Deep Learning Solution of the Eigenvalue Problem for Differential Operators

Ido Ben-Shaul[1] **Leah Bar**[1] **Dalia Fishelov**[2] **Nir Sochen**[1]

[1]Department of Applied Mathematics, Tel- Aviv University, Israel.

[2]Department of Mathematics, Afeka Tel-Aviv Academic College of Engineering, Tel-Aviv, Israel.

Keywords: Eigenvalue Problem, PDEs, Deep Learning

Abstract

Solving the eigenvalue problem for differential operators is a common problem in many scientific fields. Classical numerical methods rely on intricate domain discretization, and yield non-analytic or non-smooth approximations. We introduce a novel Neural Network (NN) -based solver for the eigenvalue problem of differential self-adjoint operators, where the eigenpairs are learned in an unsupervised end-to-end fashion. We propose several training procedures for solving increasingly challenging tasks towards the general eigenvalue problem. The proposed solver is capable of finding the M smallest eigenpairs for a general differential operator. We demonstrate the method on the Laplacian operator which is of particular interest in image processing, computer vision and shape analysis among many other applications. In addition, we solve the Legendre differential equation. Our proposed method *simultaneously* solves several eigenpairs and can be easily used on free-form domains. We exemplify it on L-shape and circular cut domains. A significant contribution of this work is an analysis of the numerical error of this method. In particular an upper bound for the (unknown) solution error is given in terms of the (measured) truncation error of the PDE and the network structure.

1 Introduction

Eigenfunctions and eigenvalues of the Laplacian (among other operators) are important in various applications ranging, inter alia, from image processing to computer vision, shape analysis and quantum mechanics. It is also of major importance in various engineering applications where resonance is crucial for design and safety Benouhiba and Belyacine (2013). Laplacian eigenfunctions allow us to perform spectral analysis of data measured in more general domains or even on graphs and networks Shi and Malik (2000). Additionally, the M-smallest eigenvalues of the Laplace-Beltrami operator, are fundamental features for comparing geometric objects such as 3D shapes, images or point clouds via the functional maps method in statistical shape analysis Ovsjanikov et al. (2012). Moreover, in quantum mechanics, the smallest eigenvalues of the Hamiltonian and their corresponding eigenfunctions, are of great physical significance Han et al. (2019). In this paper we present a novel numerical method for the computation of these eigenfunctions (efs.) and eigenvalues (evs.), where the efs. are parameterized by neural networks (NNs) with continuous activation functions, and the evs. are directly calculated via the Rayleigh quotient. The resulting efs. are therefore *smooth* functions defined in a parametric way. This is in contrast to the finite element Pradhan and Chakraverty (2019) and finite difference Saad (2005); Knyazev (2000) methods in which the efs. are defined on either a grid or as piecewise linear/polynomial functions with limited smoothness. Direct and iterative matrix-based methods for large eigenvalues problems are reported in the book of Saad (2011). In these matrix-based approaches, one has to discretize the problem first. In the finite difference method, there is an inherent problem handling curved boundaries where the boundary conditions are needed to be cropped by the computational cartesian domain. Moreover, discretization or truncation error is emerged by the gap between a differential operator and a difference operator. The error increases as one increases the order of the derivative of the function to be approximated. As for FEM, the linear system representing a finite element discretization might be sensitive to the discretizing mesh, especially in curved domains.

Following and generalizing Bar and Sochen (2019, 2021), we suggest an unsupervised approach to learn the eigenpairs of a differential operator. It is performed on a specified domain with boundary conditions, where the network simultaneously approximates one or several eigenfunctions at every entry x. The method is based on a uniformly distributed point set, which is trained to satisfy three fidelity terms of the eigenvalue problem formulated as the L_1, L_2, and L_∞-like norms. In addition, the proposed loss function includes boundary conditions, orthogonality constraints and regularization. There are several advantages to the proposed setting: (i) The framework is general and can be used for general non linear differential operators with high order derivatives. (ii) Since we sample the domain using a point cloud, we are not limited to standard domains. Using this fact may assist when attempting to solve problems on arbitrary domains. (iii) Different priors and regularization schemes can be easily integrated into the loss function, allowing specific domain adaptivity. (iv) The suggested framework *simultaneously* solves for multiple eigenpairs. Our method attains to a family of PDEs (one for each eigenvalue and eigenfunction) in one/several network(s) that solve these multiple PDEs together. (v) From theoretical perspective, we analyze the error of the PDE's solution and relate it to the network architecture and the measured truncation error of the equation. We illustrate the theory, on an example, by relating the truncation error to the structure of the network.

Let us stress that this work does not address the quality of the optimization that is done in deep learning literature. We use standard techniques in this domain and the optimization is not guaranteed to attain the global minimum. Our point in the analysis presented in this paper, is to bound the approximation error of the eigenfunction given the *empirical error in the equation*. Indeed, The truncation error is the main term in our loss function and is measured empirically. What we proved is that given this truncation error, we are able to bound the error in the solution of the equation.

We demonstrate the capabilities of the proposed solution for both known and multiple unknown eigenvalues of Legendre's equation and the Laplacian operator in 1D and different 2D domains: rectangular, L-shaped and free-form shape. We additionally show a quantitative analysis demonstrating the robustness of the method compared with previous works.

2 Related Work

Many recent approaches have shown promise in using the power of NNs to approximate solutions of differential equations, while classical methods are often prone to weakness due to the discretization of the domain Ω. Raissi et al. (2017) solved continuous and discrete time models by NNs which model the solution using both the automatic differentiation and boundary conditions. Bar and Sochen (2019, 2021) proposed a solver for both forward and inverse problems with the additional L_∞ norm imposed on the deviation from the differential equation. In the work of Chen et al. (2018), differential equation solvers are used as part of the network architecture, and are shown to enhance the smoothness and convergence of the solutions. In order to properly solve differential equations, a representation that captures high-order derivatives is desired. Additionally, Sitzmann et al. (2020), proposed a network architecture that illustrates these requirements using periodic activation functions with proper initialization.

The well-known power method and its variants Eastman and Estep (2007) is a classical method for addressing the eigenvalue problem. It works on specific *Linear* operators, $\mathcal{L} : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^k)$, and can be used only after some discretization procedure, where the continuous equation is reduced numerically to a matrix eigenpair problem. This process introduces numerical errors even before the solution of the eigenvalue problem. The usage of the power method for spectral operators on Hilbert spaces was shown by Erickson et al. (1995). A recent modified method for non-linear differential operators was proposed by Hait-Fraenkel and Gilboa (2019). Most power method variants for operators, converge to a single eigenpair. Finding the M smallest eigenpairs can be both computationally and algorithmically challenging.

Advanced methods addressing the eigenvalue problem solve for several eigenpairs together. One such class of solutions is the Krylov-based methods Stewart (2001); Lehoucq et al. (1998) that are commonly used in practice such as the `eigs` command in MATLAB (2015). Other class of solutions use deep networks were recently introduced. These methods are based on variational Monte Carlo (VMC) and diffusion Monte Carlo (DMC) methods. VMC relies on leveraging physical knowledge to propose an ansatz of the eigenfunction and incorporates the essential physics Han et al. (2019); Hermann et al. (2019); Pfau et al. (2019b); Choo et al. (2019). Recently, Han et al. (2020) formulated the eigenvalue problem by the stochastic backward equation using the DMC method, where the loss function optimizes the eigenvalue, eigenfunction and the scaled gradient of the eigenfunction. The loss function consists of L_2 norm of two fidelity terms with additional normalization. Using this method yields the first eigenpair with an optional second eigenpair given some mild prior estimate of the eigenvalue. Pfau et al. (2019b), suggested a neural network-based method for learning the eigenfunctions of linear operator by stochastic optimization. They reformulated the Rayleigh quotient such that they replaced the operator matrix with a continuous kernel. The method was demonstrated on the 2D Schrödinger equation, where the Laplacian operator was discretized using finite differences.

In this suggested work, we formulate the eigenvalue problem in a direct setting with a flexible number of eigenpairs, where no discretization is applied. The derivatives of the differential operator are analytically calculated through the network structure. Additionally, we use the L_∞, L_2 and L_1 norms for fidelity, and boundary condition terms to accomplish a strong (pointwise) solution.

3 Preliminaries

Let $\mathcal{H}$ be a Hilbert space where the inner product for $u, v \in \mathcal{H}$ is denoted as $\langle u, v\rangle$. Let $A \in O(\mathcal{H})$ be an operator, and A^* its adjoint operator defined by $\langle A^* u, v\rangle = \langle u, Av\rangle \ \forall u, v \in \mathcal{H}$. A is then said to be self-adjoint if $A = A^*$.

Lemma 3.1. *Let $\mathcal{H}$ be a Hilbert space and $A \in O(\mathcal{H})$ a self-adjoint operator. Then all eigenvalues of A are real Conway (1985).*

In this work we focus on self-adjoint operators. An eigenpair of an operator A is defined as: (u, λ) s.t. $\lambda \in \mathbb{R}$, where u is the eigenfunction of A, and λ is the corresponding eigenvalue. Let A be a self-adjoint operator $A : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^k)$. We wish to approximate eigenpairs $\{u_i, \lambda_i\}$ such that

$$Au_i + \lambda_i u_i = 0 \ \forall i. \tag{1}$$

The proposed algorithm approximates the eigenfunction set $u_i(x)$ by $\tilde{u}_i(x; \theta_{\tilde{u}_i})$ via a neural network (NN), where $\theta_{\tilde{u}_i}$ denotes the network parameters. The NN consists of a small number of fully connected layers with a smooth activation function φ and linear sum in the final layer. For example, an architecture consisting of four hidden layers is given by

$$\tilde{u}(x; \theta_{\tilde{u}}) = W_5\varphi\Big(W_4\varphi\Big(W_3\varphi\Big(W_2\varphi\Big(W_1 x + b_1\Big) + b_2\Big) + b_3\Big) + b_4\Big) + b_5. \tag{2}$$

In this work we focus on $\varphi(\cdot) = \tanh(\cdot)$, $\sin(\cdot)$ (SIREN) or $GeLU(\cdot)$ Hendrycks and Gimpel (2016) as non-linear activations. The input samples are given as $x \in \mathbb{R}^d$, with one input node for each dimension. The network is trained to satisfy the PDE requirements with its matching boundary conditions through minimization of a loss function. We demonstrate our method with few examples: The Laplacian operator in one and two dimensions for rectangular, free-form and L-shape domains, and the Legendre's equation in one dimension.

4 Laplacian Operator

4.1 Single Known Eigenvalue

We first address the problem of finding a single eigenfunction $u(x)$ given its corresponding eigenvalue λ. We approximate $u(x)$ using a NN and optimize the following loss function:

$$\mathcal{F}_1\left(\tilde{u}(x;\theta_u)\right) = \alpha\|\mathcal{L}\tilde{u}\|_2^2 + \eta\|\mathcal{L}\tilde{u}\|_1 + \mu\|\mathcal{L}\tilde{u}\|_\infty + \delta\|\tilde{u} - u_0\|_{1,\partial\Omega} + \beta\Big|\|\tilde{u}\|_2^2 - c\Big| + \rho\|\theta_{\tilde{u}}\|_2^2, \tag{3}$$

where

$$\mathcal{L}u := Au + \lambda u,$$

and the Laplacian operator $A = \Delta$. The NN approximation of u is denoted by $\tilde{u}$ and the network parameters are given by $\theta_{\tilde{u}}$. Boundary conditions are defined as $u_0 := u|_{\partial\Omega}$. The constants $\alpha, \eta, \mu, \delta, \beta$ and ρ are positive real scalars.

The first three terms in (3) consist of the L_2, L_1 and L_∞ fidelity terms. The first term penalizes deviations from the differential equation in the L_2 or average sense. The second term encourages a robust solution, namely it enforces sparsity of the errors, and the latter promotes a *pointwise* solution such that the equation is satisfied for isolated points as well Bar and Sochen (2019, 2021). The fourth term imposes boundary conditions and the fifth normalizes the squared area of u to $c = 1$ due to scale invariance of the eigenfunction (since any (Cu, λ) is a valid eigenpair for $C \neq 0$). The last term is the standard weight decay term which stabilizes the network weights.

