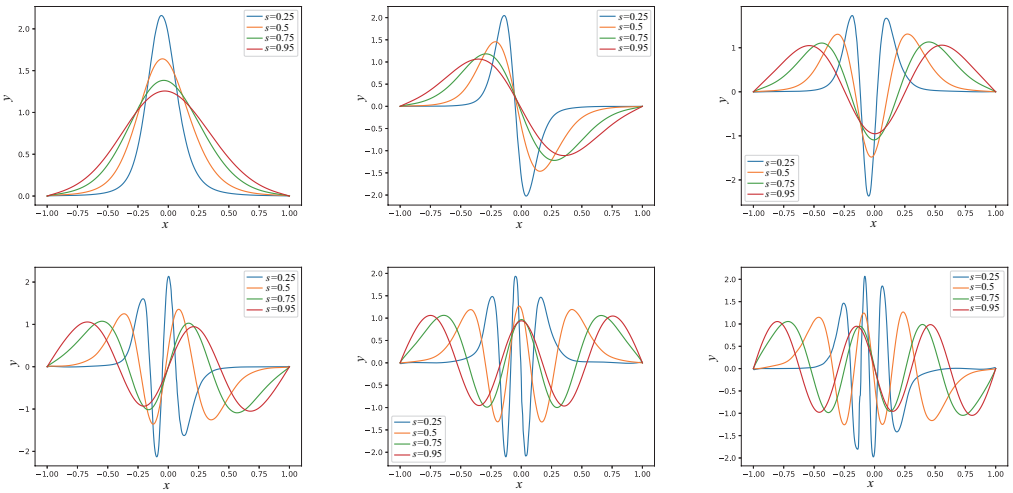
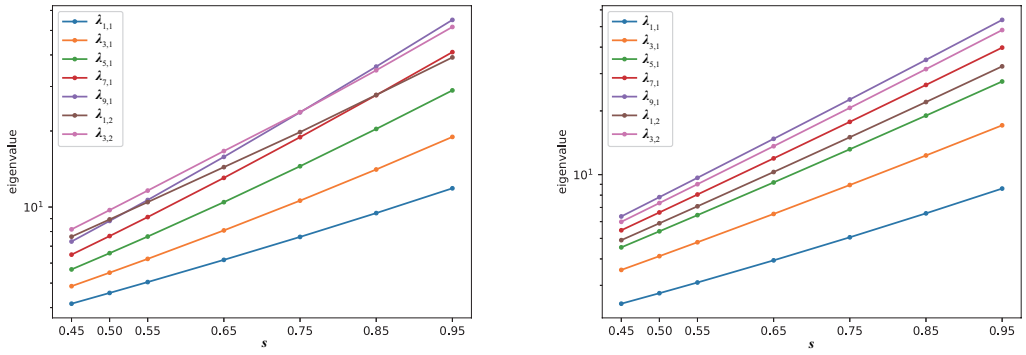


Figure 3 The first six eigenfunctions of (1) with $V(x) = 50x^2 + \sin(2\pi x)$ in $\Omega = (-1, 1)$

Last, we solve the problem (1) in the unit ball with an inverse square potential

$$V(x) = \frac{1}{2(x_1^2 + x_2^2 + x_3^2)}.$$

The feature functions we used are the same as those in $V(x) = 0$. Figure 4 shows the eigenvalues with these two different potential functions. In all other cases, the order of the eigenvalues is independent of s, i.e., they would not exchange for different s. However, in this case, the order of the eigenvalues changes. As s decreases, the value of the first 7-fold eigenvalue is no longer greater than the value of the second single eigenvalue, and the value of the first 9-fold eigenvalue is no longer greater than the value of the second triple eigenvalue.



(a) Potential $V(x) = \frac{1}{2(x_1^2+x_2^2+x_3^2)}$ (b) Potential $V(x) = 0$

Figure 4 Eigenvalues of (1) in the 3-dimensional unit ball with different potentials, $\lambda_{a,1}$ represents the value of the first eigenvalue which has a multiplicity of a while $\lambda_{a,2}$ represents the value of the second eigenvalue which has a multiplicity of a

4.3 Fractional Laplacian in General Domains

In this subsection, we focus on calculating the eigenvalues of the fractional Laplace operator over general domains. To validate our method, we compare the results with those computed by the finite element method in [22].

