STNet: Spectral Transformation Network for Solving Operator Eigenvalue Problem

Hong Wang[1,2,3]*, **Yixuan Jiang**[1,4]*, **Jie Wang**[1,2,3]†, **Xinyi Li**[1], **Jian Luo**[1,2,3], **Huanshuo Dong**[1,2,3]

[1] University of Science and Technology of China

[2] CAS Key Laboratory of Technology in GIPAS, University of Science and Technology of China

[3] MoE Key Laboratory of Brain-inspired Intelligent Perception and Cognition, University of Science and Technology of China

[4] Tsinghua University

wanghong1700@mail.ustc.edu.cn, jiangyix25@mails.tsinghua.edu.cn, jiewangx@ustc.edu.cn

Abstract

Operator eigenvalue problems play a critical role in various scientific fields and engineering applications, yet numerical methods are hindered by the curse of dimensionality. Recent deep learning methods provide an efficient approach to address this challenge by iteratively updating neural networks. These methods' performance relies heavily on the spectral distribution of the given operator: larger gaps between the operator's eigenvalues will improve precision, thus tailored spectral transformations that leverage the spectral distribution can enhance their performance. Based on this observation, we propose the **S**pectral **T**ransformation **Net**work (**STNet**). During each iteration, STNet uses approximate eigenvalues and eigenfunctions to perform spectral transformations on the original operator, turning it into an equivalent but easier problem. Specifically, we employ deflation projection to exclude the subspace corresponding to already solved eigenfunctions, thereby reducing the search space and avoiding converging to existing eigenfunctions. Additionally, our filter transform magnifies eigenvalues in the desired region and suppresses those outside, further improving performance. Extensive experiments demonstrate that STNet consistently outperforms existing learning-based methods, achieving state-of-the-art performance in accuracy [1].

1 Introduction

The operator eigenvalue problem is a prominent focus in many scientific fields [10, 4, 6, 35] and engineering applications [8, 5, 13]. However, traditional numerical methods are constrained by the curse of dimensionality, as the computational complexity increases quadratically or even cubically with the mesh size [43].

A promising alternative is using neural networks to approximate eigenfunctions [36]. These approaches reduce the number of parameters by replacing the matrix representation with a parametric nonlinear representation via neural networks. By designing appropriate loss functions, it updates parameters to approximate the desired operator eigenfunctions. These methods only require sampling specific regions without designing a discretization mesh, significantly reducing the algorithm design cost and unnecessary approximation errors [19]. Moreover, neural networks generally exhibit stronger

*Equal contribution.

†Corresponding author.

[1] Our code is available at https://github.com/j1y1x/STNet.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

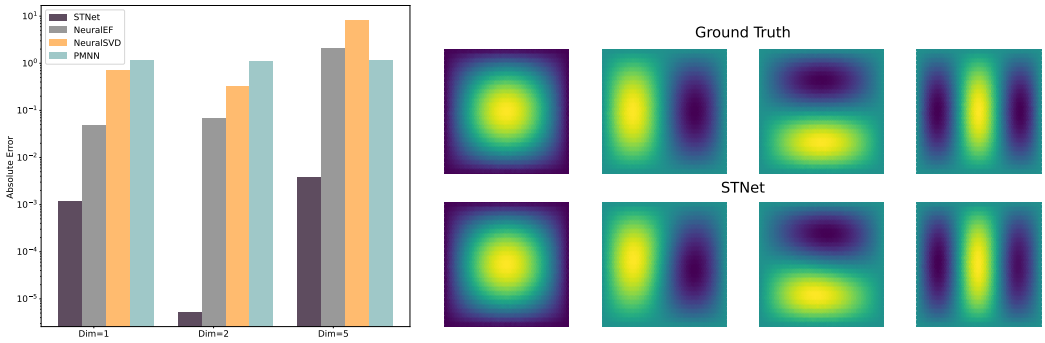

Figure 1: **Left.** Absolute error results of zero eigenvalues for the Fokker-Planck operator computed using various algorithms, the x axis represents the operator dimension. **Right.** Comparison of the eigenfunctions of the 2D Harmonic operator computed by STNet and the ground truth.

expressiveness than linear matrix representations, requiring far fewer sampling points for the same problem compared to traditional methods [34, 14].

Despite these advantages, the performance of such methods strongly depends on the operator's spectral distribution: if the target eigenvalues differs greatly to each other, the algorithm converges much more faster; otherwise, it may suffer from inefficient iterations. To improve convergence, spectral transformations can be designed based on the spectral distribution, reformulating the original problem into an equivalent but more tractable one. However, since the real spectrum of the operator is initially unknown, existing approaches do not optimize spectral properties through such transformations.

To address this limitation, we propose the Spectral Transformation Network (STNet). By exploiting approximate eigenvalues and eigenvectors learned during the iterative process, STNet applies spectral transformations to the original operator, modifying its spectral distribution and thereby converting it into an equivalent problem that converges more easily. Concretely, we employ deflation projection to remove the subspace corresponding to already computed eigenfunctions. This not only narrows the search space but also prevents subsequent eigenfunctions from collapsing into the same subspace. Meanwhile, our filter transform amplifies eigenvalues within the target region and suppresses those outside it, promoting rapid convergence to the desired eigenvalues. Extensive experiments demonstrate that STNet significantly surpasses existing methods based on deep learning, achieving state-of-the-art performance in accuracy. Figure 1 illustrates a selection of the experimental results.

2 Preliminaries

2.1 Operator Eigenvalue Problem

We primarily focus on the eigenvalue problems of differential operators, such as $\frac{\partial}{\partial x} + \frac{\partial}{\partial y}, \Delta$, etc. Mathematically, an operator $\mathcal{L} : \mathcal{H}_1 \to \mathcal{H}_2$ is a mapping between two Hilbert spaces. Considering a self-adjoint operator $\mathcal{L}$ defined on a domain $\Omega \subset \mathbb{R}^D$, the operator eigenvalue problem can be expressed in the following form [11]:

$$\mathcal{L}v = \lambda v \quad \text{in } \Omega, \tag{1}$$

where $\Omega \subseteq \mathbb{R}^D$ serves as the domain; v is the eigenfunction and λ is the eigenvalue. Typically, it is often necessary to solve for multiple eigenvalues, $\lambda_i, i = 1, \ldots, L$.

2.2 Power Method

The power method is a classical algorithm designed to approximate the eigenvalue of an operator $\mathcal{L}$ in the vicinity of a given shift σ. By applying the shift σ (often chosen as an approximation to the target eigenvalue), the original eigenvalue problem is effectively transformed into an equivalent problem for the new operator $(\mathcal{L} - \sigma I)^{-1}$. In each iteration, the current approximate solution is multiplied by this new operator, thereby amplifying the component associated with the eigenvalue closest to σ. This iterative procedure converges to the desired eigenvalue, as shown in Algorithm 1 [15].

Algorithm 1 Power Method for the Operator $\mathcal{L}$

Input: Operator $\mathcal{L}$, shift σ, initial guess v^0, maximum iterations $k_{\max}$, convergence threshold ϵ.
Output: Eigenvalue λ near σ.
$v^0 = v^0/\|v^0\|$.
for $k = 1$ **to** $k_{\max}$ **do**
  $v^k = p^k/\|p^k\|$ and solve $(\mathcal{L} - \sigma I)\, p^k = v^{k-1}$.
  if $\|v^k - v^{k-1}\| < \epsilon$ **then**
    $\lambda = \frac{\langle v^k, \mathcal{L}v^k \rangle}{\langle v^k, v^k \rangle}$ and **break**.
  end if
end for

In each iteration, solving the linear system $(\mathcal{L} - \sigma I)\, p^k = v^{k-1}$ is equivalent to applying the operator $(\mathcal{L} - \sigma I)^{-1}$ to v^{k-1}. Afterward, normalizing v^k helps maintain numerical stability. Convergence is typically assessed by evaluating the error $\|v^k - v^{k-1}\|$, ensuring that the final solution meets the desired accuracy. The fundamental reason for the convergence of the power method lies in the repeated application of $(\mathcal{L} - \sigma I)^{-1}$, which progressively magnifies the component of v^k in the direction of the eigenfunction with eigenvalue closest to σ. For a more detailed introduction to the power method, please refer to the Appendix B.1.

2.3 Deflation Projection

The deflation technique plays a critical role in solving eigenvalue problems, particularly when multiple distinct eigenvalues need to be computed. Deflation projection is an effective deflation strategy that utilizes known eigenvalues and corresponding eigenfunctions to modify the structure of the operator, thereby simplifying the computation of remaining eigenvalues [39].

The core idea of deflation projection is to construct an operator $\mathcal{P}$, often defined as $\mathcal{P}(u) = \langle u, v_1 \rangle v_1$ where v_1 is a known eigenfunction. This operator is then used to modify the original operator $\mathcal{L}$ into a new operator:

$$\mathcal{B} = \mathcal{L} - \lambda_1 \mathcal{P}. \tag{2}$$

In $\mathcal{B}$, the eigenvalue λ_1 associated with v_1 is effectively removed from the spectrum of $\mathcal{L}$. Additional details on deflation projection can be found in Appendix B.2.