As a first example, we apply the Laplacian operator in 1D where $\lambda = 4$ and $u(0) = u(\pi/2) = 0$. The PDE is then given by,

$$\mathcal{L}u = u''(x) + 4u(x).$$

The normalized analytical solution is therefore

$$u(x) = \frac{2}{\sqrt{\pi}} \sin(2x).$$

Figure 1 demonstrates the outcome of the algorithm at iterations 1, 100 and 2500. As can be seen, the predicted outputs approach the $\sin(\cdot)$ function with Relative Mean Square Error (RMSE) = $6.52\text{e}{-4}$ and peak signal-to-noise ratio (PSNR) = 28.84.

4.2 Single Eigenpair for the Smallest Eigenvalue

Next, we address the case where the eigenvalue is not known in advance. We therefore limit ourselves to the smallest nontrivial eigenpair. This approach is analogous to the power method approach, where only one dominant eigenpair is found. Recall that the Rayleigh quotient is defined as Miller (2016); Feld et al. (2019)

$$R(u) := -\frac{\langle Au, u\rangle}{\langle u, u\rangle}. \tag{4}$$

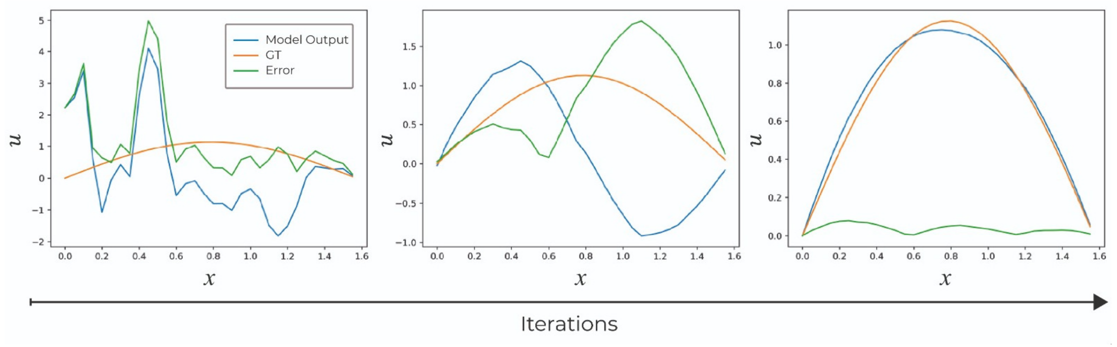

Figure 1: The solution to $u'' + 4u = 0$ with $u(0) = u(\pi/2) = 0$ at iterations (from left to right) 1, 100 and 2500. At the last iteration RMSE = $6.52\text{e}{-4}$ and PSNR = 28.84.

It can be shown that the eigenfunction u is a critical point of $R(u)$, where $R(u)$ is its corresponding (nontrivial) eigenvalue λ. Furthermore, if $\hat{u}$ is a function which is close to u, then $R(\hat{u})$ approximates λ. In the following loss function we replace λ by $R(u)$. The loss defined in (3) is then rewritten as

$$\begin{aligned}\mathcal{F}_2(\tilde{u}(x;\theta_u)) = \alpha\|\mathcal{L}\tilde{u}\|_2^2 + \eta\|\mathcal{L}\tilde{u}\|_1 + \mu\|\mathcal{L}\tilde{u}\|_\infty + \delta\|\tilde{u} - u_0\|_{1,\partial\Omega} \\ + \beta\Big|\|\tilde{u}\|_2^2 - c\Big| + \rho\|\theta_u\|_2^2 + \gamma\|R(\tilde{u})\|_2^2,\end{aligned} \tag{5}$$

where

$$\mathcal{L}u = Au + R(u)u.$$

The last term of (5) minimizes $R(\tilde{u})$ and therefore pushes the solution to the lowest nontrivial eigenvalue. The ground truth eigenpair is given by $\left(\sqrt{\frac{2}{\pi}}\sin(x), 1\right)$ for $\Omega = [0,\pi]$. Figure 2 shows the outcome of the proposed method at three iterations. The eigenvalue represented as the value of the Rayleigh quotient converges to the true value $\lambda = 1$ with decreasing standard deviation along the mini-batches. Quantitative results are shown in the first row of Table 1, where PSNR stands for peak signal-to-noise ratio, RMSE stands for root-mean-square error, AE stands for Absolute Error and RE for

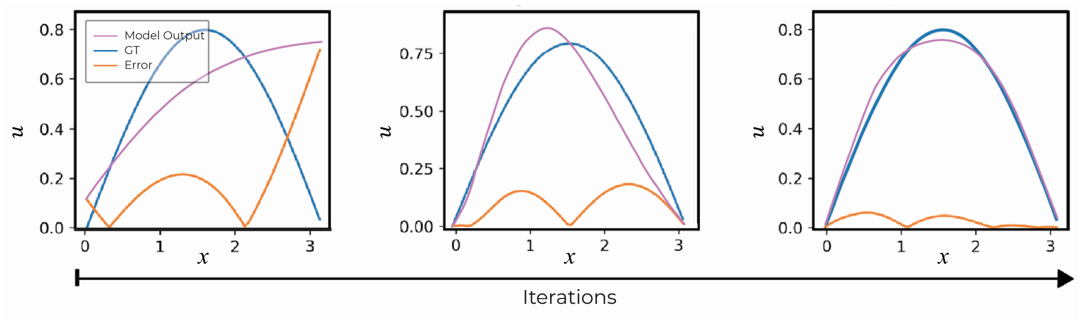

Figure 2: The solution to $u'' + \lambda u = 0$ with $u(0) = u(\pi) = 0$ at iterations 1, 500 and 1000. The associated predicted eigenvalues are $\lambda = 70 \pm 53,\ 5.5 \pm 0.5,\ 1 \pm 0.03$. Note that the solution is up to a sign.

Relative Error. In the cases of multiple eigenpairs, we calculate the mean values over M.

4.3 Multiple Eigenpairs for the Smallest Eigenvalues

A generalization of Section 4.2 is to find M eigenpairs with the corresponding bottom-M eigenvalues simultaneously. Following Hait-Fraenkel and Gilboa (2019), and using the orthogonality property of the eigenfunctions, we optimize the following loss penalty:

$$\begin{aligned}\mathcal{F}_3(\tilde{\mathbf{u}}(x;\theta_u)) = \sum_{i=1}^{M} \Big(& \alpha \|\mathcal{L}\tilde{u}_i\|_2^2 + \eta \|\mathcal{L}\tilde{u}_i\|_1 + \mu \|\mathcal{L}\tilde{u}_i\|_\infty + \delta \|\tilde{u}_i - u_0\|_{1,\partial\Omega} + \\ & \beta \Big| \|\tilde{u}_i\|_2^2 - c \Big| + \gamma_i \|R\left(\tilde{u}_i\right)\|_2^2 \Big) + \rho \|\theta_u\|_2^2 + \nu \sum_{i<j} |\langle \tilde{u}_i, \tilde{u}_j \rangle| \,,\end{aligned} \tag{6}$$

where the last term is explicitly given by

$$\nu \sum_{k=1}^{N_s} \sum_{i=1}^{M} \sum_{j=i+1}^{M} |\tilde{u}_i(x_k)\tilde{u}_j(x_k)| \,,$$

with point set size N_s. For each eigenfunction $\tilde{u}_i$ we impose the L_2, L_1 and L_∞ terms, boundary conditions and normalization as before. The weight of the Rayleigh quotient is multiplied by $\gamma_i = 1/i$ as a method to induce monotonic penalty to λ_i. The last term enforces the orthogonality of distinct eigenfunctions.

4.3.1 One Dimensional Case

Figure 3 demonstrates the outcome of the algorithm in 1D. In this case we have one network with M output values, one for every eigenfunctuion. The ground truth eigenfunctions of the Laplacian with $u(0) = u(\pi) = 0$ are given by

$$u_n(x) = \sqrt{\frac{2}{\pi}} \sin(nx),\ \ \lambda_n = n^2,\ \ n = 1, 2, 3, \ldots \quad .$$

The left panel of Figure 3 shows the results for $M = 3$ and in the right for $M = 4$, where every color stands for a different eigenfunction. The right figure in the right panel shows the convergence of the four eigenvalues (epoch stands for iteration) which as expected have the values 1, 4, 9 and 16. Quantitative results are summarized in Table 1.

4.3.2 Two Dimensional Case

Next, we tested the method in two-dimensions where we trained M different networks simultaneously, each with a single output, one for each eigenfunction. We found this architecture adequate for the multiple eigenpairs problem in the 2D case. The ground truth solution is then

$$u_{nm}(x, y) = \frac{2}{\pi} \sin(nx) \sin(my),\ \ \lambda_{nm} = n^2 + m^2,\ \ n, m = 1, 2, 3 \ldots$$

for $\Omega = [0, \pi]^2$ and $u(x, 0) = u(x, \pi) = u(0, y) = u(\pi, y) = 0$. Figure 4 shows the results at different iterations for $M = 1$. The expected eigenvalue is the lowest one ($n =$

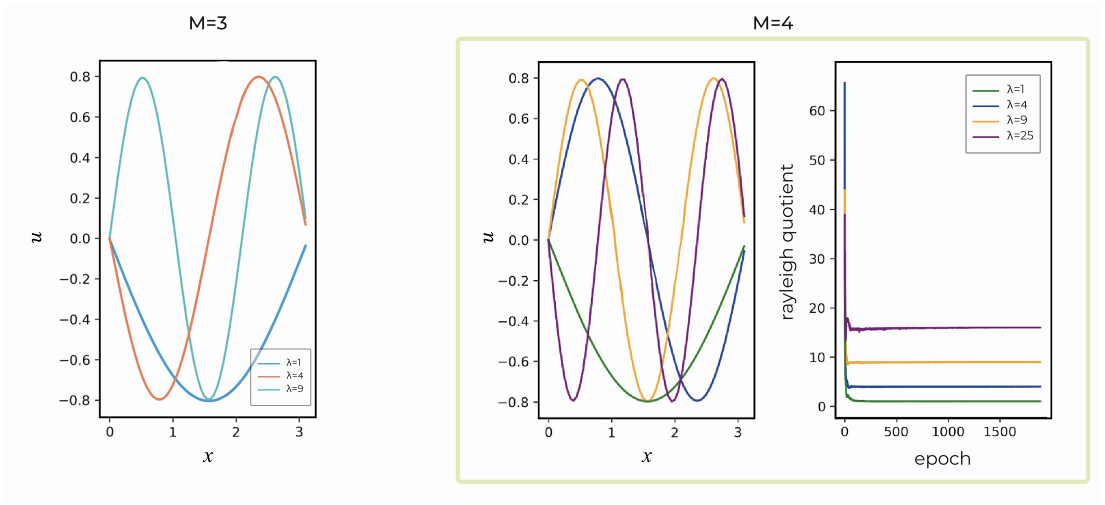

Figure 3: Left panel: eigenfunctions for $M = 3$ where every eigenfunction has a different color. Right panel: eigenfunctions for $M = 4$ are shown on the left and the convergence of the eigenvalues are shown on the right. Clearly the convergence of the functions are $u_n(x) = \sqrt{\frac{2}{\pi}}\sin(nx),\ \lambda_n = n^2$. $u_n(x)$ is up to a sign.

Table 1: Eigenpairs of the 1D Laplacian operator for different values of M. The ground truth eigenvalues are given by $(1, 4, 9, 16)$. Performance measures are averaged over M.