Firstly, we consider the problem in $\Omega = [-1, 1]^2$ and let the sampling domain $D = \Omega$. The feature function is defined as

$$q_j(x) := \left[\mathrm{ReLU}\left(1 - x_1^2\right) \cdot \mathrm{ReLU}\left(1 - x_2^2\right)\right]^{p_j}. \tag{34}$$

Here, p_j are also evenly spaced over the interval $[s, 3]$. We calculate the first eigenvalue for different s and the outcomes are summarized in Table 6. The results demonstrate that our method outperforms the finite element method over the finest grid, and these values are very close to the extrapolated values obtained through Richardson extrapolation using the finite element method's results. It is worth noting that our approach enables us to obtain the corresponding eigenfunctions, which are not provided by the extrapolation method. Table 6 also

demonstrates the efficiency of our method, as it only takes around 5 minutes to calculate a new eigenmode. We further compute more eigenvalues, and the results are presented in Table 7. The multiplicity of each computed eigenvalue is the same as that of the Laplacian, which is well-known (see, for example, [42]).

Table 6 Estimates of the first eigenvalue of (2) in the square $[-1,1]^2$

s	Our	Extrapolated	FEM	Our Time (s)
0.05	1.04054	1.0405	1.0412	290.2
0.25	1.28129	1.2813	1.2844	292.4
0.5	1.83440	1.8344	1.8395	291.8
0.75	2.88721	2.8872	2.8921	290.6
0.95	4.40568	4.4062	4.4083	296.5

Note: Extrapolated indicates the extrapolated values with FEM results in [22], FEM represents the eigenvalues calculated by the Finite Element Method over the finest mesh in [22].

Table 7 Estimates of the eigenvalues of (2) in the square $[-1,1]^2$

s	$k=1$	$k=2,3$	$k=4$	$k=5,6$	$k=7,8$	$k=9,10$	$k=11$
0.05	1.04054	1.10942	1.14281	1.15823	1.17429	1.19215	1.19685
0.25	1.28129	1.72109	1.97902	2.09892	2.26684	2.43361	2.48065
0.5	1.83440	3.14066	4.08501	4.59306	5.30757	6.09787	6.31421
0.75	2.88721	6.06243	8.79700	10.4447	12.8430	15.7654	16.5448
0.95	4.40568	10.6589	16.7414	20.7257	26.6512	34.4497	36.4295

Next, we turn to the problem in an L-shaped domain $\Omega = [-1,1]^2 \backslash [0,1]^2$. Since Ω is not convex, we let the sampling domain $D = [-1,1]^2$. We employ two types of feature functions. The first-type function is

$$q_{1,j}(x) := \max\left\{ \mathrm{ReLU}\big(-x_1(x_1+1)\big)\mathrm{ReLU}\big(1-x_1^2\big),\ \mathrm{ReLU}\big(-x_2(x_2+1)\big)\mathrm{ReLU}\big(1-x_2^2\big) \right\}^{p_j}. \quad (35)$$

Similar to the previous case, the exponents p_j are evenly spaced over the interval $[s,3]$. It is well known that the solution of Laplace's equation over the $L-$shaped domain exhibits a singularity of type $r^{2/3}f(\theta)$ at the corner. We suspect that certain eigenfunctions may also display a corner singularity. To capture this singularity, we use another type of feature function:

$$q_{2,j}(x) := B\big(2r(x)\big)\ \sin\left(\frac{2}{3}\ \mathrm{ReLU}\left(\theta(x)-\frac{\pi}{2}\right)\right)\ r(x)^{t_j}. \quad (36)$$

Here, $r(x)$ represents the distance between the point x and the corner, while the angle $\theta(x)$ is defined as the angle between the positive x-axis and the line connecting the point x and the

corner in a counterclockwise direction. The exponents t_j are evenly spaced over the interval $[2/3, 3/2]$. $B(\cdot)$ is a bump function defined as

$$B(x) = \begin{cases} \exp\left(-\dfrac{1}{1-x^2}\right), & x \in (-1,1), \\ 0, & \text{otherwise.} \end{cases} \tag{37}$$

In this example, we let the width of the network be 60 and compare two different numerical schemes to demonstrate that incorporating the knowledge about the singularity near the corner can further enhance accuracy for $s > 2/3$. The first scheme we used, denoted as Scheme A in the following paragraphs, utilizes 40 first-type feature functions and 20 second-type feature functions, while the second scheme, referred to as Scheme B, only employs the first-type feature functions. All other settings for these two schemes remain the same.