2.4 Filter Transform

The filter transform is widely used in numerical linear algebra to enhance the accuracy of eigenvalue computations [39]. By constructing a suitable filter function $F(\mathcal{L})$, the operator $\mathcal{L}$ undergoes a spectral transformation that amplifies the target eigenvalues and suppresses the irrelevant ones. The filter transform can effectively highlight the desired spectral region without altering the corresponding eigenfunctions [43]. Further details on the filter transform can be found in Appendix B.3.

3 Method

3.1 Problem Formulation

We consider the operator eigenvalue problem for a differential operator $\mathcal{L}$ defined on a domain $\Omega \subset \mathbb{R}^D$. Our goal is to approximate the L eigenvalues λ_i near a given shift σ and their corresponding eigenfunctions v_i, satisfying

$$\mathcal{L}\, v_i = \lambda_i\, v_i, \quad i = 1, 2, \ldots, L. \tag{3}$$

To achieve this, we employ L neural networks parameterized by θ_i. Each neural network $NN_{\mathcal{L}}(\cdot; \theta_i)$ maps the domain Ω into $\mathbb{R}$, providing an approximation of the eigenfunction v_i:

$$NN_{\mathcal{L}}(\cdot; \theta_i) : \Omega \to \mathbb{R}, \quad i = 1, 2, \ldots, L. \tag{4}$$

In order to represent both the functions and the operators numerically, we discretize Ω by uniformly randomly sampling N points:

$$S \equiv \{\boldsymbol{x}_j = (x_j^1, \ldots, x_j^D) \mid \boldsymbol{x}_j \in \Omega,\ j = 1, 2, \ldots, N\}, \tag{5}$$

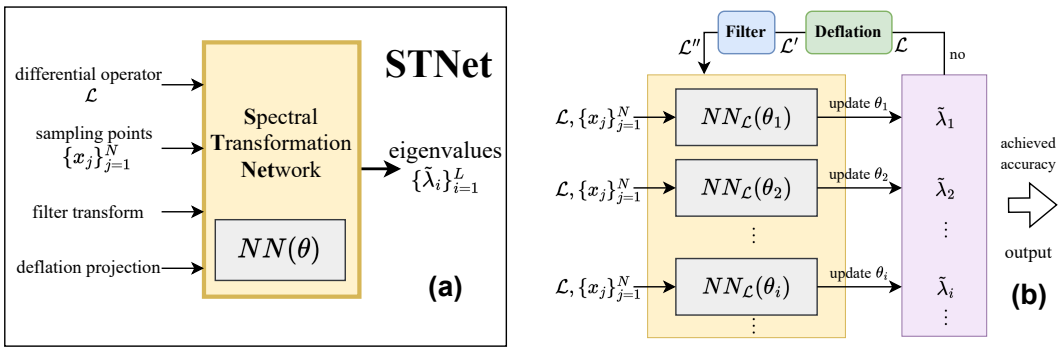

Figure 2: Overview of the **STNet**. **(a)** Introduction to the inputs and outputs. **(b)** STNet comprises multiple neural networks, each tasked with predicting distinct eigenvalues. If the accuracy of the solution reaches the expectation, then STNet will output the result.

Correspondingly, each neural network $NN_{\mathcal{L}}(\cdot;\theta_i)$ output a vector $\boldsymbol{Y}_i \in \mathbb{R}^N$, which approximate the values of the eigenfunction $\tilde{v}_i(\cdot) = NN_{\mathcal{L}}(\cdot;\theta_i)$ at these sampled points:

$$\tilde{v}_i(\boldsymbol{x}_j) \equiv \boldsymbol{Y}_i(j), \quad i = 1, 2, \ldots, L, \quad j = 1, 2, \ldots, N. \tag{6}$$

The approximate eigenvalues $\tilde{\lambda}_i$ are then obtained by applying $\mathcal{L}$ to the computed eigenfunctions $\tilde{v}_i$:

$$\tilde{\lambda}_i \equiv \frac{\langle \tilde{v}_i, \mathcal{L}\tilde{v}_i \rangle}{\langle \tilde{v}_i, \tilde{v}_i \rangle}, \quad i = 1, 2, \ldots, L. \tag{7}$$

Here, the differential operator $\mathcal{L}$ acts on the functions via automatic differentiation. We iteratively update the neural network parameters θ_i using gradient descent, aiming to minimize the overall residual. Specifically, we formulate the following optimization problem:

$$\min_{\theta_i \in \Theta} \frac{1}{N} \sum_{i=1}^{L} \sum_{j=1}^{N} [\tilde{v}_i(\boldsymbol{x}_j) - v_i(\boldsymbol{x}_j)]^2, \tag{8}$$

where Θ denotes the parameter space of the neural networks. This approach does not require any training data, as it relies solely on satisfying the differential operator eigenvalue equations over the domain Ω. Finally, this procedure provides approximations $\tilde{\lambda}_i$ of the true eigenvalues λ_i, $i = 1, \ldots, L$.

3.2 Spectral Transformation Network

Inspired by the power method and the power method neural network [45], we propose STNet to solve eigenvalue problems, as shown in Figure 2. In STNet, we replace the function v^k from the k-th iteration of the power method with $\tilde{v}_i^k(x) \equiv NN_{\mathcal{L}}(x;\theta_i^k)$, where each neural network is implemented via a multilayer perceptron (MLP). Since neural networks cannot directly perform the inverse operator $(\mathcal{L}-\sigma I)^{-1}$, we enforce $(\mathcal{L}-\sigma I)\tilde{v}^k \approx \tilde{v}^{k-1}$ through a suitable loss function. The updated parameters $\theta_i^k \to \theta_i^{k+1}$ then yield $\tilde{v}^{k+1} = NN_{\mathcal{L}}(x;\theta_i^{k+1})$. Algorithm 2 shows the detailed procedure of STNet.

Classical power method convergence is strongly influenced by the spectral distribution of the operator, which is unknown initially and thus difficult to optimize against directly. However, as the iterative process starts, we can get additional information—such as already computed eigenvalues and eigenfunctions. Using these results for the spectral transformation of the original operator can greatly improve subsequent power-method iterations. In Algorithm 2, we introduce two modules to improve the performance. Their impact on the operator spectrum is shown in Figure 3.

- **Deflation projection** uses already computed eigenvalues and eigenfunctions to construct a projection that excludes the previously resolved subspace, preventing convergence to known eigenfunctions and reducing the search space.
- **Filter transform** employs approximate eigenvalues to construct a spectral transformation (filter function) that enlarges the target eigenvalue region and suppresses others, boosting the efficiency of STNet.

Algorithm 2 Spectral Transformation Network

Input: Operator $\mathcal{L}$ over domain $\Omega \subset \mathbb{R}^D$, shift σ, number of sampling points N, number of eigenvalues L, learning rate η, convergence threshold ϵ, maximum iterations $k_{\max}$.
Output: Eigenvalues $\tilde{\lambda}_i, \quad i = 1, \ldots, L$.
Uniformly randomly sample N points $\{\boldsymbol{x}_j\}$ in Ω to form dataset S.
Randomly initialize the network parameters θ_i^0, as well as the normalized $\tilde{v}_i$, and set $\tilde{\lambda}_i = \sigma$, $i = 1, \ldots, L$.
for $k = 1$ **to** $k_{\max}$ **do**
  $\tilde{v}_i^k(\boldsymbol{x}_j) = NN_{\mathcal{L}}(\boldsymbol{x}_j; \theta_i^k), \boldsymbol{x}_j \in S$.
  $\mathcal{L}_i' = D_i(\mathcal{L}), \quad i = 1, \ldots, L$ **// Deflation Projection**
  $\mathcal{L}_i'' = F_i(\mathcal{L}'), \quad i = 1, \ldots, L$ **// Filter Transform**
  $\tilde{u}_i^k(\boldsymbol{x}_j) = \frac{\mathcal{L}_i'' \tilde{v}_i^k(\boldsymbol{x}_j)}{\|\mathcal{L}_i'' \tilde{v}_i^k(\boldsymbol{x}_j)\|}, \quad i = 1, \ldots, L.$
  $\text{Loss}_i^k = \frac{1}{N} \sum_{j=1}^{N} [\tilde{v}_i^{k-1}(\boldsymbol{x}_j) - \tilde{u}_i^k(\boldsymbol{x}_j)]^2, \quad i = 1, \ldots, L.$
  $\theta_i^{k+1} = \theta_i^k - \eta \nabla_{\theta_i} \text{Loss}_i^k, \quad i = 1, \ldots, L$ **// Parameter Update**
  for $i = 1$ **to** L **do**
    if $\text{Loss}_i^k < \epsilon_i$ **then**
      $\epsilon_i = \text{Loss}_i^k, \tilde{\lambda}_i = \frac{\langle \tilde{v}_i^k, \mathcal{L}\tilde{v}_i^k \rangle}{\langle \tilde{v}_i^k, \tilde{v}_i^k \rangle}, \tilde{v}_i = \tilde{v}_i^k.$
    end if
  end for
  if $\epsilon_i < \epsilon$ for all i **then**
    Convergence achieved; **break**.
  else
    Update deflation projection and filter function: $D_i, F_i, \quad i = 1, \ldots, L.$
  end if
end for

3.2.1 Deflation Projection

Suppose we have already approximated the eigenvalues $\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_{i-1}$ and their corresponding eigenfunctions $\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_{i-1}$. To compute the i-th eigenfunction, we focus on the residual subspace orthogonal to the subspace spanned by these previously computed eigenfunctions. The deflated projection is then defined as

$$D_i(\mathcal{L}) \;\equiv\; \mathcal{L} \;-\; \mathcal{Q}_{i-1}\, \Sigma_{i-1}\, \mathcal{Q}_{i-1}^{\top}. \tag{9}$$

Here $\mathcal{Q}_{i-1}$ maps each vector $(\alpha_1, \ldots, \alpha_{i-1}) \in \mathbb{R}^{i-1}$ to the function $\sum_{k=1}^{i-1} \alpha_k \tilde{v}_k$, thus reconstructing functions from the span of $\{\tilde{v}_1, \ldots, \tilde{v}_{i-1}\}$. $\mathcal{Q}_{i-1}^{\top}$ is the transpose of $\mathcal{Q}_{i-1}$. And Σ_{i-1} is a diagonal operator that scales each $\tilde{v}_k$ by its corresponding eigenvalue $\tilde{\lambda}_k$.