	λ	**Mean over efs.**		**Mean over evs.**	
M	GT = (1, 4, 9, 16)	PSNR	RMSE	AE	RE
1	1.02	43.49	4.50e-5	0.02	0.02
2	1.11, 4.09	56.58	2.21e-6	0.10	0.07
3	1.08, 8.93, 4.13	52.21	8.54e-6	0.09	0.04
4	1.12, 4.09, 9.03, 15.95	48.78	2.05e-5	0.07	0.04

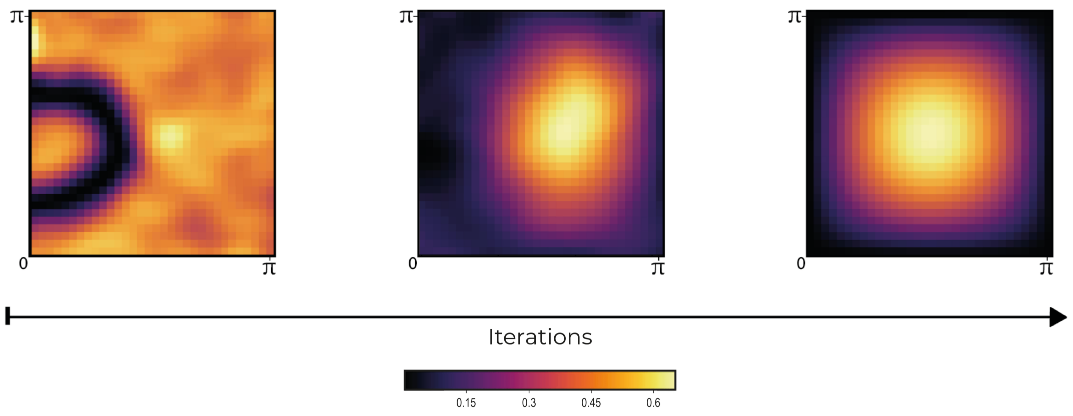

Figure 4: The solution to the 2D Laplacian eigevalue problem, with $M = 1$ at iterations (from left to right) 1, 100 and 1000.

$1, m = 1)$. As the algorithm converges, the expected eigenfunction is clearly seen (right image). We further tested our performance for $M = 4$, see Figure 5. The figures from left to right stand for the four eigenfunctions. The rows from top to bottom are iterations 1, 100 and 1000 respectively. As can be easily shown, these results are with accordance with the theoretical eigenfunctions $(n, m) = (1, 1), (1, 2), (2, 1), (2, 2)$. Quantitative results are summed up in Table 2.

4.4 Free-Form Domain

As the proposed method does not depend on the discretization of the domain, it can be easily adapted to free-form domains. We demonstrate it with the following two examples.

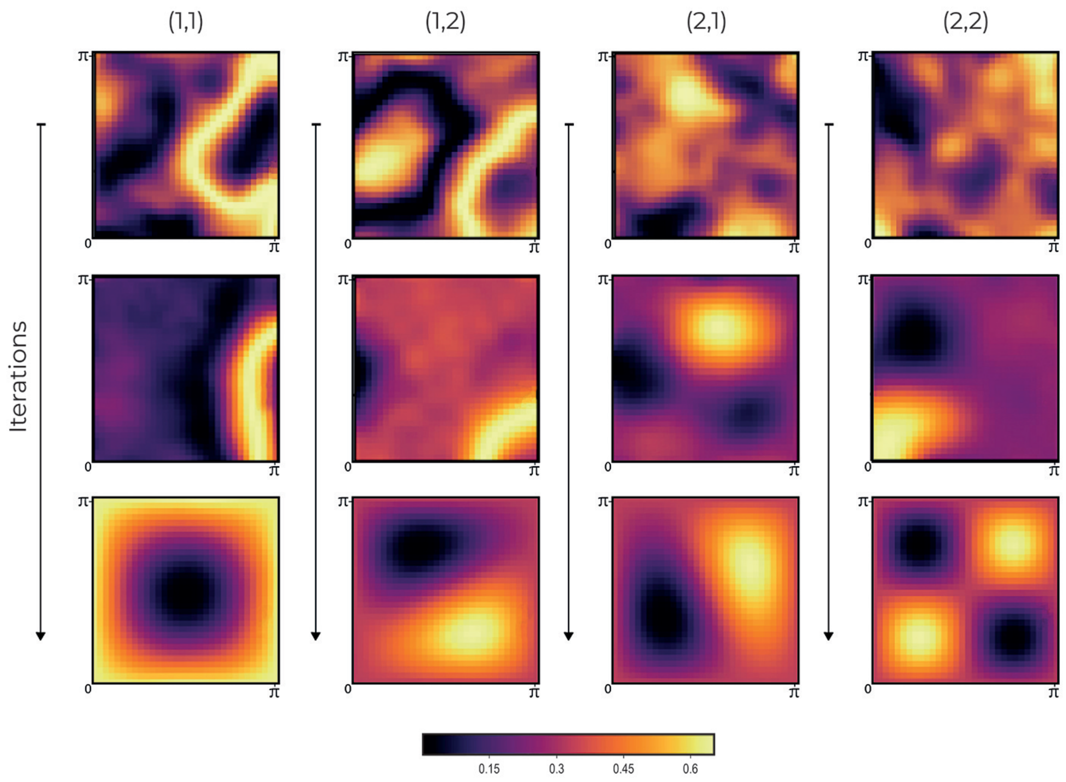

Figure 5: The solution to the 2D Laplacian eigenvalue problem for $M = 4$. The iterations $1, 100, 1000$ are shown from top to bottom for each of the 4 eigenfunctions.

Table 2: Eigenpairs of the 2D Laplacian operator for different values of M. The ground truth eigenvalues are given by $(2, 5, 5, 8)$. Performance measures are averaged over M.

	λ	**Mean over efs.**		**Mean over evs.**	
M	GT = (2, 5, 5, 8)	PSNR	RMSE	AE	RE
1	2.01	34.87	3.2e-4	0.01	0.005
2	1.99, 4.91	32.15	9.1e-4	0.05	0.01
3	1.94, 5.08, 4.93	28.49	0.002	0.07	0.02
4	1.98, 4.93, 4.96, 7.91	31.69	0.001	0.05	0.01

4.4.1 Circular Cut

For $M = 1$, we used the same square domain, with a circular piece removed. We tested three different sizes of circles as can be seen in Figure 6. The left column is the outcome of the algorithm inferred at the whole rectangle. The ground truth is on the right column and is as for the full shape. The error is depicted in the central column. It is easy to see that the error concentrates on the missing regions, and is extrapolated in a smooth fashion.

4.4.2 L-shaped Domain

In the next example, we obtained the 2D Laplacian eigenpairs with Dirichlet boundary conditions applied on an L-shape domain. Our results were compared to the Finite Element method using the Partial Differential Equation toolbox MATLAB (2015). We used a fine mesh such that the maximum edge size was 0.05. The first four eigenvalues were compared to a tight approximation as presented by Yuan and He (2009), see Table 3. As can be shown, our results are comparable (even though slight lower) to the desired values. The eigenfunctions are depicted in Figure 7, and quantitative comparisons are summarized in Table 4, where the FEM was referred to as the ground truth.

5 Legendre's Differential Equation

We now demonstrate how the proposed method can be used to solve the well-known Legendre's differential equation:

$$\frac{d}{dx}\left[\left(1-x^2\right)\frac{dP_n(x)}{dx}\right] + n(n+1)P_n(x) = 0,$$

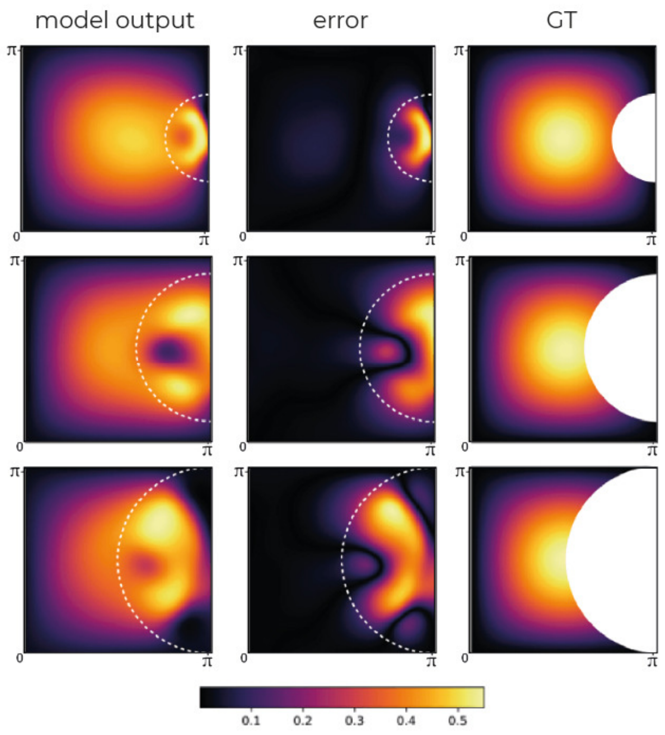

Figure 6: Free-form domain results for $M = 1$ excluding points in the right half circles. Left: The outcome of the proposed algorithm inferred in the full rectangle. The solution in the missing domain seems to be a smooth extrapolation. Middle: error from the full ground truth $|u - \text{full}(u_{\text{gt}})|$. The color bar is referred only to this column. Right: Ground Truth. It is clear that the error is condensed into the missing domain, where the information is missing.

Table 3: Eigenvalues of the Laplacian operator with L-shaped domain. The first column is a tight approximation of the ground truth. The second column is the outcome of the FEM method with maximum edge size of 0.05. The third column is the outcome of the proposed method.

λ Yuan and He (2009)	FEM MATLAB (2015)	proposed
9.639	9.649	8.626
15.739	15.202	14.711
19.739	19.746	19.288
29.522	29.538	28.901

Table 4: Comparison of the Eigenfunctions of the Laplacian operator with an L-shaped domain where the Finite Element method is referred to as ground truth.