According to Table 8, Scheme A provides lower estimates than Scheme B when $s = 0.7$ and $s = 0.9$. This can be attributed to the fact that Scheme A incorporates more knowledge about the singularity. However, the results of these two schemes are similar when $s \leq 0.6$, and both schemes outperform the results in [22].

Table 8 Estimates of the first eigenvalue of (2) in the L-shaped domain $[-1,1]^2\backslash[0,1]^2$

s	Our - A	Our - B	Extrapolate	FEM	Time(s) - A	Time(s) - B
0.1	1.14145	1.14145	1.1413	1.1434	397.7	319.9
0.3	1.59621	1.59609	1.5956	1.6025	398.1	322.1
0.5	2.43299	2.43316	2.4322	2.4440	399.1	323.5
0.6	3.09453	3.09478	3.0936	3.1072	397.4	325.4
0.7	4.00864	4.00952	4.0069	4.0228	398.1	324.1
0.9	7.08512	7.09517	7.0790	7.0975	399.5	322.3

Note: Extrapolated indicates the extrapolated values with FEM results in [22], FEM represents the eigenvalues calculated by the Finite Element Method over the finest mesh in [22].

We plot some eigenfunctions of Scheme A in Figure 5, which indicates that the eigenfunctions exhibit similar shapes. But, the eigenfunctions with smaller fractional orders change more sharply near the boundary. Consequently, the absolute value of these functions tends to approach zero in a more narrow region. These behaviors are consistent with the previous example of the fractional Laplacian. In Table 9, we present our estimates for the first few eigenvalues using Scheme A. It shows that the multiplicities of these eigenvalues are consistent with those of the Laplacian[43].