By employing the deflation projection, the gradient descent search space of the neural network is constrained to be orthogonal to the subspace spanned by $\{\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_{i-1}\}$. This projection prevents the neural network output $NN_{\mathcal{L}}(\theta_i)$ from converging to the invariant subspace formed by known eigenfunctions, thereby enhancing the orthogonality among the outputs of different neural networks $NN_{\mathcal{L}}(\theta_1), \ldots, NN_{\mathcal{L}}(\theta_{i-1})$. On the one hand, this reduction in the search space accelerates the convergence toward the eigenfunctions v_i; On the other hand, it improves the orthogonality among the neural network outputs, which reduces the error in predicting the eigenfunction $\tilde{v}_i$.

In practice, we use the approximate eigenvalues and eigenfunctions with the smallest error in iterations to construct the deflation projection. This allows us to update adaptively, ensuring that the method remains effective when calculating more eigenfunctions.

3.2.2 Filter Transform

During the iterative process, we can obtain approximate eigenvalues $\tilde{\lambda}_i$, and assume the corresponding true eigenvalues lie within $[\tilde{\lambda}_i - \xi, \tilde{\lambda}_i + \xi]$, where ξ is a tunable parameter, typically $\xi = 0.1$ or $\xi = 1$. We employ a rational function-based filter transform on the original operator to simultaneously

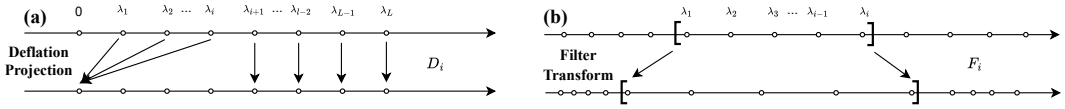

Figure 3: Illustration of the modules' impact on the operator spectrum: **(a)** Deflation projection sets the solved eigenvalues to zero, **(b)** Filter transform enlarges the target eigenvalue region and suppresses others.

amplify the eigenvalues in these intervals and thus improve convergence performance. Specifically, we transform

$$\mathcal{L} \longrightarrow \prod_{i_0=0}^{i-1} \left[(\mathcal{L} - (\tilde{\lambda}_{i_0} - \xi) I) \, (\mathcal{L} - (\tilde{\lambda}_{i_0} + \xi) I) \right]^{-1}. \tag{10}$$

By contrast, the basic power method shift-invert strategy, $\mathcal{L} \to (\mathcal{L} - \sigma I)^{-1}$, can be viewed as a special case of this more general construction. In STNet, we simulate the inverse operator via a suitably designed loss function. Therefore, the corresponding pseudocode filter function F removes the inverse, namely:

$$F_i(\mathcal{L}) \; = \; \prod_{i_0=0}^{i-1} \left[(\mathcal{L} - (\tilde{\lambda}_{i_0} - \xi) I) \, (\mathcal{L} - (\tilde{\lambda}_{i_0} + \xi) I) \right]. \tag{11}$$

When λ_i lies within $[\tilde{\lambda}_i - \xi, \tilde{\lambda}_i + \xi]$, the poles $\tilde{\lambda}_i \pm \xi$ make $\|F_i(v_i)\|$ sufficiently large for the corresponding eigenvector v_i. This repeated amplification causes that direction to dominate in the subsequent iterations, while eigenvalues outside those intervals are gradually suppressed. Consequently, the method converges more efficiently to the desired eigenvalues.

4 Experiments

We conducted comprehensive experiments to evaluate STNet, focusing on:

- Solving multiple eigenvalues in the Harmonic eigenvalue problem.
- Solving the principal eigenvalue in the Schrödinger oscillator equation.
- Solving zero eigenvalues in the Fokker-Planck equation.
- Comparative experiment with traditional algorithms.
- The ablation experiments.

Baselines: For these experiments, we selected three learning-based methods for computing operator eigenvalues as our baselines: 1. PMNN [45]; 2. NeuralEF [7]; 3. NeuralSVD [38]. NeuralSVD and NeuralEF were implemented using the publicly available code provided by the authors of NeuralSVD. PMNN was implemented using the code provided by the authors of PMNN. An introduction to related works can be found in Appendix A. In the comparative experiments with traditional algorithms, we chose the finite difference method (FDM) [24].

Experiment Settings: To ensure consistency, all experiments were conducted under the same computational conditions. For further details on the experimental environment and algorithm parameters, please refer to Appendices C.1 and C.2.

4.1 Harmonic Eigenvalue Problem

Harmonic eigenvalue problems are common in fields such as structural dynamics and acoustics, and can be mathematically expressed as follows [45, 33]:

$$\begin{cases} -\Delta v = \lambda v, & \text{in } \Omega, \\ v = 0, & \text{on } \partial\Omega. \end{cases} \tag{12}$$

Here Δ denotes the Laplacian operator. We consider the domain $\Omega = [0, 1]^D$ where D represents the dimension of the operator, and the boundary conditions are Dirichlet. In this setting, the eigenvalue

Table 1: Relative error comparison for eigenvalues of Harmonic operators. The first row lists the methods, the second row lists eigenvalue indices, and the first column lists the operator dimensions. The most accurate method is in bold.

Method	NeuralEF				NeuralSVD				STNet			
	λ_1	λ_2	λ_3	λ_4	λ_1	λ_2	λ_3	λ_4	λ_1	λ_2	λ_3	λ_4
Dim = 1	1.98e+0	2.53e+0	8.89e-1	7.81e-1	1.72e-2	1.83e-2	**1.88e-2**	9.38e-1	**6.38e-11**	**8.61e-3**	2.13e-2	**4.05e-1**
Dim = 2	1.27e+0	1.66e+1	1.61e+2	1.31e+1	7.23e-3	7.50e-3	7.82e-3	7.64e-3	**5.07e-7**	**1.01e-3**	**2.30e-3**	**2.53e-3**
Dim = 5	3.98e-1	7.24e+0	1.16e+1	1.12e+1	2.88e-2	3.56e-2	3.36e-2	3.31e-2	**4.66e-6**	**1.60e-6**	**1.05e-6**	**2.20e-5**

problem has analytical solutions, with eigenvalues and corresponding eigenfunctions given by:

$$\lambda_{n_1,\ldots,n_D} = \pi^2 \sum_{k=1}^{D} n_k^2, \quad n_k \in \mathbb{N}^+, \quad u_{n_1,\ldots,n_D}(x_1,\ldots,x_k) = \prod_{k=1}^{D} \sin(n_k \pi x_k). \tag{13}$$

These experiments aim to calculate the smallest four eigenvalues of the Harmonic operator in $1, 2$ and 5 dimensions. Since the PMNN model only computes the principal eigenvalue and cannot compute multiple eigenvalues simultaneously, it is not considered for comparison.

Firstly, as demonstrated in Table 1, the accuracy of STNet in most tasks is significantly better than that of existing methods. This enhancement primarily stems from the deflation projection. It effectively excludes solved invariant subspaces during the multi-eigenvalue solution process, thereby preserving the accuracy of multiple eigenvalues. This strongly validates the efficacy of our algorithm.

Table 2: Residual comparison for eigenpairs for solving 5-dimensional Harmonic eigenvalue problems. The first row indicates the eigenpair index.

Index	(v_1, λ_1)	(v_2, λ_2)	(v_3, λ_3)	(v_4, λ_4)
NeuralSVD	1.90e+0	2.63e+0	2.70e+0	3.02e+0
NeuralEF	3.45e+1	2.69e+2	2.10e+1	1.83e+1
STNet	4.864e-4	3.060e-3	5.980e-3	4.447e-3

Secondly, in 5-dimension, STNet consistently maintains a precision improvement of at least three orders of magnitude. As shown in Table 2, this is largely due to the STNet computed eigenpairs having smaller residuals (defined as $||\mathcal{L}\tilde{v} - \tilde{\lambda}\tilde{v}||_2$, see Appendix C.3 for details), indicating that STNet can effectively solve for accurate eigenvalues and eigenfunctions simultaneously.