	MSE	PSNR
1	0.002	26.267
2	0.002	27.233
3	0.003	25.247
4	0.003	25.323

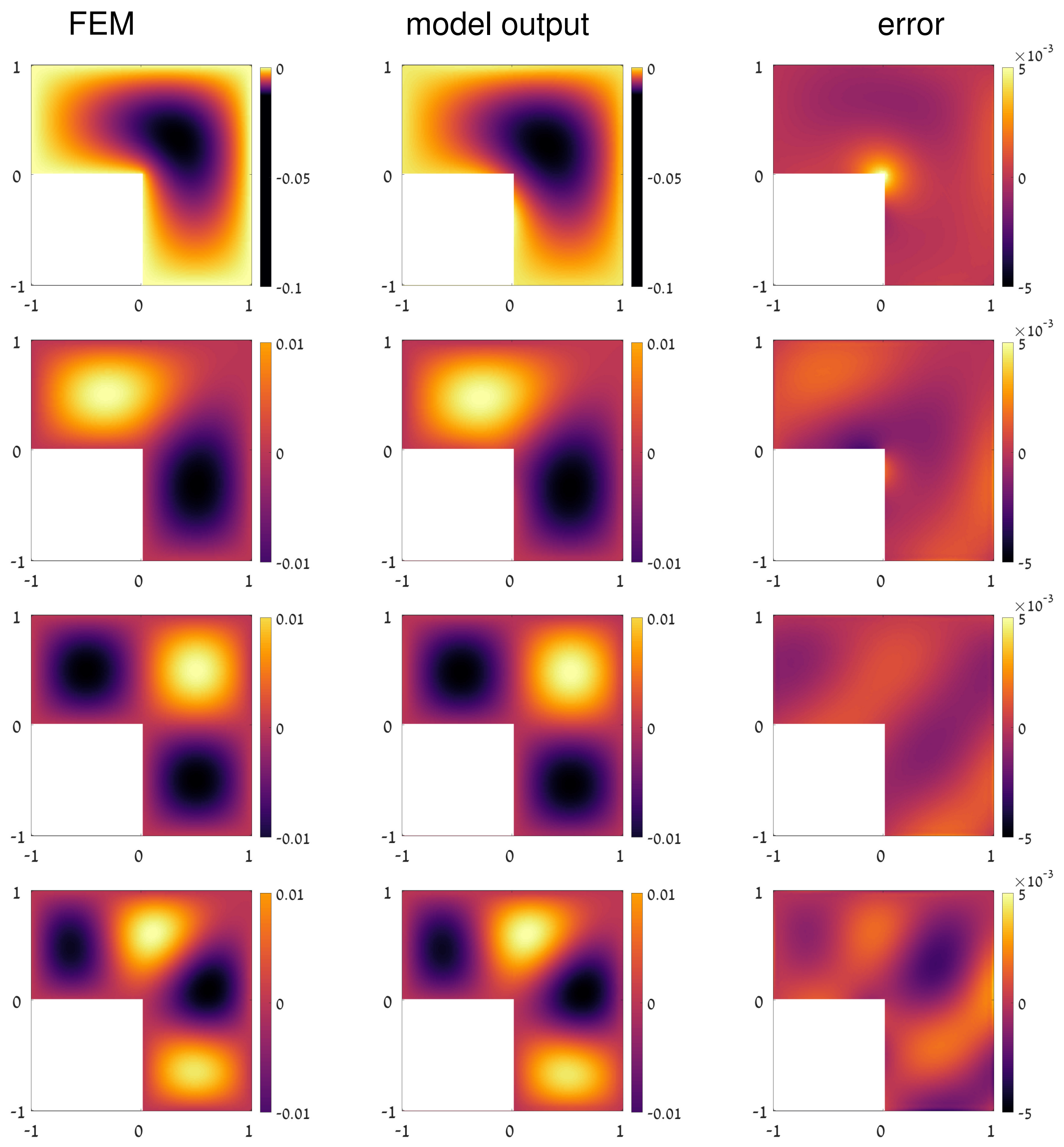

Figure 7: Eigenfunctions of the Laplacian operator of an L-shaped domain. Left: Finite Element Method, Middle: proposed method, right: The difference between them

which can be written as the following eigenvalue problem

$$\mathcal{T}P_n := BP_n + \lambda_n P_n,$$

where

$$B := \frac{d}{dx}\left[\left(1 - x^2\right)\frac{d}{dx}\right]$$

and

$$\lambda_n = n(n+1).$$

To find the M smallest eigenpairs, we deploy the same method as before, where in this case a different operator is used. Our network consists of 5-layers with SIREN activations, with a training set of 10,000 points on the interval $[-1, 1]$. The boundary conditions consist of a single point: $(1, 1)$. We add a regularization term of $||\nabla P_n(x)||^2$ to promote smoothness. Since in this case, each eigenfunction has a different energy, we use the following relation:

$$\int_{-1}^{1} P_m(x)P_n(x)dx = \frac{2}{2n+1}\delta_{mn}$$

to arrive at

$$E_n := \int_{-1}^{1} |P_n(x)|^2 dx = \frac{2}{2n+1}.$$

Therefore, given the fact that the Rayleigh quotient $R(P_n)$ approximates the sought eigenvalue, $R(P_n) \sim n(n+1)$, we have:

$$E_n = ||P_n||^2 \sim \frac{2}{\sqrt{1 + 4R(P_n)}}.$$

The loss function then takes the form

$$\begin{aligned}\mathcal{F}_4(\tilde{\mathbf{P}}(x;\theta_p)) = \sum_{n=1}^{M} \Big(&\alpha\|\mathcal{T}\tilde{P}_n\|_2^2 + \eta\|\mathcal{T}\tilde{P}_n\|_1 + \mu\|\mathcal{T}\tilde{P}_n\|_\infty + \xi\|\nabla\tilde{P}_n(x)\|_2^2 \\ &+ \delta|\tilde{P}_n(1) - 1| + \beta\left|\|\tilde{P}_n\|_2^2 - \frac{2}{\sqrt{1+4R(\tilde{P}_n)}}\right| + \gamma_n\|R(\tilde{P}_n)\|_2^2\Big) \\ &+ \rho\|\theta_p\|_2^2 + \nu\sum_{n<m}\left|\langle\tilde{P}_n,\tilde{P}_m\rangle\right|.\end{aligned} \tag{7}$$

We report the results for $M = 3,\dots,6$ in Table 5. Some examples of eigenpairs are shown in Figure 8.

Neural Networks have a tendency of instability, especially when encountered with noise. Improving network robustness is an active area of research Zheng et al. (2016). Monte-Carlo Dropout(MC-DO) Gal and Ghahramani (2015) was recently introduced to quantify network confidence, by adding dropout Srivastava et al. (2014) during inference and running the model for many iterations. We run MC-DO on the model trained for $M = 6$ with 5,000 iterations and dropout rate $q = 0.01$, and show the confidence intervals in Figure 9 as the std of the outputs. The outputs are computed for eigenvalues: $\lambda = 0, 12, 20$. From this analysis, we can see that the model is robust to the proposed noise, as the mean prediction across the iterations does not vary largely from the ground truth. It is also apparent that for $\lambda = 0$, and near the boundary, the model is more certain. The mean std for $\lambda = 0, 12, 20$ are $0.0022, 0.0136$ and 0.0141 respectively. This suggests that for functions of higher frequencies, the model will be less confident. For each eigenfunction, we also show the points with the highest variance (larger than the 65^{th} percentile. We see that these points concentrate around the highest frequencies of the eigenfunctions. Future work might leverage such methods to sample difficult areas during training Kendall et al. (2018).

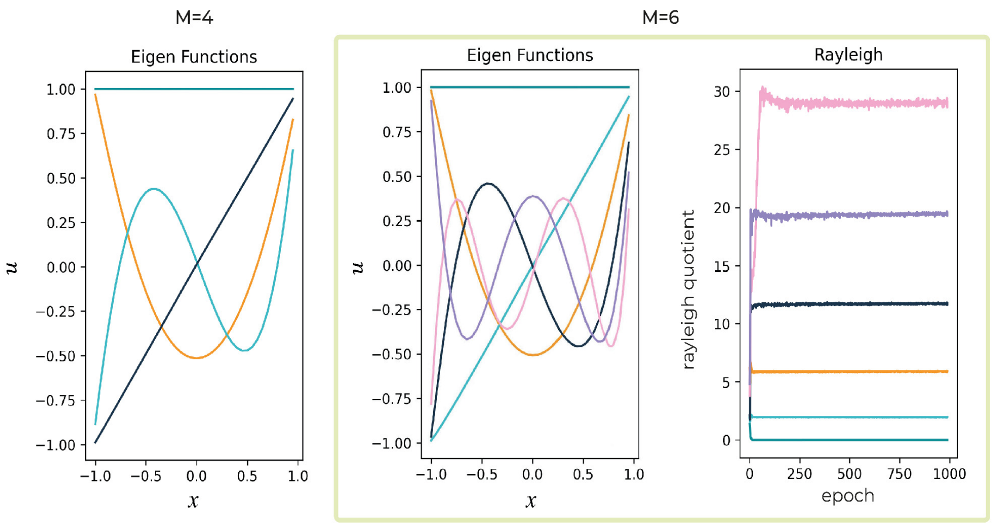

Figure 8: Learned solutions to Legendre's differential equation using the proposed eigenvalue method. Left: Approximated eigenfunctions for $M = 4$, Middle: Approximated eigenfunctions for $M = 6$. Right: convergence of the eigenvalues based on the Rayleigh value.

Table 5: Eigenpairs of Legendre's equation for different values of M. The ground truth eigenvalues are given by $(0, 2, 6, 12, 20, 30)$. Performance measures are averaged over M.

	λ			**Mean over efs.**		**Mean over evs.**	
M	GT = $(0, 2, 6, 12, 20, 30)$			PSNR	RMSE	AE	RE
3	5.13e-05	1.96	5.86	33.48	8.7e-4	0.09	0.02
4	-3.96e-05	1.96	5.84	34.34	7.1e-4	0.19	0.03
	11.63						
5	7.62e-05	1.97	5.89	31.42	3.2e-3	0.1	0.01
	11.74	20.00					
6	3.94e-06	1.97	5.89	33.74	9.4e-4	0.41	0.02
	11.71	19.48	28.98				

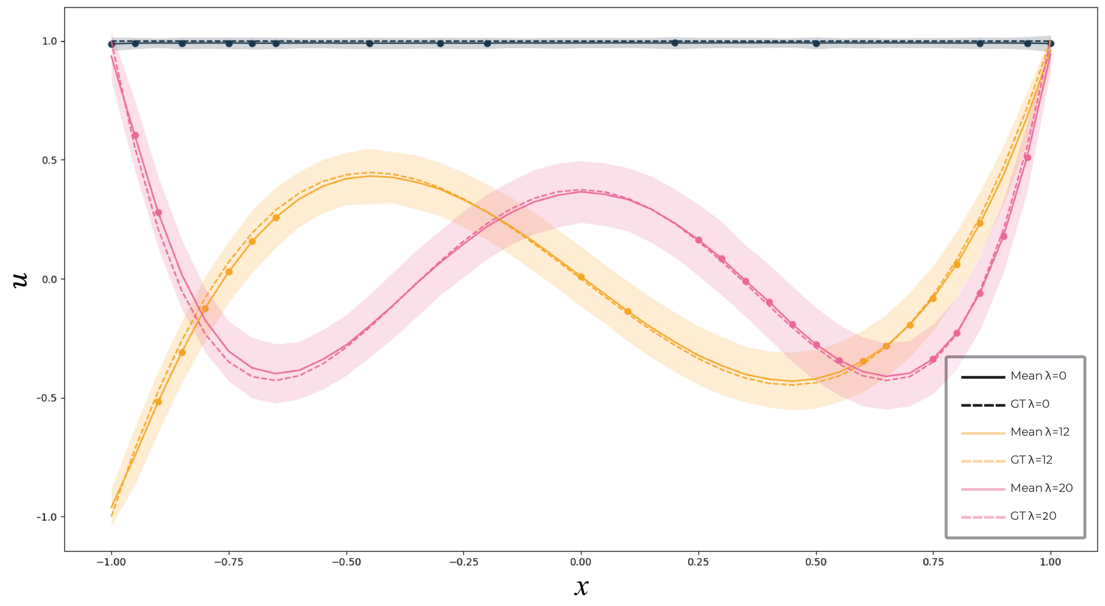

Figure 9: The eigenfunctions associated with $\lambda = 0, 12, 20$ obtained by Monte-Carlo Dropout with Dropout probability of $q = 0.01$ and $5,000$ MC-iterations.

6 Comparison to Other Methods

Before getting to the details of the comparison to other methods, few remarks should be made. Note that most methods in numerical solutions to PDEs start by a *discretization of the differential operator and the eigenfunctions*, transforming thus the problem to that of eigenvalues and eigenvecors of a matrix. There is an inherent error in the discretization of the problem which our proposed method overcome. The output of most methods is the eigenvalues and the corresponding eigenvectors which are supposed to be a discretized version of the eigenfunctions. In our proposed method, the operator is not discretized. The derivatives are analytically computed with no approximation and the solution is a set of eigenvalues and their corresponding (non-discretized) eigenfunctions.

The only method, to date, that provides eigenfunctions as the output of a learned Neural Network is the Spectral Inference Network (SpIN) Pfau et al. (2019a). The comparison details are presented in this Section. We also compare to the more traditional methods like the Krylov-Schur method. Both methods calculate multiple eigenpairs. For completeness, we added a comparison of the proposed method with the inverse power method in the Appendix. In this method we calculate only the *dominant* eigenfunction and the eigenvalue is given by the corresponding Rayleigh quotient.