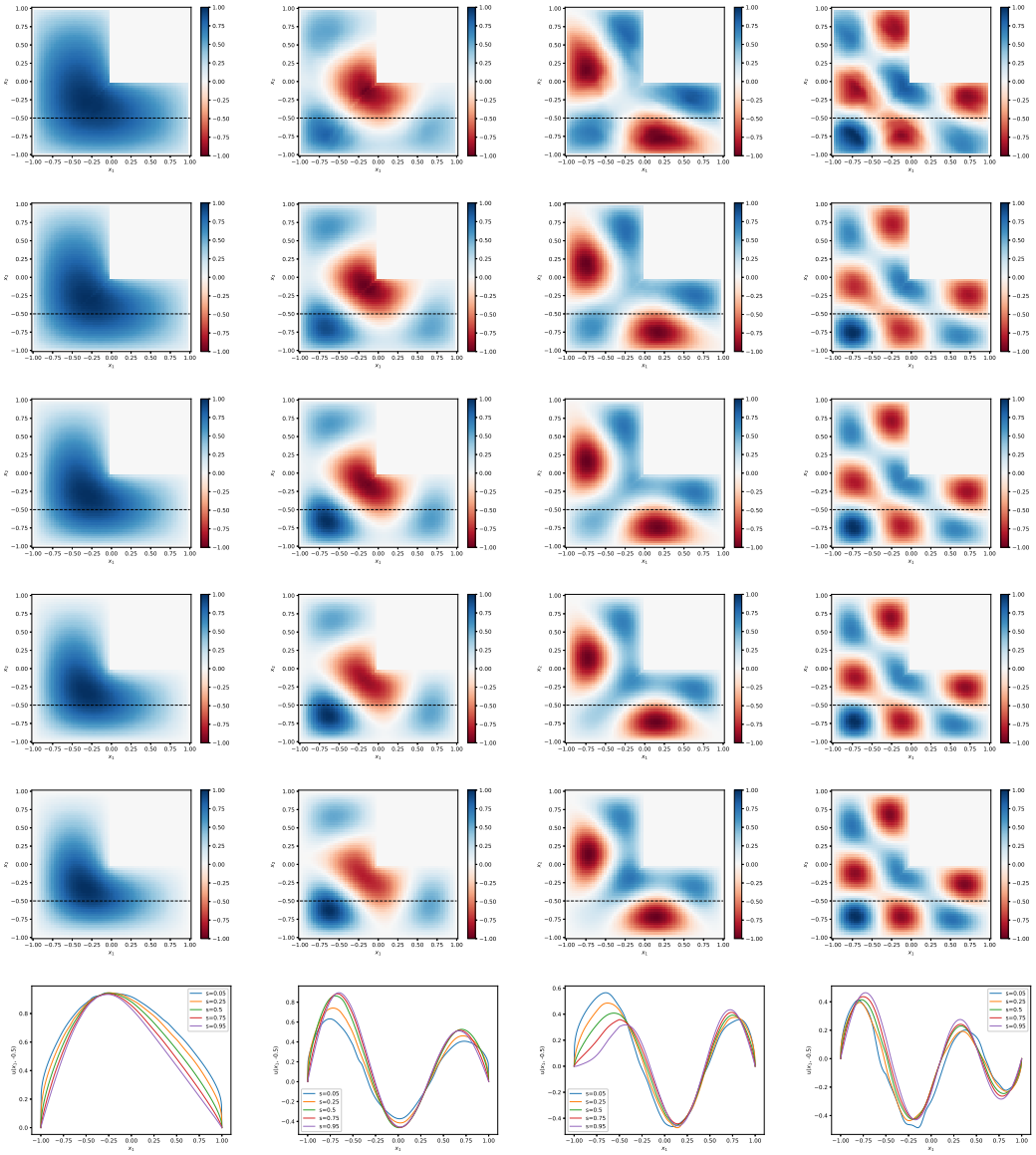


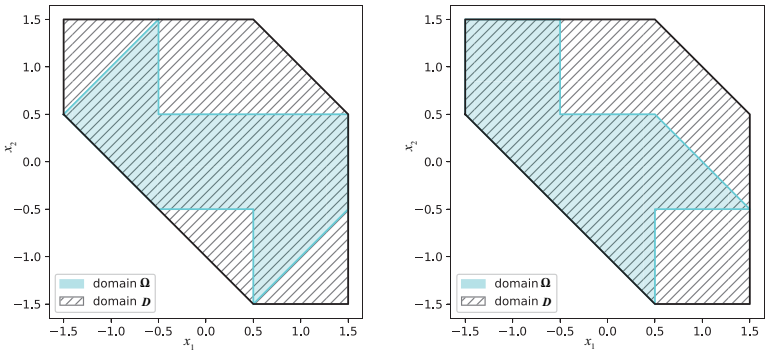
Figure 5 The eigenfunctions of (2) in the L-shaped domain $[-1,1]^2\backslash[0,1]^2$. Each eigenfunction is normalized to let the maximum absolute value equal 1. The first column shows the 1st eigenfunctions for different fractional orders s while the second, third and fourth columns show the 5th, 6th and 10th eigenfunctions, respectively. The first five rows show the eigenfunctions corresponding to $s = 0.05, 0.25, 0.5, 0.75$, and 0.95, respectively. The last row shows eigenfunctions at $x_2 = -0.5$

Table 9 Estimates of the eigenvalues of (2) in the L-shaped domain $[-1,1]^2\backslash[0,1]^2$

s	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8,9$
0.05	1.06508	1.11541	1.13495	1.16620	1.16646	1.18297	1.18921	1.20039
0.25	1.45545	1.77523	1.92973	2.19286	2.20084	2.35759	2.41845	2.52001
0.5	2.43299	3.38167	3.95538	5.00859	5.10676	5.84385	6.12008	6.56155
0.75	4.59315	6.91928	8.59204	11.9203	12.4673	15.2133	16.2296	17.6734
0.95	8.25174	12.9116	16.6466	24.5501	26.3578	33.8279	36.5487	40.0902

5 Isospectral Problem

In this section, we explore the fractional order isospectral problem. In 1966, Kac posed the famous isospectral problem[44], "Can one hear the shape of a drum", which asks whether the Laplace operator with Dirichlet boundary conditions on two different domains can have the same spectrum? In 1992, Gordan, et al. gave a negative answer to this question with a counterexample[45, 46], proving that it is possible for two domains to have the same spectrum. Figure 6 represents their counterexample. Since then, researchers have discovered numerous pairs of domains with identical spectra, and the eigenvalues of many of them are calculated by some numerical works[47–49].