4.2 Schrödinger Oscillator Equation

The Schrödinger oscillator equation is a common problem in quantum mechanics, and its time-independent form is expressed as follows:

$$-\frac{1}{2}\Delta\psi + V\psi = E\psi, \quad \text{in } \Omega = [0,1]^D, \tag{14}$$

where ψ is the wave function, Δ represents the Laplacian operator indicating the kinetic energy term, V is the potential energy within Ω, and E denotes the energy eigenvalue [38, 16]. This equation is formulated in natural units, simplifying the constants involved. Typically, the potential $V(x_1,\ldots,x_D) = \frac{1}{2}\sum_{k=1}^{D} x_k^2$ characterizes a multidimensional quadratic potential. The principal eigenvalue E_0 and corresponding eigenfunction ψ_0 are given by:

Table 3: Relative error comparison for the principal eigenvalues of oscillator operators. The first row lists the methods, and the first column lists the operator dimensions. The most accurate method is in bold.

Method	PMNN	NeuralEF	NeuralSVD	STNet
Dim = 1	2.34e-6	4.58e+0	1.25e-3	**7.24e-7**
Dim = 2	9.07e-5	3.74e+0	5.95e-2	**2.35e-6**
Dim = 5	1.57e-1	1.78e+0	3.32e-1	**1.29e-1**

$$E_0 = \frac{D}{2}, \quad \psi_0(x_1,\ldots,x_D) = \prod_{k=1}^{D} \left(\frac{1}{\pi}\right)^{\frac{1}{4}} e^{-\frac{x_k^2}{2}}. \tag{15}$$

This experiment focuses on calculating the ground states of the Schrödinger equation in one, two, and five dimensions, i.e., the smallest principal eigenvalues.

Firstly, as shown in Table 3, the STNet achieves significantly higher precision than existing algorithms in computing the principal eigenvalues of the oscillator operator.

Furthermore, the accuracy of STNet surpasses that of PMNN. Both are designed based on the concept of the power method. When solving for the principal eigenvalue, the deflation projection loss may be considered inactive. This outcome suggests that the filter transform significantly enhances the accuracy.

4.3 Fokker-Planck Equation

The Fokker-Planck equation is central to statistical mechanics and is extensively applied across diverse fields such as thermodynamics, particle physics, and financial mathematics [45, 21, 12]. It can be mathematically formulated as follows:

$$-\Delta v - V \cdot \nabla v - \Delta V v = \lambda v, \quad \text{in } \Omega = [0, 2\pi]^D, \quad V(x) = \sin\left(\sum_{i=1}^{D} c_i \cos(x_i)\right). \tag{16}$$

Here $V(x)$ is a potential function with each coefficient c_i varying within $[0.1, 1]$, λ the eigenvalue, and v the eigenfunction. When the boundary conditions are periodic, there are multiple zero eigenvalues.

The eigenvalue at zero significantly impacts the numerical stability of the algorithm during iterative processes. This experiment investigates the computation of two zero eigenvalues for the Fokker-Planck equations with different parameters in 1, 2, and 5 dimensions. Due to the inherent limitation of the PMNN method, which can only compute a single eigenvalue, we restrict our analysis to calculating one eigenvalue when employing this approach.

Table 4: Absolute error comparison for the zero eigenvalues of Fokker-Planck operators across algorithms. The first row lists the methods, the second row lists the eigenvalue index, the first column lists the Fokker-Planck parameter, and the second column lists the operator dimensions. The most accurate method is in bold.

Method		PMNN	NeuralEF		NeuralSVD		STNet	
c_i	Dim	λ_1	λ_1	λ_2	λ_1	λ_2	λ_1	λ_2
0.5	1	1.16e+0	4.98e-2	1.05e+0	7.19e-1	1.02e+0	**1.17e-3**	**8.75e-3**
	2	1.11e+0	6.71e-2	1.57e+0	3.33e-1	1.03e+0	**5.26e-6**	**5.14e-2**
	5	1.17e+0	2.11e+0	9.17e+0	2.11e+0	4.82e+0	**3.90e-3**	**1.29e-1**
1.0	1	8.60e-1	5.21e-1	5.95e-1	2.73e-1	3.19e-1	**3.86e-2**	**2.33e-1**
	2	8.30e-1	6.58e-1	8.45e-1	2.75e-1	3.94e-1	**1.99e-2**	**3.91e-2**
	5	7.58e-1	7.71e-1	1.02e+0	2.01e-1	3.08e-1	**5.64e-2**	**2.67e-2**

As indicated in Table 4, the STNet algorithm significantly outperforms existing methods in computing the zero eigenvalues of the Fokker-Planck operator, effectively solving cases where the eigenvalue is zero. It is mainly due to the filter function, which performs a spectral transformation on the operator, converting the zero eigenvalue into other eigenvalues that are easier to calculate without changing the eigenvector.

4.4 Comparative Experiment with Traditional Algorithms

This experiment compares the accuracy of STNet and the traditional finite difference method (FDM) with a central difference scheme under varying grid densities [24]. Both methods aim to compute the four smallest eigenvalues (λ_1 to λ_4) of the 5D harmonic operator.

As shown in Table 5, STNet consistently achieves higher accuracy than FDM across all eigenvalues. While FDM's precision improves with increasing grid density, this comes at the cost of exponentially higher memory consumption. For instance, as the grid points increases from 4^5 to 45^5, the relative error for λ_1 decreases from 3.20×10^{-1} to 3.82×10^{-3}, but memory usage grows from 0.0001 GB to 22.9 GB. This demonstrates the inefficiency of FDM in high-dimensional problems, where maintaining accuracy requires an impractical number of grid points.

In contrast, STNet employs uniform random sampling instead of fixed grids, enabling it to achieve superior accuracy with fewer parameters and lower memory requirements. For example, with a grid

Table 5: Relative error comparison for eigenvalues of 5D Harmonic operators. The first column lists the methods, the second column lists grid points, and the third column lists memory consumption (in GB). Columns λ_1 to λ_4 list the eigenvalue relative errors.

Method	Grid Points	Memory (GB)	λ_1	λ_2	λ_3	λ_4
FDM	4^5	1.01e-4	3.20e-1	1.04e+0	1.04e+0	1.04e+0
	7^5	1.86e-3	1.26e-1	4.15e-1	4.15e-1	4.15e-1
	9^5	6.74e-3	8.10e-2	2.68e-1	2.68e-1	2.68e-1
	15^5	9.05e-2	3.17e-2	1.05e-1	1.05e-1	1.05e-1
	25^5	1.19e+0	1.20e-2	4.00e-2	4.00e-2	4.00e-2
	35^5	6.48e+0	6.26e-3	2.09e-2	2.09e-2	2.09e-2
	45^5	2.29e+1	3.82e-3	1.27e-2	1.27e-2	1.28e-2
STNet	9^5	1.14e+0	4.62e-5	1.59e-5	1.04e-5	2.32e-4

density of 9^5, STNet achieves relative errors of 2.32×10^{-4} for λ_4, outperforming FDM by orders of magnitude. Its memory usage remains relatively low at 1.14 GB, highlighting its scalability and efficiency in high-dimensional eigenvalue problems. By leveraging the expressive power of neural networks, STNet effectively approximates eigenfunctions without relying on dense grids. This makes it a promising alternative to traditional numerical methods for solving high-dimensional operators. A more detailed comparison with traditional algorithms is provided in Appendix D.4.

Traditional algorithms and neural network-based methods each excel in different scenarios. In low-dimensional problems, traditional algorithms are faster and can improve accuracy by increasing grid density. However, in high-dimensional problems, the number of required grid points grows exponentially with dimensionality, making grid-based methods impractical. For example, while a 2D problem requires 100^2 grid points, a 5D problem requires 100^5 grid points. Neural network-based algorithms, such as STNet, offer an effective solution to these challenges, providing high accuracy without the need for dense grids.

4.5 Ablation Experiments

We conducted ablation experiments to validate the effectiveness of the deflation projection and filter transform modules. As shown in Table 6, the results for "w/o F" indicate that removing the filter transform significantly reduces solution accuracy. In the cases of "w/o D" and "w/o F and D," while the residuals remain small, the absolute errors for λ_2 and λ_3 are notably larger compared to λ_1. This suggests that without the deflation projection module, the network converges exclusively to the first eigenfunction v_1 corresponding to λ_1, failing to capture subsequent eigenfunctions. These findings underscore the critical roles of both modules: the filter transform enhances accuracy through spectral transformation. The deflation projection removes the subspace of already solved eigenfunctions from the search space, enabling the computation of multiple eigenvalues.

Additionally, experiments detailing the performance of STNet as a function of model depth, model width, and the number of sampling points are provided in Appendix D.1. Runtimes and convergence processes for selected experiments are presented in Appendices D.2 and D.3.

Table 6: A comparison of different settings of STNet for the 2-dimensional Harmonic eigenvalue problem. "w/o" denotes the absence of a specific module, "F" represents the filter transform module, and "D" indicates the deflation projection module.