6.1 Krylov-Schur Method

The first comparison consists of the Krylov-Schur algorithm for finding few eigenpairs of a large matrix Stewart (2001); Lehoucq et al. (1998). By this method the continuous operator was approximated by a finite differences scheme to form the 2D Laplacian and 1D Legendre matrices with spacing h. The approximation of the continuous operator, therefore, may yield significant numerical errors Knyazev (2000). In addition, the incorporation of the boundary condition is not straightforward since the boundary conditions may affect the construction of the matrix approximation of the operator. As in the inverse power method, we had to manually normalize the Krylov-Schur eigenfunctions. The eigenpairs were calculated by MATLAB (2015) solver with different spacing $h = 0.1$, 0.05, 0.025 and 0.017. Quantitative results of the 2D Laplacian are shown in lines 1-4 in Table 6. Note that grid refinement ($h < 0.025$) does not improve the results anymore. Eigenpairs of the 1D Legendre's equation are summarized in Table 7. As can be seen, the results are getting better as h decreases (number of grid points N increases). The proposed algorithm outperforms in eigenfunctions measures of the

2D Laplacian operator. The Krylov-Schur method works better for the 1D Legendre's case.

6.2 Spectral Inference Networks

Spectral Inference Networks (SpIN) Pfau et al. (2019a) is a framework for learning multiple eigenfunctions of linear operators by stochastic optimization. In this method, the function u is parameterized by a Neural Network. To find the bottom-M eigenpairs, the Rayleigh quotient is rewritten in a special matrix form and minimized by a bilevel optimization. Unlike our approach, the differential operator in SpIN is approximated in a *discrete* fashion. Additionally, like in Krylov-based methods, integrating boundary conditions is not straightforward. In the solution of of the Schrödinger equation, the network output was multiplied by some function $f(x)$ such that the boundary conditions will be satisfied (Appendix C.1 of Pfau et al. (2019a)). We tested the eigenpairs of the Laplacian operator with 16000 sampling points where this number was manually optimized. The results are shown in line 5 of Table 6. Our method outperformed in eigenfunctions accuracy, while the Krylov-Schur method with $h = 0.025$ gave the best eigenvalues.

7 Implementation Details

Our network was constructed as a dense fully connected network with 5-7 hidden layers architecture, each with a varying number of neurons, from 26-100. Our code was implemented in PyTorch. Each of our models was trained for 5000 iterations on GeForce RTX

Table 6: Eigenpairs of the 2D Laplacian operator. The ground truth eigenvalues are $(2, 5, 5, 8)$. N stands for the number of points. In the Krylov-based method we specify the discretization spacing by h. Performance measures are averaged over M.

			λ	**Mean (efs.)**		**Mean (evs.)**	
Method	h	N	GT = (2, 5, 5, 8)	PSNR	RMSE	AE	RE
Krylov	0.1	1K	1.92, 4.81, 4.81, 7.68	15.17	0.12	0.19	0.038
	0.05	4K	1.98, 4.96, 4.96, 7.95	18.10	0.12	0.03	0.006
	0.025	16K	1.98, 4.97, 4.97, 7.95	21.09	0.12	**0.03**	**0.005**
	0.017	34K	1.97, 4.93, 4.93, 7.89	22.77	0.12	0.065	0.013
SpIN		16K	2.06, 5.69, 5.86, 8.32	21.46	0.04	0.488	0.097
Proposed		40K	1.98, 4.93, 4.96, 7.91	**22.88**	**0.024**	0.05	0.011

3080 Graphics card. In the 2D case all the networks were trained simultaneously. For the activation function, since we are modeling smooth functions, we found that *ReLU* is less suitable both theoretically and experimentally. The activations functions used were *SIREN* Sitzmann et al. (2020), *tanh*, and *GeLU* Hendrycks and Gimpel (2016). The *SIREN* activation has been shown to excel in modeling complex signals, and their higher-order derivatives. We used an Adam optimizer Kingma and Ba (2014) with default parameters. We found that the weight initialization Katanforoosh and Kunin (2019) was important for convergence. In our 1D experiments a Gaussian initialization with 0.0 mean and 1.0 std was used. When using the *SIREN* activation, we used the standard initialization proposed in Sitzmann et al. (2020). For the L-shape experiments, we used a Gaussian initialization with 0.0 mean and 0.7 std. For all experiments our

Table 7: Eigenpairs of the Legendre's equation. The ground truth eigenvalues are $(0, 2, 6, 12, 20, 30)$. N stands for the number of points. In the Krylov-based method we specify the discretization spacing by h. Performance measures are averaged over M. The Eigenpair ($\lambda = 0, u = 1$) was excluded from the calculation.

			λ			**Mean (efs.)**		**Mean (evs.)**	
Method	h	N	GT = (0,2,6,12,20,30)			PSNR	RMSE	AE	RE
Krylov	0.05	40	4.96e-14	1.94	5.54	20.83	0.011	1.10	0.07
			10.86	18.25	27.91				
	0.01	200	4.57e-13	1.98	5.90	30.82	0.0018	0.54	0.03
			11.66	19.21	28.55				
	0.005	400	7.07e-13	1.99	5.95	36.69	0.00406	0.28	0.01
			11.83	19.60	29.23				
	0.002	1000	6.47e-12	1.99	5.98	**44.83**	**8.0e-5**	**0.11**	**0.01**
			11.93	19.84	29.70				
Proposed		1000	3.94e-6	1.9	5.89	33.74	9.0e-4	0.41	0.02
			11.71	19.48	28.98				

weight decay used was $\rho = 1\mathrm{e}{-8}$. The weighing between the different eigenvalues was set to $\gamma_i = \frac{1}{i}$. The remaining coefficients and hyperparameters used are given in tables 8 and 9. In the one-dimensional case, the boundary conditions had two actual points. In the two-dimensional case, points were taken along the boundary of the square. Instead of using an exact $\|\mathcal{L}u\|_\infty$, we used a relaxed approximation: $\frac{\mu}{K}\sum_{k\in \mathrm{top}_K(|L_i|)}|L_k|$, with K as a coefficient for broader effect of the L_∞ fidelity term.

Table 8: Hyper parameters used for each of the experiments. The columns describe: Learning Rate, Batch-Size, Number of neurons at every layer, Number of points inside the domain, Number of boundary points, Number of layers, and Activation Function used.

Task	**lr**	**BS**	**N**	$\lvert\Omega\rvert$	$\lvert\partial\Omega\rvert$	**#Layers**	**Activation**
1D Example	3e-3	2048	50	42K	1K	5	Tanh
Rayleigh 1	3e-3	2048	50	42K	1K	5	Tanh
2D_1	3e-3	256	100	5K	500	7	SIREN
2D_2	3e-3	256	100	5K	500	7	SIREN
2D_3	3e-3	256	100	5K	500	7	SIREN
2D_4	3e-3	256	100	5K	500	7	SIREN
L-Shaped	8e-4	64	26	42K	1K	5	GeLU
Legendre	3e-3	512	26	10K	100	5	SIREN

Table 9: Loss coefficients using parameters defined in the loss functions along the paper.

Task Name	**K**	η	α	μ	β	δ	γ	ξ
1D Example	40	0	0.1	0.1	1.5	0.5	0.003	0.0
Rayleigh 1	40	0	0.3	0.3	1.5	1	0.003	0.0
2D_1	10	0.5	0.5	0.5	10	10	0.01	0.0
2D_2	10	0.5	0.5	0.5	10	10	0.01	0.0
2D_3	10	0.5	0.5	0.5	10	10	0.01	0.0
2D_4	10	0.5	0.5	0.5	10	10	0.01	0.0
L-Shaped	40	0.5	5	5	35	15	3e-05	0.0
Legendre	40	0	1	1	20	20	0.003	0.2

8 Convergence Analysis

In this section we analyze the error in the Neural Network approximation of the differential equation. Neural Network (NN) is a graphic representation of a specific parametric function. For smooth / ReLU activation function, the network is an embedding of a finite dimensional class of smooth / piecewise linear (PL) functions in the infinite dimensional functional space of smooth / PL functions. Given a fully connected network with a given depth, any function in the space of all smooth / PL functions can be approximated by a NN which is wide enough. In fact, one layer with an infinite width is a universal approximator (see Hornik et al. (1989)) which is analogous to the space of all polynomials. Nevertheless, the large dimension of the weight space together with the non-convexity of the loss function, makes the global optimization of the problem non attainable and only local minima can be guaranteed. For review on these issues please refer to Ding et al. (2020). In view of this situation, we concentrate first on a slightly different issue, the apparent error. Indeed, whenever the network is converging, one has a direct access to the value of the various terms in the loss function. In particular to the first two terms where the error in the equation in the L_2 sense and in the L_∞ sense are given empirically. While these values may be the outcome of a local minimum, it is still very useful to provide a bound on the error in the approximation of the eigenfunctions (or eigenpairs), given the empirical error of the equation i.e. the truncation error. We apply this idea to several classes of the forthcoming equations. After analyzing the error of the eigenfunction approximation, we attempt to link this analysis to other works Ohn and Kim (2019); Gühring and Raslan (2021), where the architecture of the network is related to the approximation error of the function.

Definition 8.1. The operator

$$\mathcal{L}u = \sum_{i,j} a_{i,j}(x)\frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_i b_i(x)\frac{\partial u}{\partial x_i} + c(x)u \tag{8}$$

is **elliptic** at a point $x \in \Omega$ if there exists $\lambda(x) > 0$ and $\Lambda(x)$, such that

$$0 < \lambda(x)\sum_i |\xi_i|^2 \leq \sum_{i,j} a_{i,j}(x)\xi_i\xi_j \leq \Lambda(x)\sum_i |\xi_i|^2, \tag{9}$$

for $(\xi_1, \xi_2, \ldots, \xi_n) \in \mathrm{R}^n \setminus (0)$.

If $\lambda(x) \geq \lambda > 0$, for all $x \in \Omega$, then the operator is called **strictly elliptic**. We quote Theorem 3.7 form Gilbarg and Trudinger (1998).

Theorem 8.1. *Let $\mathcal{L}u = F$ in a bounded domain Ω, where $\mathcal{L}$ is elliptic, $c \leq 0$ and $u \in C^0(\bar{\Omega}) \cap C^2(\Omega)$. Here $\bar{\Omega} = \Omega \cup \partial\Omega$. Then*

$$\sup_{\Omega} |u| \leq \sup_{\partial\Omega} |u| + C \sup_{\Omega} \frac{|F|}{\lambda}, \tag{10}$$

where $C = C(\Omega, \beta)$ is a constant, depending strictly on the diameter of Ω and on

$$\beta = \sup_{\Omega} \frac{|\mathbf{b}|}{\lambda},$$

where $|\mathbf{b}| = \sqrt{\sum_i |b_i|^2}$ (see Definition (8.1)).

Remark 8.2. *In particular, if Ω lies between two parallel planes at distant ℓ apart, then (10) is satisfied with $C = e^{(\beta+1)\ell} - 1$.*

When the condition $c \leq 0$ is not satisfied, it is still possible to assert an apriori bound analogous to (10) provided the domain Ω lies between sufficiently close parallel planes. We quote Corollary 3.8 form Gilbarg and Trudinger (1998).

Corollary 8.3. *Let $\mathcal{L}u = F$ in a bounded domain Ω, where $\mathcal{L}$ is elliptic and $u \in C^0(\bar{\Omega}) \cap C^2(\Omega)$. Let C be the constant of Theorem 8.1, $c^+ := \max(c, 0)$, and suppose that*

$$C_1 = 1 - C \sup_{\Omega} \frac{c^+}{\lambda} > 0. \tag{11}$$

Then

$$\sup_{\Omega} |u| \leq \frac{1}{C_1} \left(\sup_{\partial\Omega} |u| + C \sup_{\Omega} \frac{|F|}{\lambda} \right). \tag{12}$$

We also quote the following remark.