(a) Drum A (b) Drum B

Figure 6 Shape of the problem domain Ω and the sampling domain D of the drum-shaped problem

Now, we wonder whether two different domains that have the same spectrum for the Laplace operator will also have the same spectrum for the fractional Laplace operator. We solve the eigenvalue problem of the fractional Laplace operator in these two domains to draw a conjecture to this question. The relative differences between two eigenvalues are calculated by

$$R_k^{(s)} = \frac{\lambda_{B,k}^{(s)} - \lambda_{A,k}^{(s)}}{\left(\lambda_{A,k}^{(s)} + \lambda_{B,k}^{(s)}\right)/2}. \tag{38}$$

Here, $\lambda_{A,k}^{(s)}$ and $\lambda_{B,k}^{(s)}$ are the kth eigenvalues for the fractional Laplacian with order s in the domains of drum A and drum B, respectively. The previous conclusion stated that

$$R_k^{(1)} = 0, \quad \text{for any } k. \tag{39}$$

But based on the following experiments, we speculate that when $0 < s < 1$,

$$R_k^{(s)} \neq 0, \quad \text{for some } k. \tag{40}$$

Since these two domains are not convex, we select a convex domain D for sampling and the shape of the domain D is plotted in Figure 6 also. We construct two types of feature functions similar to the previous example. The first-type feature functions are used to capture the singularity near the boundary while the second-type captures the singularity at the corners. The network has a width of 60 and consists of 40 first-type feature functions and 20 second-type feature functions. All other settings remain the same as the previous examples.

Table 10 reports the first two eigenvalues in these two domains. It is evident that the first eigenvalue in the domain of drum A is smaller than that in drum B, whereas the second eigenvalue in drum A is greater than that in drum B.

Table 10 Estimates of the first two eigenvalues of (2) in two drum-shaped domains

	s	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$k=1$	drum A	1.1429	1.3406	1.6114	1.9815	2.4887	3.1880	4.1578	5.5182	7.4383
	drum B	1.1438	1.3437	1.6172	1.9921	2.5054	3.2131	4.1913	5.5537	7.4694
$k=2$	drum A	1.2258	1.5244	1.9237	2.4663	3.2077	4.2342	5.6653	7.6851	10.566
	drum B	1.2222	1.5152	1.9056	2.4335	3.1559	4.1591	5.5648	7.5679	10.450

We further compute more eigenvalues and calculate the relative difference between them. The results for different k and s are shown in Figure 7. The value $R_k^{(s)}$ for these k and s we calculated are significantly different from 0 and the maximum relative difference reaches 1.8%. These discrepancies cannot be explained solely by the sampling error of the Monte Carlo method. Therefore, we conjecture that even if the spectra of two domains are identical when $s = 1$, they would not be the same for $0 < s < 1$.

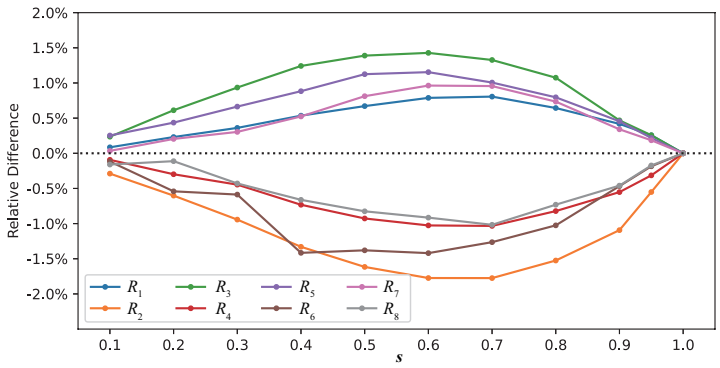


Figure 7 The relative difference $R_k^{(s)}$ for difference k and s

6 Conclusion

Based on a new loss function and a knowledge-based neural network architecture, we propose a novel deep learning method for computing eigenvalues of the fractional Schrödinger operator. We apply the method to problems in high-dimensional space and irregular domains in low dimensions. The numerical results demonstrate that the accuracy of our method in calculating the first few dozen eigenvalues of various problems, and this method outperforms the finite element method[22]. We also draw a new conjecture to the fractional order isospectral problem for exhibiting the capability of the method.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Metzler R and Klafter J, The random walk's guide to anomalous diffusion: A fractional dynamics approach, *Physics Reports*, 2000, **339**(1): 1–77.

[2] Shlesinger M F, West B J, and Klafter J, Lévy dynamics of enhanced diffusion: Application to turbulence, *Physical Review Letters*, 1987, **58**(11): 1100–1103.

[3] de Pablo A, Quirós F, Rodríguez A, et al., A fractional porous medium equation, *Advances in Mathematics*, 2011, **226**(2): 1378–1409.

[4] Laskin N, Fractional quantum mechanics and Lévy path integrals, *Physics Letters A*, 2000, **268**(4–6): 298–305.