	Index	λ Absolute Error	Residual
STNet	(v_1, λ_1)	1.02e-5	4.12e-3
	(v_2, λ_2)	3.04e-2	1.24e+1
	(v_3, λ_3)	6.76e-1	1.43e+1
w/o F	(v_1, λ_1)	6.73e-5	1.35e-2
	(v_2, λ_2)	5.10e-2	4.72e+1
	(v_3, λ_3)	1.06e-1	1.70e+2
w/o D	(v_1, λ_1)	1.42e-5	4.12e-3
	(v_2, λ_2)	2.96e+1	7.09e-3
	(v_3, λ_3)	2.97e+1	1.09e-2
w/o F and D	(v_1, λ_1)	6.73e-5	1.35e-2
	(v_2, λ_2)	2.96e+1	1.45e-2
	(v_3, λ_3)	2.97e+1	1.37e-2

5 Limitations and Conclusions

In this paper, we propose STNet, a learning-based approach for solving operator eigenvalue problems. By leveraging approximate eigenvalues and eigenvectors from iterative processes, STNet uses spectral transformations to reformulate the operator, enhancing convergence properties. Experiments show that STNet outperforms existing deep learning methods, achieving state-of-the-art accuracy.

While STNet shows strong performance in solving operator eigenvalue problems, several limitations and avenues for future exploration remain: 1. Although STNet utilizes spectral transformations, the potential benefits of broader matrix preconditioning techniques have not been investigated. 2. Its current scope is limited to linear operators, with future work needed to address nonlinear eigenvalue problems.

Acknowledgements

The authors would like to thank all the anonymous reviewers for their insightful comments and valuable suggestions. This work was supported by the National Key R&D Program of China under contract 2022ZD0119801, and the National Nature Science Foundations of China grants U23A20388, 62021001.

References

[1] Amartya S Banerjee, Lin Lin, Wei Hu, Chao Yang, and John E Pask. Chebyshev polynomial filtered subspace iteration in the discontinuous galerkin method for large-scale electronic structure calculations. *The Journal of chemical physics*, 145(15), 2016.

[2] Ido Ben-Shaul, Leah Bar, Dalia Fishelov, and Nir Sochen. Deep learning solution of the eigenvalue problem for differential operators. *Neural Computation*, 35(6):1100–1134, 2023.

[3] Ido Ben-Shaul, Leah Bar, and Nir Sochen. Solving the functional eigen-problem using neural networks. *arXiv preprint arXiv:2007.10205*, 2020.

[4] AG Buchan, CC Pain, F Fang, and IM Navon. A pod reduced-order model for eigenvalue problems with application to reactor physics. *International Journal for Numerical Methods in Engineering*, 95(12):1011–1032, 2013.

[5] Q Chen and YW Chan. Integral finite element method for dynamical analysis of elastic–viscoelastic composite structures. *Computers & Structures*, 74(1):51–64, 2000.

[6] Alice Cuzzocrea, Anthony Scemama, Wim J Briels, Saverio Moroni, and Claudia Filippi. Variational principles in quantum monte carlo: The troubled story of variance minimization. *Journal of chemical theory and computation*, 16(7):4203–4212, 2020.

[7] Zhijie Deng, Jiaxin Shi, and Jun Zhu. Neuralef: Deconstructing kernels by deep neural networks. In *International Conference on Machine Learning*, pages 4976–4992. PMLR, 2022.

[8] Huaian Diao, Hongjie Li, Hongyu Liu, and Jiexin Tang. Spectral properties of an acoustic-elastic transmission eigenvalue problem with applications. *Journal of Differential Equations*, 371:629–659, 2023.

[9] Huanshuo Dong, Hong Wang, Haoyang Liu, Jian Luo, and Jie Wang. Accelerating pde data generation via differential operator action in solution space. *arXiv preprint arXiv:2402.05957*, 2024.

[10] Mohamed H Elhareef and Zeyun Wu. Physics-informed neural network method and application to nuclear reactor calculations: A pilot study. *Nuclear Science and Engineering*, 197(4):601–622, 2023.

[11] Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

[12] Till Daniel Frank. *Nonlinear Fokker-Planck equations: fundamentals and applications*. Springer Science & Business Media, 2005.

[13] Wenhan Gao and Chunmei Wang. Active learning based sampling for high-dimensional nonlinear partial differential equations. *J. Comput. Phys.*, 475:111848, February 2023.

[14] Wenhan Gao, Ruichen Xu, Yuefan Deng, and Yi Liu. Discretization-invariance? on the discretization mismatch errors in neural operators. *The Thirteenth International Conference on Learning Representations*, 2025.

[15] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[16] David J Griffiths and Darrell F Schroeter. *Introduction to quantum mechanics*. Cambridge university press, 2018.

[17] Jiequn Han, Jianfeng Lu, and Mo Zhou. Solving high-dimensional eigenvalue problems using deep neural networks: A diffusion monte carlo like approach. *Journal of Computational Physics*, 423:109792, 2020.

[18] Liu Haoyang, Wang Jie, Cai Yuyang, Han Xiongwei, Kuang Yufei, and HAO Jianye. Optitree: Hierarchical thoughts generation with tree search for LLM optimization modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[19] Cuiyu He, Xiaozhe Hu, and Lin Mu. A mesh-free method using piecewise deep neural network for elliptic interface problems. *Journal of Computational and Applied Mathematics*, 412:114358, 2022.

[20] Henry Jin, Marios Mattheakis, and Pavlos Protopapas. Physics-informed neural networks for quantum eigenvalue problems. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[21] Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

[22] Walter Kohn. Nobel lecture: Electronic structure of matter—wave functions and density functionals. *Reviews of Modern Physics*, 71(5):1253, 1999.

[23] Yufei Kuang, Jie Wang, Haoyang Liu, Fangzhou Zhu, Xijun Li, Jia Zeng, HAO Jianye, Bin Li, and Feng Wu. Rethinking branching on exact combinatorial optimization solver: The first deep symbolic discovery framework. In *The Twelfth International Conference on Learning Representations*, 2024.

[24] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM, 2007.

[25] Xijun Li, Fangzhou Zhu, Hui-Ling Zhen, Weilin Luo, Meng Lu, Yimin Huang, Zhenan Fan, Zirui Zhou, Yufei Kuang, Zhihai Wang, et al. Machine learning insides optverse ai solver: Design principles and applications. *arXiv preprint arXiv:2401.05960*, 2024.

[26] Haoyang Liu, Yufei Kuang, Jie Wang, Xijun Li, Yongdong Zhang, and Feng Wu. Promoting generalization for exact solvers via adversarial instance augmentation, 2023.

[27] Haoyang Liu, Jie Wang, Zijie Geng, Xijun Li, Yuxuan Zong, Fangzhou Zhu, Jianye HAO, and Feng Wu. Apollo-MILP: An alternating prediction-correction neural solving framework for mixed-integer linear programming. In *The Thirteenth International Conference on Learning Representations*, 2025.

[28] Haoyang Liu, Jie Wang, Wanbo Zhang, Zijie Geng, Yufei Kuang, Xijun Li, Bin Li, Yongdong Zhang, and Feng Wu. MILP-studio: MILP instance generation via block structure decomposition. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[29] Hongyu Liu, Haoyang Liu, Yufei Kuang, Jie Wang, and Bin Li. Deep symbolic optimization for combinatorial optimization: Accelerating node selection by discovering potential heuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 2067–2075, 2024.

[30] Jian Luo, Jie Wang, Hong Wang, Zijie Geng, Hanzhu Chen, Yufei Kuang, et al. Neural krylov iteration for accelerating linear system solving. *Advances in Neural Information Processing Systems*, 37:128636–128667, 2024.

[31] Qitan Lv, Tianyu Liu, and Hong Wang. Exploiting edited large language models as general scientific optimizers. *arXiv preprint arXiv:2503.09620*, 2025.

[32] Cun-Qiang Miao and Wen-Ting Wu. On relaxed filtered krylov subspace method for non-symmetric eigenvalue problems. *Journal of Computational and Applied Mathematics*, 398:113698, 2021.

[33] Ronald B Morgan and Min Zeng. Harmonic projection methods for large non-symmetric eigenvalue problems. *Numerical linear algebra with applications*, 5(1):33–55, 1998.

[34] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020.

[35] David Pfau, Simon Axelrod, Halvard Sutterud, Ingrid von Glehn, and James S Spencer. Natural quantum monte carlo computation of excited states. *arXiv preprint arXiv:2308.16848*, 2023.

[36] David Pfau, Stig Petersen, Ashish Agarwal, David GT Barrett, and Kimberly L Stachenfeld. Spectral inference networks: Unifying deep and spectral learning. *arXiv preprint arXiv:1806.02215*, 2018.

[37] Tianle Pu, Zijie Geng, Haoyang Liu, Shixuan Liu, Jie Wang, Li Zeng, Chao Chen, and Changjun Fan. RoME: Domain-robust mixture-of-experts for MILP solution prediction across domains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[38] J Jon Ryu, Xiangxiang Xu, HS Erol, Yuheng Bu, Lizhong Zheng, and Gregory W Wornell. Operator svd with neural networks via nested low-rank approximation. *arXiv preprint arXiv:2402.03655*, 2024.

[39] Yousef Saad. *Numerical methods for large eigenvalue problems: revised edition*. SIAM, 2011.

[40] Pablo Salas, Luc Giraud, Yousef Saad, and Stéphane Moreau. Spectral recycling strategies for the solution of nonlinear eigenproblems in thermoacoustics. *Numerical Linear Algebra with Applications*, 22(6):1039–1058, 2015.