Remark 8.4. *Since $C = e^{(\beta+1)\ell} - 1$ is a possible value of the constant in (10), where ℓ is the width of any slab containing Ω, condition (11) will be satisfied in any sufficiently narrow domain in which the quantities $|\mathbf{b}|/\lambda$ and c/λ are bounded from above.*

We turn now the deriving a bound on the error $u - \tilde{u}$ for two differential problems. In Section 8.1 we consider the Divergence-form equation and in Section 8.2 we consider the Helmholz Equation.

8.1 Bounds for Divergence-Form Equation

Let u is a solution of the divergence-form problem Bar and Sochen (2021)

$$\mathcal{L}_1 u = F(x), \quad x \in \Omega \subset \mathbb{R}^n, \tag{13}$$

where

$$\mathcal{L}_1 u = \nabla \cdot (G(x)\nabla u)$$

satisfying the boundary condition

$$u(x) = u_0, \quad x \in \partial\Omega. \tag{14}$$

Let $\tilde{u}$ be its approximation. Then the error $e_u = u - \tilde{u}$ satisfies

$$\mathcal{L}_1 e_u = \tau_L, \quad x \in \Omega \subset \mathbb{R}^n, \tag{15}$$

where τ_L is the truncation error. The boundary condition for e_u is

$$e_u(x) = 0, \quad x \in \partial\Omega. \tag{16}$$

Here Ω is a bounded domain with smooth boundary $\partial\Omega$ and $G(x)$ is a smooth function satisfying

$$|G(x)| \geq l_b > 0, \quad |G(x)| \leq l_u, \quad |\nabla G(x)| \leq C_2, \qquad \forall x \in \Omega. \tag{17}$$

Lemma 8.5. *The operator $\mathcal{L}_1$ is* **strictly elliptic** *in Ω.*

Proof. Here

$$a_{i,i} = G(x), \qquad i = 1, \ldots, n$$

$$a_{i,j} = 0, \qquad i, j = 1, \ldots, n, \quad i \neq j, \tag{18}$$

$$b_i = \frac{\partial G(x)}{\partial x_i}, \qquad c = 0.$$

Therefore,

$$\sum_{i,j} a_{i,j}(x)\xi_i\xi_j = G(x)\sum_i |\xi_i|^2. \tag{19}$$

Using our assumptions, we have that $0 < l_b \leq |G(x)| \leq l_u$ in Ω. Thus,

$$0 < l_b \sum_i |\xi_i|^2 \leq \sum_{i,j} a_{i,j}(x)\xi_i\xi_j \leq l_u \sum_i |\xi_i|^2, \tag{20}$$

for (ξ_i, ξ_j) in $\mathbb{R}^n \setminus (0)$ and $x \in \Omega$. Therefore

$$\lambda \geq l_b = \min_{\Omega} |G(x)|, \quad \Lambda \leq l_u = \max_{\Omega} |G(x)|. \tag{21}$$

□

Now, we apply Theorem 8.1 for the error $e_u = u - \tilde{u}$.

Theorem 8.6. *Let Ω be a bounded domain with smooth boundary $\partial\Omega$. Assume that $u \in C^0(\bar{\Omega}) \cap C^2(\Omega)$ is the solution of*

$$\nabla \cdot (G(x)\nabla u) = F(x), \quad x \in \Omega, \tag{22}$$

satisfying the boundary condition

$$\tilde{u}(x) = u(x) = u_0, \quad x \in \partial\Omega. \tag{23}$$

Assume that $G(x)$ is smooth function satisfying

$$|G(x)| \geq l_b > 0, \qquad |G(x)| \leq l_u, \qquad |\nabla G(x_1, x_2)| \leq C_2 \quad \forall x \in \Omega. \tag{24}$$

Assume also that the truncation error defined as $\tau_L := \mathcal{L}(u - \tilde{u})$ satisfies

$$|\tau_L| \leq \varepsilon_0, \qquad \forall x \in \Omega. \tag{25}$$

Then,

$$\boxed{\sup_{\Omega} |e_u| \leq C\frac{\varepsilon_0}{l_b}}, \tag{26}$$

where $C = C\big(\Omega, \sup_{\Omega} |\nabla G|/l_b\big)$ is a constant, depending only on Ω and $\sup_{\Omega} |\nabla G|/l_b$.

Proof. We have to show that our problem satisfies the assumption in Theorem 8.1. In Lemma 8.5 we have shown that the operator $\mathcal{L}u = \nabla \cdot (G(x)\nabla u)$, with the conditions (24) is strictly elliptic, thus elliptic. In addition, in our case $c = 0$, thus satisfies the condition $c \leq 0$. We have assumed that the solution u of the problem (22)-(23)-(24) is in $C^0(\bar{\Omega}) \cap C^2(\Omega)$. Then, using Theorem 8.1 for $e_u = u - \tilde{u}$, we may conclude that

$$\sup_{\Omega} |e_u| \leq \sup_{\partial\Omega} |e_u| + C \sup_{\Omega} \frac{|\tau_L|}{\lambda}. \tag{27}$$

Since $e_u = 0$ on $\partial\Omega$, then the first term in (27) vanishes. As for the second term in (27), we have from assumption (24) that $\lambda \geq l_b > 0$ and $|\mathbf{b}| = \sqrt{\sum_i |b_i|^2} = |\nabla G|$. Thus,

$$\beta = \sup_{\Omega} \frac{|\mathbf{b}|}{\lambda} = \sup_{\Omega} \frac{|\nabla G|}{\lambda} = \frac{C_2}{l_b}. \tag{28}$$

Therefore, C, appearing in Theorem 8.1 depends only of Ω and $\beta = C_2/l_b$. By assumption (25)

$$\sup_{\Omega} |\tau_L| \leq \varepsilon_0, \tag{29}$$

we conclude that

$$\sup_{\Omega} |e_u| \leq C\big(\Omega, C_2/l_b\big)\, \varepsilon_0, \tag{30}$$

where $C_2 = \sup_{\Omega} |\nabla G|$. $\square$

8.2 Bounds for the Helmholtz Equation

The Helmholtz equation which is eigenvalue problem of the Laplacian operator is given by

$$\mathcal{L}_2 u = F(x), \qquad x \in \Omega$$

where

$$\mathcal{L}_2 u = \nabla^2 u + k^2 u, \qquad k > 0. \tag{31}$$

Lemma 8.7. *The operator $\mathcal{L}_2$ is strictly elliptic.*

Proof. Here

$$a_{i,i} = 1, \qquad a_{i,j} = 0, \quad i \neq j$$

$$b_i = 0, \qquad c = k^2. \tag{32}$$

Therefore,

$$\sum_{i,j} a_{i,j}(x)\xi_i\xi_j = \sum_i |\xi_i|^2. \tag{33}$$

Thus,

$$0 < \sum_i |\xi_i|^2 = \sum_{i,j} a_{i,j}(x)\xi_i\xi_j = \sum_i |\xi_i|^2, \tag{34}$$

for (ξ_i, ξ_j) in $\mathbb{R}^n \setminus (0)$ and $(x) \in \Omega$. Therefore,

$$\lambda = 1, \quad \Lambda = 1. \tag{35}$$

□

Consider now the Helmholtz problem, for which we derive a bound for $e_u = u - \tilde{u}$.

Theorem 8.8. *Let Ω be a bounded domain with a smooth boundary $\partial\Omega$. Assume that $u \in C^0(\bar{\Omega}) \cap C^2(\Omega)$ is the solution of*

$$\nabla^2 u + k^2 u = F(x), \quad x \in \Omega, \tag{36}$$

satisfying the boundary condition

$$u(x) = u_0, \quad x \in \partial\Omega. \tag{37}$$

Suppose that $\tau_L = \mathcal{L}_2(u - \tilde{u})$ satisfies

$$|\tau_L| \leq \varepsilon_0, \qquad \forall x \in \Omega. \tag{38}$$

Assume that the domain Ω lies between two parallel planes a distant ℓ apart, such that

$$\ell < \ln\left(1 + \frac{1}{k^2}\right). \tag{39}$$

Then,

$$\boxed{\sup_{\Omega} |e_u| \le C_2\, \varepsilon_0}. \tag{40}$$

Proof. We have to show that our problem satisfies the assumption in Corollary 8.3. In Lemma 8.7 we have shown that the operator $\mathcal{L}_2 u = \nabla^2 u + k^2 u$ is strictly elliptic, thus elliptic. Note that in our case $c = k^2 > 0$.

Using Remark 8.4, where in our case $\beta = 0$, we may choose $C = e^{\ell} - 1$ as a possible value of the constant in (10), where ℓ is the width of any slab containing Ω. Then condition (11) will be satisfied in any sufficiently narrow domain in which the quantities $|\mathbf{b}|/\lambda$ and c/λ are bounded from above. In our case $|\mathbf{b}|/\lambda = 0$ and $c/\lambda = k^2$, which are both bounded from above.

In particular, in order to satisfy condition (11), since $C = e^{\ell} - 1$ and $\sup_{\Omega} c^+/\lambda = k^2$, we need to require that

$$1 - (e^{\ell} - 1)k^2 > 0, \tag{41}$$

thus we have to require that

$$\ell < \ln\left(1 + \frac{1}{k^2}\right). \tag{42}$$

Then, we conclude that

$$\sup_{\Omega} |e_u| \le C_2\, \varepsilon_0 = \frac{C}{C_1}\varepsilon_0 = \frac{e^{\ell} - 1}{1 - k^2}\, \varepsilon_0, \tag{43}$$

where $C_1 = 1 - \sup_{\Omega} c^+/\lambda = 1 - k^2$. □

Inversly we can learn from this result that given a domain Ω such that the minimal slab in which it can be embedded has distance ℓ then the maximal eigenfunction that one can hope to recover corresponds to $|k| \leq \frac{1}{\sqrt{e^{\ell}-1}}$.

8.3 Relation to the Network's Structure

In this subsection we relate the empirical truncation error to the network structure and parameters. For this aim we bound the truncation by the network approximation error and bound the latter in terms of the network's structure.

Let $v \in W^{s,p}(\Omega)$ be a function, with continuous derivatives up to order s, and $\tilde{v}$, the approximation of v due to the neural network. First, we state a theorem, which relates the architecture of the network with the error $v - \tilde{v}$ in the Sobolev space $W^{s,p}(\Omega)$.

Let the network approximation error of a function v be defined as

$$e_v := v - \tilde{v}.$$

This error was addressed by Shen et al. (2020, 2021); Lu et al. (2020), where they presented an optimal error characterization of deep ReLU and floor-ReLU networks for smooth and Hölder functions in terms of network architecture (network depth and length). Ohn and Kim (2019) derived the required depth, width and sparsity of deep neural networks to approximate any Hölder smooth function up to a given approximation error. They address two classes of activation functions, which include most of the commonly used activation functions. The first class is the piecewise linear activation functions which includes ReLU and Leaky ReLU. The second class includes locally quadratic activation functions, where there is an interval on which the activation function has nonzero curvature, e.g. sigmoid, $tanh$, soft clipping, soft plus, and Exponential linear unit.

Gühring and Raslan (2021) recently introduced necessary and sufficient complexity of neural networks to approximate functions in Sobolev spaces for a wide class of activation functions: leaky ReLU, ELU, softsign, sigmoid, tanh, arctan, softplus, swish and RepU. Jiao et al. (2021) provided a rigorous numerical analysis on deep Ritz methods for second order elliptic equations with Dirichlet, Neumann and Robin boundary conditions, respectively. They also provided a corollary for the main result of Gühring and Raslan (2021) which we use below.