[5] Laskin N, Fractional Schrödinger equation, *Physical Review E*, 2002, **66**(5): 056108.

[6] Longhi S, Fractional Schrödinger equation in optics, *Optics Letters*, 2015, **40**(6): 1117–1120.

[7] Zhang Y, Zhong H, Belić M R, et al., PT symmetry in a fractional Schrödinger equation, *Laser & Photonics Reviews*, 2016, **10**(3): 526–531.

[8] Ainsworth M and Glusa C, Aspects of an adaptive finite element method for the fractional Laplacian: A priori and a posteriori error estimates, efficient implementation and multigrid solver, *Computer Methods in Applied Mechanics and Engineering*, 2017, **327**: 4–35.

[9] Ainsworth M and Glusa C, Towards an efficient finite element method for the integral fractional Laplacian on polygonal domains, *Contemporary Computational Mathematics — A Celebration of the* 80*th Birthday of Ian Sloan*, Eds. by Dick J, Kuo F Y, and Woźniakowski H, Springer, Cham, 2018.

[10] Del Teso F, Endal J, and Jakobsen E R, Robust numerical methods for nonlocal (and local) equations of porous medium type, Part II: Schemes and experiments, *SIAM Journal on Numerical Analysis*, 2018, **56**(6): 3611–3647.

[11] Mao Z, Chen S, and Shen J, Efficient and accurate spectral method using generalized Jacobi functions for solving Riesz fractional differential equations, *Applied Numerical Mathematics*, 2016, **106**: 165–181.

[12] Xu K and Darve E, Spectral method for the fractional Laplacian in 2D and 3D, arXiv preprint, 2018, arXiv: 1812.08325.

[13] Kyprianou A E, Osojnik A, and Shardlow T, Unbiased 'walk-on-spheres' Monte Carlo methods for the fractional Laplacian, *IMA Journal of Numerical Analysis*, 2018, **38**(3): 1550–1578.

[14] Shardlow T, A walk outside spheres for the fractional Laplacian: Fields and first eigenvalue, *Mathematics of Computation*, 2019, **88**(320): 2767–2792.

[15] Sheng C, Su B, and Xu C, Efficient Monte Carlo method for integral fractional Laplacian in multiple dimensions, arXiv preprint, 2022, arXiv: 2204.08860.

[16] D'Elia M, Du Q, Glusa C, et al., Numerical methods for nonlocal and fractional models, *Acta Numerica*, 2020, **29**: 1–124.

[17] Lischke A, Pang G, Gulian M, et al., What is the fractional Laplacian? A comparative review with new results, *Journal of Computational Physics*, 2020, **404**: 109009.

[18] Bonito A, Borthagaray J P, Nochetto R H, et al., Numerical methods for fractional diffusion, *Computing and Visualization in Science*, 2018, **19**: 19–46.

[19] Dyda B, Kuznetsov A, and Kwaśnicki M, Eigenvalues of the fractional Laplace operator in the unit ball, *Journal of the London Mathematical Society*, 2017, **95**(2): 500–518.

[20] Dyda B, Fractional calculus for power functions and eigenvalues of the fractional Laplacian, *Fractional Calculus and Applied Analysis*, 2012, **15**(4): 536–555.

[21] Bao W, Chen L, Jiang X, et al., A Jacobi spectral method for computing eigenvalue gaps and their distribution statistics of the fractional Schrödinger operator, *Journal of Computational Physics*, 2020, **421**: 109733.

[22] Borthagaray J P, Del Pezzo L M, and Martínez S, Finite element approximation for the fractional eigenvalue problem, *Journal of Scientific Computing*, 2018, **77**(1): 308–329.

[23] Samardzija N and Waterland R L, A neural network for computing eigenvectors and eigenvalues, *Biological Cybernetics*, 1991, **65**(4): 211–214.

[24] Cichocki A and Unbehauen R, Neural networks for computing eigenvalues and eigenvectors, *Biological Cybernetics*, 1992, **68**: 155–164.

[25] Carleo G and Troyer M, Solving the quantum many-body problem with artificial neural networks, *Science*, 2017, **355**(6325): 602–606.

[26] Choo K, Mezzacapo A, and Carleo G, Fermionic neural-network states for ab-initio electronic structure, *Nature Communications*, 2020, **11**(1): 2368.