[41] Roel Van Beeumen. Rational krylov methods for nonlinear eigenvalue problems, 2015.

[42] Hong Wang, Zhongkai Hao, Jie Wang, Zijie Geng, Zhen Wang, Bin Li, and Feng Wu. Accelerating data generation for neural operators via krylov subspace recycling. *arXiv preprint arXiv:2401.09516*, 2024.

[43] David S Watkins. *The matrix eigenvalue problem: GR and Krylov subspace methods*. SIAM, 2007.

[44] Jan Winkelmann, Paul Springer, and Edoardo Di Napoli. Chase: Chebyshev accelerated subspace iteration eigensolver for sequences of hermitian eigenvalue problems. *ACM Transactions on Mathematical Software (TOMS)*, 45(2):1–34, 2019.

[45] Qihong Yang, Yangtao Deng, Yu Yang, Qiaolin He, and Shiquan Zhang. Neural networks based on power method and inverse power method for solving linear eigenvalue problems. *Computers & Mathematics with Applications*, 147:14–24, 2023.

[46] Zherui Yang, Zhehao Li, Kangbo Lyu, Yixuan Li, Tao Du, and Ligang Liu. Learning sparse approximate inverse preconditioners for conjugate gradient solvers on gpus. In *Advances in Neural Information Processing Systems*, 2025.

[47] Bing Yu et al. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims are put in the abstract and Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: For further details, please refer to the "Limitations and Conclusions" section, found in Section 5.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: For more information, please see the "Preliminaries" section in Section 2, as well as the theoretical analysis in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: For detailed information, please refer to the "Method" section in Section 3. Complete pseudocode can be found in Algorithm 2. Specific experimental parameters are discussed in Section 4, Appendix C and Appendix D.1.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the instruction and code in supplemental material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Specific experimental parameters are discussed in Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For a detailed comparison, please refer to the experimental sections in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Specific experimental environments can be found in Appendix C.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have checked the NeurIPS code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The work presented in this paper aims to develop a novel eigenvalue-solving algorithm, focused solely on the field of scientific computing. It does not have any direct positive or negative social impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We have checked this and confirmed the paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The original papers or URLs of the codes, models, and data sets used in this article have been cited in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide the instruction in supplemental material about our code and data.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: This paper only utilizes LLMs for writing and editing.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Related work

Recent advancements in applying neural networks to eigenvalue problems have shown promising results [42, 9, 30, 26, 27, 28, 37, 23, 29, 25, 18, 31]. Innovations such as spectral inference networks (SpIN) [36], which model eigenvalue problems as kernel problem optimizations solved via neural networks. Neural eigenfunctions (NeuralEF) [7], which significantly reduce computational costs by optimizing the costly orthogonalization steps, are noteworthy. Neural singular value decomposition (NeuralSVD) employs truncated singular value decomposition for low-rank approximation to enhance the orthogonality required in learning functions [38].

Another class of algorithms originates from optimizing the Rayleigh quotient. The deep Ritz method (DRM) utilizes the Rayleigh quotient for computing the smallest eigenvalues, demonstrating significant potential [47]. Several studies have employed the Rayleigh quotient to construct variation-free functions, achieved through physics-informed neural networks (PINNs) [2, 3]. Extensions of this approach include enhanced loss functions with regularization terms to improve the learning accuracy of the smallest eigenvalues [20]. Additionally, [17] reformulates the eigenvalue problem as a fixed-point problem of the semigroup flow induced by the operator, solving it using the diffusion Monte Carlo method. The power method neural network (PMNN) integrates the power method with PINNs, using an iterative process to approximate the exact eigenvalues [45] closely. While PMNN has proven effective in solving for a single eigenvalue [45], it has yet to be developed for computing multiple distinct eigenvalues simultaneously.

B Background Knowledge and Relevant Analysis

B.1 Convergence Analysis of the Power Method

Suppose $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ with $\boldsymbol{V} = [\boldsymbol{v}_1 \quad \cdots \quad \boldsymbol{v}_n]$. Assume that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. The pseudocode for the power method is shown below [15]:

Algorithm 1: Power method for finding the largest principal eigenvalue of the matrix A

Given $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ an $n \times n$ matrix, an arbitrary unit vector $x^{(0)} \in \mathbb{R}^n$, the maximum number of iterations $k_{\max}$, and the stopping criterion ϵ.
for $k = 1, 2, \ldots, k_{max}$ **do**
  Compute $\boldsymbol{y}^{(k)} = \boldsymbol{A}\boldsymbol{x}^{(k-1)}$.
  Normalize $\boldsymbol{x}^{(k)} = \frac{\boldsymbol{y}^{(k)}}{\|\boldsymbol{y}^{(k)}\|}$.
  Compute the difference $\delta = \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\|$.
  if $\delta < \epsilon$ **then**
    Record the largest principal eigenvalue using the Rayleigh quotient.

$$\lambda^{(k)} = \frac{\langle \boldsymbol{x}^{(k)}, \boldsymbol{A}\boldsymbol{x}^{(k)} \rangle}{\langle \boldsymbol{x}^{(k)}, \boldsymbol{x}^{(k)} \rangle}.$$

    The stopping criterion is met, and the iteration can be stopped.

Let us examine the convergence properties of the power iteration. If

$$\boldsymbol{x}^{(0)} = a_1\boldsymbol{v}_1 + a_2\boldsymbol{v}_2 + \cdots + a_n\boldsymbol{v}_n$$

and $\boldsymbol{v}_1 \neq 0$, then

$$\boldsymbol{A}^k\boldsymbol{x}^{(0)} = a_1\lambda_1^k \left(\boldsymbol{v}_1 + \sum_{j=2}^{n} \frac{a_j}{a_1} \left(\frac{\lambda_j}{\lambda_1} \right)^k \boldsymbol{v}_j \right).$$

Since $\boldsymbol{x}^{(k)} \in \operatorname{span}\{\boldsymbol{A}^k\boldsymbol{x}^{(0)}\}$, we conclude that

$$\operatorname{dist}\Big(\operatorname{span}\{\boldsymbol{x}^{(k)}\}, \operatorname{span}\{\boldsymbol{v}_1\}\Big) = O\left(\left(\frac{\lambda_2}{\lambda_1} \right)^k \right).$$

It is also easy to verify that

$$|\lambda_1 - \lambda^{(k)}| = O\left(\left(\frac{\lambda_2}{\lambda_1}\right)^k\right).$$

Since λ_1 is larger than all the other eigenvalues in modulus, it is referred to as the largest principal eigenvalue. Thus, the power method converges if λ_1 is the largest principal and if $\boldsymbol{x}^{(0)}$ has a component in the direction of the corresponding dominant eigenvector $\boldsymbol{x}_1$.

In practice, the effectiveness of the power method largely depends on the ratio $|\lambda_2|/|\lambda_1|$, as this ratio determines the convergence rate. Therefore, applying specific spectral transformations to the matrix to increase this ratio can significantly accelerate the convergence of the power method.

B.2 Deflation Projection Details

Consider the scenario where we have determined the largest modulus eigenvalue, λ_1, and its corresponding eigenvector, $\boldsymbol{v}_1$, utilizing an algorithm such as the power method. These algorithms consistently identify the eigenvalue of the largest modulus from the given matrix along with an associated eigenvector. We ensure that the vector $\boldsymbol{v}_1$ is normalized such that $\|\boldsymbol{v}_1\|_2 = 1$. The task then becomes computing the subsequent eigenvalue, λ_2, of the matrix $\boldsymbol{A}$. A traditional approach to address this is through what is commonly known as a deflation procedure. This technique involves a rank-one modification to the original matrix, aimed at shifting the eigenvalue λ_1 while preserving all other eigenvalues intact. The modification is designed in such a way that λ_2 emerges as the eigenvalue with the largest modulus in the adjusted matrix. Consequently, the power method can be reapplied to this updated matrix to extract the eigenvalue-eigenvector pair $\lambda_2, \boldsymbol{v}_2$.

When the invariant subspace requiring deflation is one-dimensional, consider the following Proposition B.1. The propositions and proofs below are derived from [39] P90.

Proposition B.1. *Let $\boldsymbol{v}_1$ be an eigenvector of $\boldsymbol{A}$ of norm 1, associated with the eigenvalue λ_1 and let $\boldsymbol{A}_1 \equiv \boldsymbol{A} - \sigma \boldsymbol{v}_1 \boldsymbol{v}_1^H$. Then the eigenvalues of $\boldsymbol{A}_1$ are $\tilde{\lambda}_1 = \lambda_1 - \sigma$ and $\tilde{\lambda}_j = \lambda_j, j = 2, 3, \ldots, n$. Moreover, the Schur vectors associated with $\tilde{\lambda}_j, j = 1, 2, 3, \ldots, n$ are identical with those of $\boldsymbol{A}$.*

Proof. Let $\boldsymbol{AV} = \boldsymbol{VR}$ be the Schur factorization of $\boldsymbol{A}$, where $\boldsymbol{R}$ is upper triangular and $\boldsymbol{V}$ is orthonormal. Then we have

$$\boldsymbol{A}_1\boldsymbol{V} = \left[\boldsymbol{A} - \sigma \boldsymbol{v}_1 \boldsymbol{v}_1^\top\right] \boldsymbol{V} = \boldsymbol{VR} - \sigma \boldsymbol{v}_1 \boldsymbol{e}_1^\top = \boldsymbol{V}[\boldsymbol{R} - \sigma \boldsymbol{e}_1 \boldsymbol{e}_1^\top].$$

Here, $\boldsymbol{e}_1$ is the first standard basis vector. The result follows immediately. □

According to Proposition B.1, once the eigenvalue λ_1 and eigenvector $\boldsymbol{v}_1$ are known, we can define the deflation projection matrix $\boldsymbol{P}_1 = \boldsymbol{I} - \lambda_1 \boldsymbol{v}_1 \boldsymbol{v}_1^\top$ to compute the remaining eigenvalues and eigenvectors.