Theorem 8.9. *Let* $\Omega := [0,1]^d, \mathcal{F}_{s,p,d} = \{v \in W^{s,p}(\Omega) : \|v\|_{W^{s,p}(\Omega)} \leq 1\}$. *Let* $p \geq 1$, $s, k, d \in \mathbb{N}^+$, $s \geq k+1$ *and* $\mu > 0$ *a small scalar. Let* φ *be logistic or tanh activation functions. Assume that* $\tilde{v}$ *is the approximation of* v *by the neural network. For any* $\varepsilon_0 > 0$ *and* $v \in \mathcal{F}_{s,p,d}$*, there exists a neural network with depth* $C \log(d+s)$ *and* $N = C(d,s,p,k)\varepsilon_0^{-d/(s-k-\mu k)}$*, where* N *is the number of non-zero weights, such that*

$$\|v - \tilde{v}\|_{W^{s,p}(\Omega)} \leq \varepsilon_0.$$

Proof. Proposition 4.8 in Gühring and Raslan (2021) and Remark 4.1 in Jiao et al. (2021). □

Next, we relate the error $\|v - \tilde{v}\|_{W^{s,p}(\Omega)}$ to the truncation error in a linear differential operator $\mathcal{L}$, with bounded coefficients, operating on a smooth function v. We define the norm of the truncation error by

$$T_L(v) := \|\tau_L(v)\|.$$

Corollary 8.10. *Let $\tilde{v} \in W^{s,p}(\Omega)$ be an approximation of $v \in W^{s,p}(\Omega)$ such that $\|v - \tilde{v}\|_{W^{s,p}(\Omega)} \leq \varepsilon_0$ then the truncation error of the operator $\mathcal{L}$ satisfies*

$$T_L(v) = \|\mathcal{L}(v - \tilde{v})\|_\infty \leq C_0\varepsilon_0, \tag{44}$$

where C_0 depends on the coefficients of the derivatives of v appearing in the differential operator $\mathcal{L}$.

Proof. By the general Sobolev inequality (Theorem 6 in Evans (2010)), in the case that $sp > d$, if one chooses $p = d = 2$, then, as a result of Theorem 6, we have

$$(v - \tilde{v}) \in C^{s-2,\gamma}(\Omega)$$

where $0 < \gamma < 1$, for $s > 2$, and

$$\|v - \tilde{v}\|_{C^{s-2,\gamma}} \leq C_0\|v - \tilde{v}\|_{W^{s,2}(\Omega)}.$$

Therefore for $s > 4$,

$$\|v^{(m)} - \tilde{v}^{(m)}\|_\infty \leq C\|v - \tilde{v}\|_{W^{s,2}(\Omega)} \leq C\varepsilon_0,$$

for $m = 0, 1, \ldots, s-3$. Therefore, all derivatives up to order $s-3$, appearing in the differential equations of order $s-3$, are approximated with errors which are bounded by ε_0 in the infinity norm. Since the coefficients are bounded as well, it follows that the truncation error in the operator $\mathcal{L}$, defined by T_L, satisfies

$$T_L(v) = \|\mathcal{L}(v - \tilde{v})\|_\infty \leq C_0\varepsilon_0. \tag{45}$$

□

The result of this section and the previous one is that the approximation error and the truncation error are linearly dependent. It follows that given the network's structure and the empirical truncation error, we can bound the behavior of the approximation error.

8.3.1 Practical Architecture Analysis

In this subsection we display the truncation error $T_L = \|\mathcal{L}(u - \tilde{u})\|_\infty$ with respect to the network structure. In our problems $\mathcal{L}u = 0$, therefore

$$T_L = \|\mathcal{L}\tilde{u}\|_\infty.$$

We solve the 1D Laplacian eigenproblem with $M = 1$ for different architectures: the number of layers were set to $(2, 3, 4)$ with $(15, 20, 25, 30, 50)$ neurons per layer (width). Given that neural networks are stochastic processes, the optimization plays an integral role in the results. As such, we ran each experiment with 5 different initialization seeds, and took the average scores. We do not include in the plot the runs which had a loss over a certain threshold of 10.0 as to not include experiments which did not converge. As expected, the error generally decreases as the number of weights increases. Of note, for each depth, we see a general decrease in the error as the width increases.

The Laplacian operator is a special case of a Divergence-form equation, and therefore, by (26), if $\|\mathcal{L}\tilde{u}\|_\infty \leq C_0\varepsilon_0$, then $\|u - \tilde{u}\|_\infty \leq C\varepsilon_0$. Figure 10 shows the relation between the total number of weights N, and the truncation error $T_L = \|\mathcal{L}\tilde{u}\|_\infty$, both in a log scale. We fit the results to a linear curve using least squares. This curve is in accordance to Theorem 8.9 and (26), where

$$N = C_0\varepsilon_0^{-\tau},$$

which yields

$$\log N = -\tau \log \varepsilon_0 + C,$$

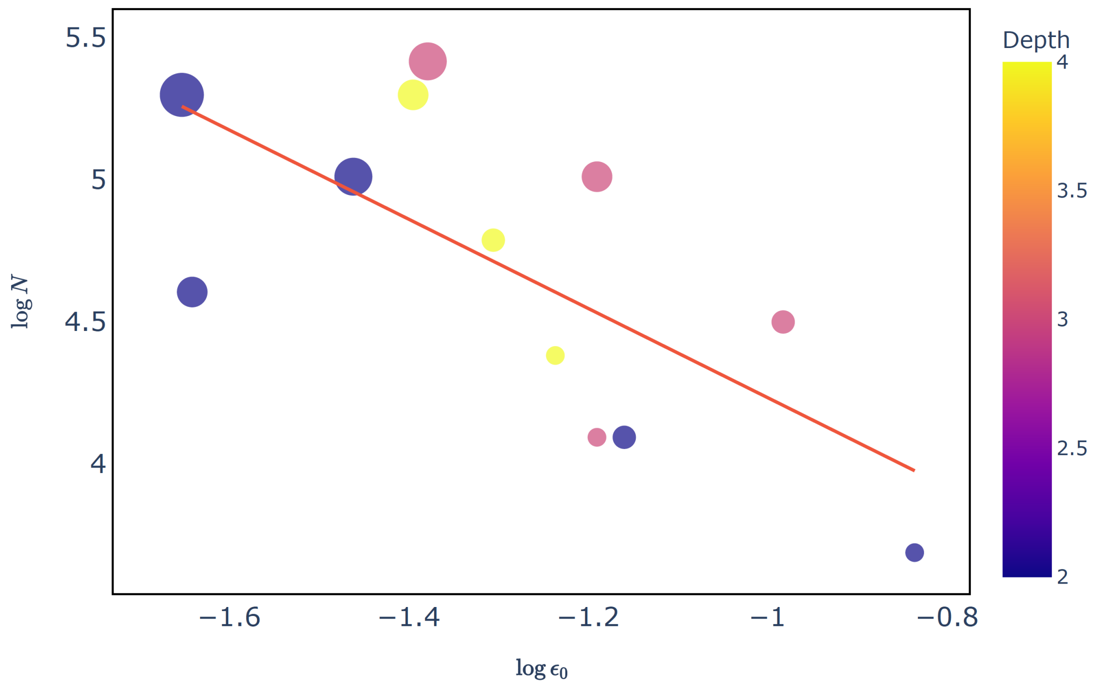

Figure 10: Empirical dependency between the truncation error $\|\mathcal{L}\tilde{u}\|_\infty$ and the total number of weights. The marker size represents the layer width: $(15, 20, 25, 30, 50)$. The relation is in accordance with Theorem 8.9, where $\log N = -\tau \log \varepsilon_0 + C$. In our case, $\tau = 1.56$ and $C = 2.66$.

with $\tau = d/(s - k(1 + \mu))$. In our case $d = 1$. Since the solution is given as a $\sin(\cdot)$ function, then $s = 5, k = 4$, and μ may be chosen as 0.09. Note, that we have picked $s = 5$, which satisfy our theoretical requirement $s > 4$. For these values, $\tau = 1/(5 - 4(1 + 0.09)) = 1.56$. Notice that as the smoothness s of the function increases, then the number of total weights decreases. The *convergence rate* is then given by

$$\varepsilon_0 = C_1 N^{-1/\tau} \approx C_1 N^{-0.64}.$$

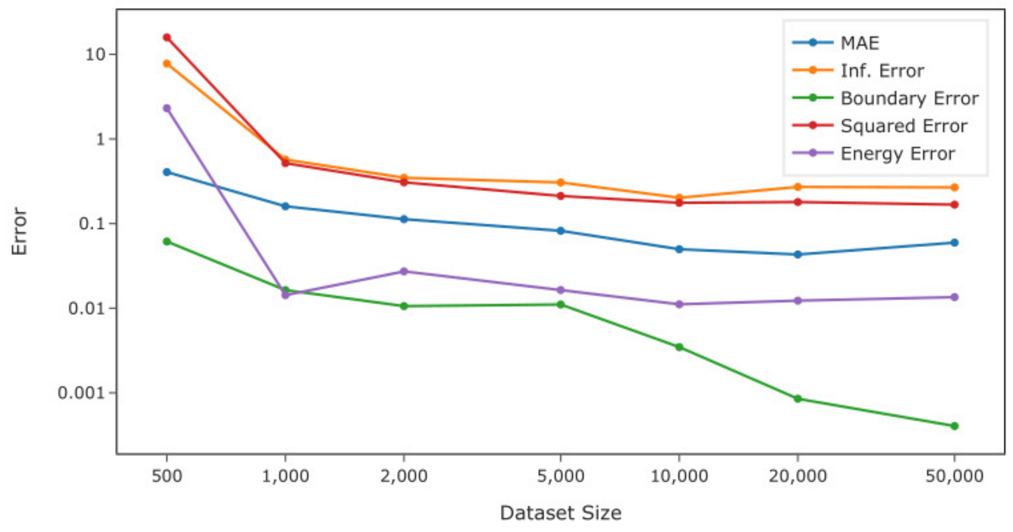

Figure 11: Empirical dependency between the approximation performance and the dataset size on the 1D Laplacian eigenvalue problem with $M = 1$, as discussed in Section 4.2. For each dataset size we analyze over 6 different initialization seeds, and average the results. The results are given in a $\log - \log$ plot. Formally: MAE = $\|u - \tilde{u}\|_1$, Inf. Error = $\|Lu - L\tilde{u}\|_\infty$, Boundary Error = $\|u_0 - \tilde{u}\|_1$, Squared Error = $\|Lu - L\tilde{u}\|_2$, Energy Error = $|\|\tilde{u}\|_2^2 - c|$.

8.4 Dependence on Dataset Size

Classical methods such as FEM Pradhan and Chakraverty (2019), are based on a mesh with interval h. Generally speaking, the number of points $N \sim V_\Omega / h^d$ where V_Ω is the volume of Ω, and d is the dimension. We give an example of the 1D Laplace equation, and test different dataset sizes N in Figure 11 in log-log scale. Each dataset size is averaged across 6 initialization seeds. For each N, we show the different forms of loss functions. We see that generally, all errors decrease as more samples are used.

9 Conclusion

In this work we introduced a NN-based method for the eigevalue problem. We have shown that a NN can be devised to solve several PDEs of the same class simultaneously, and demonstrated our approach for the class of eigenfunctions of a given elliptic operator. We have further verified that we can approximate the eigenfunctions and the eigenvalues (eigenpairs) at the same time. We proposed a novel method to exploit these new innovations, along with specific constraints that exist in the mathematical nature of such problems that shed some light on the well-known eigenvalue problem.

Our method is applicable for solving eigenvalue problem on complex domains, not necessarily on standard domains like rectangles or circles. Furthermore, the proposed method may be used on higher order, non-linear operators and on higher dimensional manifolds. When using the Rayleigh quotient to learn both the eigenfunction and its corresponding eigenvalue, the end-to-end learning process seems to accelerate the convergence. This suggests that the orthogonality term serves as a regularizer on each eigenpair in the course of the training process. The method produces *smooth* solutions, due to the fact that the networks are composed of linear layers and smooth activation functions (namely *tanh*, *SIREN* and *GeLU*). Since these solutions are deterministic and parameterized by the learned weights, they are infinitely differentiable. The fact that the solution is given in an explicit form, enables an exact analytical differentiation, and therefore is applicable for high order PDEs.

The suggested approach can be further generalized and optimized. The computation time is still slower than classical methods like FEM. This optimization is a significant task for future work. Another research direction might refer the boundary conditions as an input to the network. With this generalization, new training is not necessary for a new boundary condition. Further directions include high dimensional problems, non-linear differential operators, and non-uniform point set sampling.