[27] Han J, Zhang L, and Weinan E, Solving many-electron Schrödinger equation using deep neural networks, *Journal of Computational Physics*, 2019, **399**: 108929.

[28] Han J, Lu J, and Zhou M, Solving high-dimensional eigenvalue problems using deep neural networks: A diffusion Monte Carlo like approach, *Journal of Computational Physics*, 2020, **423**: 109792.

[29] Li H and Ying L, A semigroup method for high dimensional elliptic PDEs and eigenvalue problems based on neural networks, *Journal of Computational Physics*, 2022, **453**: 110939.

[30] Simonnet E and Chekroun M D, Deep spectral computations in linear and nonlinear diffusion problems, arXiv preprint, 2022, arXiv: 2207.03166.

[31] Zhang W, Li T, and Schtte C, Solving eigenvalue PDEs of metastable diffusion processes using artificial neural networks, *Journal of Computational Physics*, 2022, **465**: 111377.

[32] Elhamod M, Bu J, Singh C, et al., CoPhy-PGNN: Learning physics-guided neural networks with competing loss functions for solving eigenvalue problems, *ACM Transactions on Intelligent Systems*

and Technology, 2022, **13**(6): 1–23.

[33] Finol D, Lu Y, Mahadevan V, et al., Deep convolutional neural networks for eigenvalue problems in mechanics, *International Journal for Numerical Methods in Engineering*, 2019, **118**(5): 258–275.

[34] Lu J and Lu Y, A priori generalization error analysis of two-layer neural networks for solving high dimensional Schrödinger eigenvalue problems, *Communications of the American Mathematical Society*, 2022, **2**(1): 1–21.

[35] Kwaśnicki M, Ten equivalent definitions of the fractional Laplace operator, *Fractional Calculus and Applied Analysis*, 2017, **20**(1): 7–51.

[36] Valdinoci E, From the long jump random walk to the fractional Laplacian, *Bol. Soc. Esp. Mat. Apl.*, 2009, **49**: 33–44.

[37] Frank R L, Eigenvalue bounds for the fractional Laplacian: A review, *Recent Developments in Nonlocal Theory*, Eds. by Palatucci G and Kuusi T, Walter de Gruyter, Berlin, 2017, 210–235.

[38] Chen Z Q and Song R, Two-sided eigenvalue estimates for subordinate processes in domains, *Journal of Functional Analysis*, 2005, **226**(1): 90–113.

[39] Bao W, Ruan X, Shen J, et al., Fundamental gaps of the fractional Schrödinger operator, *Communications in Mathematical Sciences*, 2019, **17**(2): 447–471.

[40] Grubb G, Fractional Laplacians on domains, a development of Hörmander's theory of μ-transmission pseudodifferential operators, *Advances in Mathematics*, 2015, **268**: 478–528.

[41] Khoo Y, Lu J, and Ying L. Solving for high-dimensional committor functions using artificial neural networks, *Research in the Mathematical Sciences*, 2019, **6**: 1–13.

[42] Liu X and Oishi S, Verified eigenvalue evaluation for the Laplacian over polygonal domains of arbitrary shape, *SIAM Journal on Numerical Analysis*, 2013, **51**(3): 1634–1654.

[43] Yuan Q and He Z, Bounds to eigenvalues of the Laplacian on L-shaped domain by variational methods, *Journal of Computational and Applied Mathematics*, 2009, **233**(4): 1083–1090.

[44] Kac M, Can one hear the shape of a drum? *The American Mathematical Monthly*, 1966, **73**(492): 1–33.

[45] Gordon C, Webb D L, and Wolpert S, One cannot hear the shape of a drum, *Bulletin of the American Mathematical Society*, 1992, **27**(1): 134–138.

[46] Gordon C, Webb D L, and Wolpert S, Isospectral plane domains and surfaces via Riemannian orbifolds, *Inventiones Mathematicae*, 1992, **110**(1): 1–22.

[47] Driscoll T A, Eigenmodes of isospectral drums, *SIAM Review*, 1997, **39**(1): 1–17.

[48] Borz A and Borz G, Algebraic multigrid methods for solving generalized eigenvalue problems, *International Journal for Numerical Methods in Engineering*, 2006, **65**(8): 1186–1196.

[49] Li H and Zhang Z, Efficient spectral and spectral element methods for eigenvalue problems of Schrödinger equations with an inverse square potential, *SIAM Journal on Scientific Computing*, 2017, **39**(1): A114–A140.