When deflating with multiple vectors, let $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_j$ be a set of Schur vectors associated with the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_j$. We denote by $\boldsymbol{Q}_j$ the matrix of column vectors $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_j$. Thus, $\boldsymbol{Q}_j \equiv [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_j]$ is an orthonormal matrix whose columns form a basis of the eigenspace associated with the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_j$. An immediate generalization of Proposition B.1 is the following [39] P94.

Proposition B.2. *Let $\boldsymbol{\Sigma}_j$ be the $j \times j$ diagonal matrix $\boldsymbol{\Sigma}_j = diag(\sigma_1, \sigma_2, \ldots, \sigma_j)$, and $\boldsymbol{Q}_j$ an $n \times j$ orthogonal matrix consisting of the Schur vectors of $\boldsymbol{A}$ associated with $\lambda_1, \ldots, \lambda_j$. Then the eigenvalues of the matrix*

$$\boldsymbol{A}_j \equiv \boldsymbol{A} - \boldsymbol{Q}_j \boldsymbol{\Sigma}_j \boldsymbol{Q}_j^\top,$$

are $\tilde{\lambda}_i = \lambda_i - \sigma_i$ for $i \leq j$ and $\tilde{\lambda}_i = \lambda_i$ for $i > j$. Moreover, its associated Schur vectors are identical with those of $\boldsymbol{A}$.

Proof. Let $\boldsymbol{AU} = \boldsymbol{UR}$ be the Schur factorization of $\boldsymbol{A}$. We have

$$\boldsymbol{A}_j\boldsymbol{U} = \left[\boldsymbol{A} - \boldsymbol{Q}_j \boldsymbol{\Sigma}_j \boldsymbol{Q}_j^\top\right] \boldsymbol{U} = \boldsymbol{UR} - \boldsymbol{Q}_j \boldsymbol{\Sigma}_j \boldsymbol{E}_j^\top,$$

where $\boldsymbol{E}_j = [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_j]$. Hence

$$\boldsymbol{A}_j\boldsymbol{U} = \boldsymbol{U}\left[\boldsymbol{R} - \boldsymbol{E}_j \boldsymbol{\Sigma}_j \boldsymbol{E}_j^\top\right]$$

and the result follows. □

According to Proposition B.2, if $\boldsymbol{A}$ is a normal matrix and the eigenvalues $\lambda_1, \ldots, \lambda_j$ along with their corresponding eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_j$ are known, we can construct the deflation projection matrix $\boldsymbol{P}_j = \boldsymbol{I} - \boldsymbol{V}_j \boldsymbol{\Sigma}_j \boldsymbol{V}_j^\top$ to compute the remaining eigenvalues and eigenvectors. Here, $\boldsymbol{\Sigma}_j = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_j)$ and $\boldsymbol{V}_j = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_j]$.

B.3 Filtering Technique

The primary objective of filtering techniques is to manipulate the eigenvalue distribution of a matrix through spectral transformations [39]. This enhances specific eigenvalues of interest, facilitating their recognition and computation by iterative solvers. Filter transformation functions, $F(x)$, typically fall into two categories:

1. Polynomial Filters, expressed as $P(x)$, such as the Chebyshev filter [32, 1].
2. Rational Function Filters, often denoted as $P(x)/Q(x)$, such as the shift-invert method [41, 43]. Below, we describe this strategy in detail.

Shift-Invert Strategy The shift-invert strategy applies the transformation $(A - \sigma I)^{-1}$ to the matrix A, where σ is a scalar approximating a target eigenvalue, termed as shift. This operation transforms each eigenvalue λ of A into $\frac{1}{\lambda - \sigma}$, amplifying those eigenvalues close to σ in the transformed matrix, making them larger and more distinguishable [43].

For instance, consider the power method, where the convergence rate is primarily governed by the ratio of the matrix's largest modulus eigenvalue to its second largest. Suppose matrix A has three principal eigenvalues: $\lambda_1 = 10$, $\lambda_2 = 3$, and $\lambda_3 = 2$. Our objective is to compute λ_1, the largest eigenvalue. In the original matrix A, the convergence rate of the power method hinges on the spectral gap ratio, defined as:

$$\text{Spectral Gap Ratio} = \frac{\lambda_1}{\lambda_2} \approx 3.33$$

Applying the shift-invert transformation with $\sigma = 9.5$ strategically selected close to λ_1, the new eigenvalues μ are recalculated as:

$$\mu_i = \frac{1}{\lambda_i - \sigma}$$

This results in transformed eigenvalues:

$$\mu_1 = 2, \quad \mu_2 \approx -0.133, \quad \mu_3 \approx -0.125$$

Under this transformation, $\mu_1 = 2$ emerges as the dominant eigenvalue in the new matrix, with the other eigenvalues significantly smaller. Consequently, the new spectral gap ratio escalates to:

$$\text{New Spectral Gap Ratio} = \frac{2}{0.133} \approx 15.04$$

This enhanced spectral gap notably accelerates the convergence of the power method in the new matrix configuration.

Filtering techniques are often synergized with techniques like the implicit restarts of Krylov algorithms [43, 15, 46], employing matrix operation optimizations to minimize the computational demands of evaluating matrix functions. These methods enable more precise localization and computation of multiple eigenvalues spread across the spectral range, particularly vital in physical [40, 1] and materials science [22] simulations where these eigenvalues frequently correlate with the system's fundamental properties [44].

C Details of Experimental Setup

C.1 Experimental Environment

To ensure consistency in our evaluations, all comparative experiments were conducted under uniform computing environments. Specifically, the environments used are detailed as follows:

- CPU: 72 vCPU AMD EPYC 9754 128-Core Processor
- GPU: NVIDIA GeForce RTX 4090D (24GB)

C.2 Experimental Parameters

NeuralSVD and NeuralEF were implemented using the publicly available code provided by the authors of NeuralSVD (`https://github.com/jongharyu/neural-svd`). PMNN was implemented using the code provided by the authors of PMNN (`https://github.com/SummerLoveRain/PMNN_IPMNN`). No modifications were made to their original code for the experiments, except for changes to the target operator to be solved. All experiments are fully reproducible. The detailed baseline experimental parameter files and code are provided in the supplementary materials.

- NeuralSVD and NeuralEF: (Using the original paper settings)
  - Optimizer: RMSProp with a learning rate scheduler.
  - Learning rate: 1e-4, batch size: 128
  - Neural Network Architecture: layers = [128,128,128]
  - Laplacian regularization set to 0.01, with evaluation frequency every 10000 iterations.
  - Fourier feature mapping enabled with a size of 256.
  - Neural network structure: hidden layers of 128,128,128 using softplus as the activation function.
- PMNN and STNet
  - Optimizer: Adam
  - Learning rate: 1e-4
  - Neural Network Architecture: Assuming d is the dimension of the operator. For d = 1 or 2, layers = [d, 20, 20, 20, 20, 1]. For d=5, layers = [d, 40, 40, 40, 40, 1]. For the else cases, layers = [d, 40, 40, 40, 40, 1].
  - For the 1-dimensional problem, the number of points is $20,000$, with $400,000$ iterations. For the 2-dimensional problem, the number of points is $40,000 = 200 \times 200$, also with $400,000$ iterations. For the 5-dimensional problem, the number of points is $59,049 = 9^5$, with $500,000$ iterations.

C.3 Error Metrics

- Absolute Error:
  We employ absolute error to estimate the bias of the output eigenvalues of the model:

  $$\text{Absolute Error} = |\tilde{\lambda} - \lambda|.$$

  Here $\tilde{\lambda}$ represents the eigenvalue predicted by the model, while λ denotes the true eigenvalue.
- Relative Error:
  We employ relative error to estimate the bias of the output eigenvalues of the model:

  $$\text{Relative Error} = \frac{|\tilde{\lambda} - \lambda|}{|\lambda|}.$$

  Here $\tilde{\lambda}$ represents the eigenvalue predicted by the model, while λ denotes the true eigenvalue.
- Residual Error:
  To further analyze the error in eigenpair $(\tilde{v}, \tilde{\lambda})$ predictions, we use the following metric:

  $$\text{Residual Error} = ||\mathcal{L}\tilde{v} - \tilde{\lambda}\tilde{v}||_2.$$

  Here, $\tilde{v}$ represents the eigenfunction predicted by the model. When $\tilde{\lambda}$ is the true eigenvalue and $\tilde{v}$ is the true eigenfunction, the Residual Error equals 0.

D Supplementary Experiments

D.1 Analysis of Hyperparameters

Model Depth:

Table 7: Consider the 2-dimensional Harmonic problem, with the fixed layer width of 20, and compare the performance of STNet at different model layers. Other experimental details are the same as Appendix C.2. The error metric is absolute error.

Eigenvalue	Layer 3	Layer 4	Layer 5	Layer 6
λ_1	1.02e-5	1.42e-5	4.36e-6	1.06e-5
λ_2	3.04e-2	2.96e-1	8.63e-1	8.21e-1
λ_3	6.76e-2	4.17e-1	1.98e+0	1.17e+0
λ_4	1.00e-1	2.00e+1	8.94e+1	3.81e+1

Model Width:

Table 8: Consider the 2-dimensional Harmonic problem, with the fixed layer depth of 3, and compare the performance of STNet at different model widths. Other experimental details are the same as Appendix C.2. The error metric is absolute error.

Eigenvalue	Width 10	Width 20	Width 30	Width 40
λ_1	1.68e-6	1.42e-5	3.26e-5	1.57e-5
λ_2	3.82e-1	2.96e-1	1.50e+0	2.67e+0
λ_3	7.54e-1	4.17e-1	1.59e+0	7.93e+1
λ_4	1.71e-1	2.00e+1	3.52e+2	1.50e+2

The Number of Points:

Table 9: Consider the 2-dimensional Harmonic problem and compare the performance of STNet at different numbers of points. Other experimental details are the same as Appendix C.2. The error metric is absolute error.

Eigenvalue	20000 Points	30000 Points	40000 Points	50000 Points
λ_1	1.11e-5	4.40e-5	1.42e-5	4.94e-6
λ_2	1.25e+0	3.58e-1	2.96e-1	2.53e-1
λ_3	1.61e+0	1.70e-1	4.17e-1	3.73e-1

The influence of model depth, model width, and the number of points on STNet is illustrated in Tables 7, 8, and 9, respectively. Experimental results indicate that STNet is relatively unaffected by changes in model depth and model width. However, it is significantly influenced by the number of points, with performance improving as more points are used.

D.2 Computational Complexity and Runtimes

While neural network-based methods may exhibit a larger computational overhead than traditional methods in low-dimensional problems, their strength lies in high-dimensional settings where they offer a significant precision and computational cost advantages. To illustrate this, we provide a runtime comparison for the 5D Harmonic problem in Table 10.

Table 10: Runtime comparison for the 5D Harmonic problem. "STNet (partial)" refers to the time taken to achieve an error of 1.00×10^{-3}.

Method	Computation Time (s)	Principal Eigenvalue Absolute Error
FDM (45^5 points)	$\sim$ 1.9e+4	3.8e-3
STNet (partial)	$\sim$ 1.3e+3	1.0e-3
STNet (converged)	$\sim$ 3.1e+4	4.6e-5
NeuralEF (converged)	$\sim$ 1.2e+4	3.9e-1
NeuralSVD (converged)	$\sim$ 1.2e+4	2.8e-2

The results in Table 10 highlight two key points:

1. STNet achieves a lower error more rapidly than other learning-based methods and converges to a significantly more accurate solution than NeuralEF and NeuralSVD.

2. The total time to full convergence for STNet is longer. This is an inherent trade-off of the Filter Transform, which iteratively refines the problem to unlock further optimization, thus requiring more iterations to reach its high-precision potential.

D.3 Partial Convergence Process

we provide the convergence process for the Harmonic problem in Table 11. The table shows the absolute error for the first two eigenvalues (λ_1, λ_2) at different iteration counts across dimensions 1, 2, and 5.

Table 11: Absolute Error vs. Iterations for the Harmonic problem. Empty cells indicate that convergence for that eigenvalue was already achieved in a prior stage.

Dim	Eigenvalue	Iter: 100	500	1000	40000	80000	130000	180000
d=1	λ_1	1.6e-2	3.5e-3	3.2e-4	2.6e-4	2.7e-5	1.2e-3	
	λ_2	1.0e+4	2.4e+3	2.8e+3	8.0e+2	9.8e-1	8.8e-3	
d=2	λ_1	6.9e-2	6.9e-3	5.9e-3	5.3e-6			
	λ_2	6.3e01	5.9e+0	4.2e+0	5.1e-2			
d=5	λ_1	5.8e-2	6.6e-3	4.8e-4	7.8e-3	3.9e-3	6.2e-3	3.9e-3
	λ_2	6.3e+3	4.1e+2	1.5e+2	2.9e+0	5.5e+0	3.8e+0	1.3e-1

D.4 Impact of Discretization on Traditional Methods

A critical point of comparison is understanding the primary sources of error in different methodologies. For traditional methods like the Finite Difference Method (FDM), the dominant bottleneck in high-dimensional problems is not the algebraic eigenvalue solver's accuracy but the fundamental discretization error introduced when mapping a continuous operator onto a discrete grid. While advanced solver techniques such as deflation or filtering can improve the precision of the matrix eigenvalue solution, they cannot overcome the inherent error from the initial discretization. Reducing this error necessitates a denser grid, which leads to an exponential increase in memory and computational cost, a challenge known as the "curse of dimensionality."

To empirically validate this, we conducted a experiment for the 5D Harmonic problem, comparing STNet against FDM enhanced with Richardson extrapolation—a technique designed to improve

accuracy. The results, presented in Table 12, show that even when FDM is augmented with advanced techniques, it cannot match the accuracy of STNet.

Table 12: Comparison with advanced FDM techniques for the 5D Harmonic problem.

Method	Grid/Points	Memory (GB)	λ_1 Error	λ_2 Error	λ_3 Error	λ_4 Error
FDM (Richardson Extrap.)	25^5	3.4e+0	8.6e-3	2.9e-2	2.9e-2	2.9e-2
FDM (with deflation & filtering; Richardson Extrap.)	25^5	3.4e+0	8.6e-3	2.9e-2	2.9e-2	2.9e-2
FDM (Central Difference)	45^5	2.2e+1	3.8e-3	1.2e-2	1.2e-2	1.2e-2
FDM (with deflation & filtering; Central Difference)	45^5	2.2e+1	3.8e-3	1.2e-2	1.2e-2	1.2e-2
STNet	$\mathbf{9^5}$	**1.1e+0**	**4.6e-5**	**1.6e-5**	**1.0e-5**	**2.3e-4**

This experiment reinforces that neural network-based methods, which rely on mesh-free random sampling, are inherently better suited for high-dimensional problems, as they effectively bypass the discretization step that limits traditional grid-based approaches.

While traditional methods are often superior for low-dimensional problems, their practicality diminishes rapidly in high dimensions due to the "curse of dimensionality". To provide a more granular analysis, Table 13 details the convergence time versus accuracy for the 5D Harmonic problem. Our traditional baseline uses 'scipy.eigs', which is based on the highly-optimized ARPACK library. The data clearly shows that FDM's accuracy is capped by the grid resolution, and STNet can achieve higher precision. The total time for STNet is longer, but it reaches a level of accuracy that is infeasible for FDM under reasonable memory constraints.

Table 13: Detailed convergence analysis for the 5D Harmonic problem (λ_1). The table shows the time required to reach specific absolute error thresholds. A dash (-) indicates that the method could not reach the specified error due to limitations imposed by grid resolution.

Method	Grid/Sample Pts	Memory (GB)	Final Abs. Error	Time to Error < 1e+0	Time to Error < 1e-1	Time to Error < 1e-2	Time to Error < 1e-3	Total Time
FDM+eigs	15^5 ($\sim$ 8e+5)	9.1e-2	3.2e-2	1.3s	4.0s	1.1e+1s	-	1.1e+1s
	25^5 ($\sim$ 1e+7)	1.2e+0	1.2e-2	2.5e+1s	3.2e+1s	8.1e+1s	-	1.1e+2s
	35^5 ($\sim$ 5e+7)	6.5e+0	6.3e-3	1.8e+2s	2.5e+2s	4.9e+2s	1.9e+3s	1.9e+3s
	45^5 ($\sim$ 2e+8)	2.3e+1	3.8e-3	9.4e+2s	3.1e+3s	6.7e+3s	1.9e+4s	1.9e+4s
STNet	5.0e+4	9.5e-1	3.4e-4	7.2s	1.4e+1s	1.7e+2s	1.3e+3s	1.0e+5s
	9^5 ($\sim$ 6e+4)	1.1e+0	4.6e-5	9.6s	1.9e+1s	1.5e+2s	1.3e+3s	3.1e+4s
	7.0e+4	1.3e+0	6.5e-5	5.7s	1.1e+1s	1.4e+2s	3.0e+2s	5.4e+4s
	8.0e+4	1.5e+0	2.5e-5	8.9s	1.6e+1s	1.6e+2s	9.8e+2s	2.9e+4s

This detailed analysis validates our conclusion that for large-scale, high-dimensional problems, STNet holds a significant advantage in both achievable accuracy and memory efficiency. Its core strength lies in its mesh-free random sampling strategy, which leverages the expressive power of neural networks to approximate eigenfunctions directly, thereby circumventing the need to construct and store a massive matrix.