References

Bar, L. and Sochen, N. (2019). Unsupervised deep learning algorithm for PDE-based forward and inverse problems. *arXiv 1904.05417v1*.

Bar, L. and Sochen, N. (2021). Strong solutions for pde-based tomography by unsupervised learning. *SIAM Journal on Imaging Sciences*, 14(1):128–155.

Benouhiba, N. and Belyacine, Z. (2013). On the solutions of the (p,q)-laplacian problem at resonance. *Nonlinear Analysis*, 77.

Bozorgnia, F. (2016). Convergence of inverse power method for first eigenvalue of p-laplace operator. *Numerical Functional Analysis and Optimization*, 37.

Bronson, R., Costa, G. B., and Saccoman, J. T. (2014). Chapter 4 - eigenvalues, eigenvectors, and differential equations. In *Linear Algebra (Third Edition)*, pages 237 – 288. Academic Press, third edition edition.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. In *Advances in Neural Information Processing Systems 31*, pages 6571–6583.

Choo, K., Mezzacapo, A., and Carleo, G. (2019). Fermionic neural-network states for ab-initio electronic structure. *arXiv 1909.12852*.

Conway, J. B. (1985). *A Course in Functional Analysis*. Springer-Verlag New York.

Ding, T., Li, D., and Sun, R. (2020). Sub-optimal local minima exist for neural networks with almost all non-linear activations. *arXiv:1911.01413v3*.

Eastman, S. and Estep, D. (2007). A power method for nonlinear operators. *Applicable Analysis*, 86(10):1303–1314.

Erickson, M. A., SMITH, R. S., and LAUB, A. J. (1995). Power methods for calculating eigenvalues and eigenvectors of spectral operators on hilbert spaces. *International Journal of Control*, 62(5):1117–1128.

Evans, L. C. (2010). *Partial Differential Equations*. American Mathematical Society.

Feld, T., Aujol, J. F., Gilboa, G., and Papadakis, N. (2019). Rayleigh quotient minimization for absolutely one-homogeneous functionals. *Inverse Problems*.

Gal, Y. and Ghahramani, Z. (2015). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *arXiv e-prints*, page arXiv:1506.02142.

Gilbarg, D. and Trudinger, N. S. (1998). *Elliptic Partial Differential Equations of Second Order*. Springer.

Gühring, I. and Raslan, M. (2021). Approximation rates for neural networks with encodable weights in smoothness spaces. *Neural Networks*, 134:107–130.

Hait-Fraenkel, E. and Gilboa, G. (2019). Numeric solutions of eigenvalue problems for generic nonlinear operators. *arXiv 1909.12775*.

Han, J., Lu, J., and Zhoh, M. (2020). Solving high-dimensional eigenvalue problems using deep neural networks: A diffusion Monte Carlo like approach. *arXiv 2002.02600v2*.

Han, J., Zhang, L., and Weinan, E. (2019). Solving many-electron Schrödinger equation using deep neural networks. *Journal of Computational Physics*.

Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv: Learning*.

Hermann, J., Schätzle, Z., and Noé, F. (2019). Deep neural network solution of the electronic Schrödinger equation. *arXiv 1909.08423*.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359—-366.

Jiao, Y., Lai, Y., Lo, Y., Wang, Y., and Yang, Y. (2021). Error analysis of deep Ritz methods for elliptic equations. *arXiv 2107.14478v2*.

Katanforoosh and Kunin (2019). Initializing neural networks.

Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv 1412.6980*.

Knyazev, A. (2000). Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM Journal on Scientific Computing*, 23(2).

Lehoucq, R. B., Sorensen, D. C., and Yang, C. (1998). *ARPACK User's Guide*. SIAM, Philadelphia, PA.

Lu, J., Shen, Z., Yang, H., and Zhang, S. (2020). Deep network approximation for smooth functions. *arXiv 2001.030402v2*.

MATLAB (2015). *version 8.6.0 (R2015b)*. The MathWorks Inc., Natick, Massachusetts.

Miller, G. (2016). *Spectral Graph Theory and The Laplacian Paradigm, Lecture 27*.

Ohn, I. and Kim, Y. (2019). Smooth function approximation by deep neural networks with general activation functions. *Entropy*, 21(7).

Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A., and Guibas, L. (2012). Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics*.

Pfau, D., Petersen, S., Agarwal, A., Barrett, D. G. T., and Stachenfeld, K. L. (2019a). Spectral inference networks: unifying deep and spectral learning. In *ICLR*.

Pfau, D., Spencer, J. S., de G Matthews, A. G., and Foulkes, W. M. C. (2019b). Solution of the many-electron Schrödinger equation with deep neural networks. *arXiv 1909.02487*.

Pradhan, K. K. and Chakraverty, S. (2019). Chapter four - finite element method. In *Computational Structural Mechanics*, pages 25 – 28.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017). Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv 1711.10561*.

Saad, M. H. (2005). 5 - formulation and solution strategies. In *Elasticity*, pages 83 – 102.

Saad, Y. (2011). *Numerical Methods for Large Eigenvalue Problems*. SIAM, Philadelfia.

Shen, Z., Yang, H., and Zhang, S. (2020). Deep network approximation characterized by number of neurons. *Communications in Computational Physics*, 28:1768–1811.

Shen, Z., Yang, H., and Zhang, S. (2021). Deep network with approximation error being reciprocal of width to power of square root of depth. *Neural Computation*.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22.

Sitzmann, V., Martel, J. N. P., Bergman, A. W., Lindell, D. B., and Wetzstein, G. (2020). Implicit Neural Representations with Periodic Activation Functions. *arXiv 2006.09661*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Stewart, G. W. (2001). A krylov-schur algorithm for large eigenproblems. *SIAM Journal of Matrix Analysis and Applications*, 23:601–614.

Yuan, Q. and He, Z. (2009). Bounds to eigenvalues of the laplacian on l-shaped domain by variational methods. *Journal of Computational and Applied Mathematics*.

Zheng, S., Song, Y., Leung, T., and Goodfellow, I. (2016). Improving the Robustness of Deep Neural Networks via Stability Training. *arXiv e-prints*, page arXiv:1604.04326.

Appendix

A Inverse Power Method

The *power iteration method* or *power method*, is a well known iterative algorithm where given a diagonalizable matrix A, finds the maximal eigenvalue and its corresponding eigenvector Bronson et al. (2014). Given a diagonalizable matrix A, the algorithm produces a number λ , which is the greatest (in absolute value) eigenvalue of A, and a nonzero vector v, which is the corresponding eigenvector of λ , that is, $Av = \lambda v$. The power iteration algorithm starts with a vector b_0, which may be an approximation to the dominant eigenvector or a random vector. Then at iteration k,

$$b_{k+1} = \frac{Ab_k}{\|Ab_k\|}.$$

If we assume A has an eigenvalue that is strictly greater in magnitude than its other eigenvalues, then b_k convergence to an eigenvector associated with the dominant eigenvalue. Ideally, one should use the Rayleigh quotient in order to get the associated eigenvalue,

$$\lambda = \frac{b_k^T A b_k}{b_k^T b_k}.$$

The *inverse power method* is the power method applied to the inverse of a matrix A. In general, the inverse power method converges to the *smallest* eigenvalue in absolute value of A, where

$$b_{k+1} = \frac{A^{-1}b_k}{\|A^{-1}b_k\|}.$$

We adopted the inverse power method to our problem following Bozorgnia (2016). The discrete 2D Laplacian operator was formulated as a convolution kernel

$$k = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix},$$

where the convolution operation was constructed as a $N^2 \times N^2$ block Toeplitz matrix, with $N = \pi/h$, where h is the discretization spacing. The approximation of the continuous operator, therefore, may yield significant numerical errors Knyazev (2000). Figure 12 shows the outcome of the inverse power method regarding the Laplacian operator implemented in Matlab MATLAB (2015) with $h = 0.05$ and initial eigenvalue

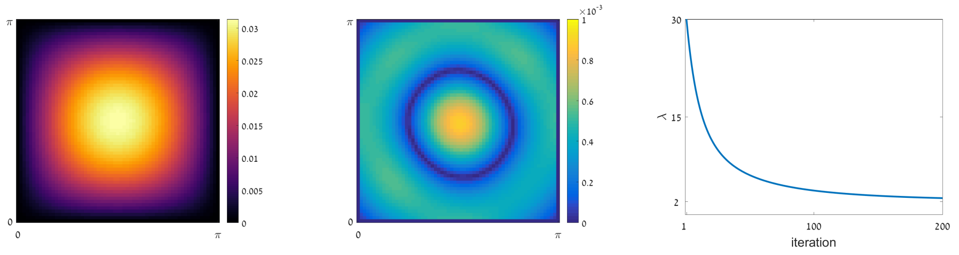

Figure 12: Inverse power method for the Laplacian operator. Left: Approximated eigenfunction, Middle: error $|u_{gt} - u|$. Right: convergence of the eigenvalue. $\lambda = 2.18$, RMSE=$1.60\mathrm{e}{-4}$, PSNR=37.95

Table 10: Laplacian operator: comparison to the Inverse Power Method. The ground truth eigenvalue is 2. PSNR and MSE are referred to the eigenfunctions.

	λ (GT=2)	PSNR	RMSE
Inverse power method	2.18	**37.95**	**1.6e-4**
Proposed (our)	**2.01**	34.87	3.2e-4

$\lambda_0 = 1$ in two dimensions. On the left is the estimated eigenfunction which corresponds to the lowest eigenvalue ($m = 1, n = 1$). In the middle is the error $|u_{gt}(x, y) - u(x, y)|$ and on the right is the convergence plot of the eigenvalue which was converged to 2.18. Table 10 shows the performance of the proposed algorithm compared with the inverse power method.

The discrete operator of the Legendre's equation was built as a multiplications of three matrices of size $N \times N$ with $N = 2/h$: the first is the forward difference operator matrix, the second is a diagonal matrix of vector of length N stands for $1 - x^2$, $x \in (-1, 1)$, and the third is the backward difference operator. The convergence of

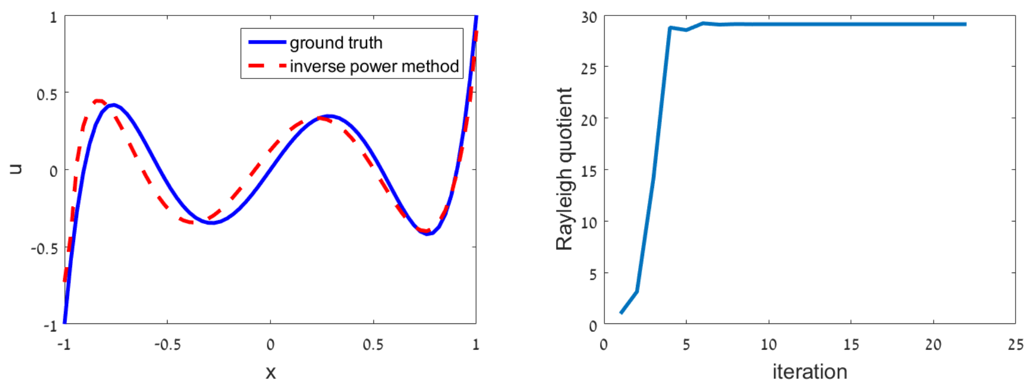

Figure 13: Inverse power method for the Legendre's equation. Left: Ground truth (solid line) and approximated (dashed line) eigenfunctions. Right: convergence of the eigenvalue with $\lambda = 29.12$, RMSE=1.77e-2, PSNR=17.51.

the algorithm required a normalization of the eigenfunctions such that the L_2 norm was $2/(2n+1)$. We provided this prior manually. We also added the boundary condition constraint. Note that in our method the normalization is automatic. We set $h = 0.03$ and initial eigenvalue $\lambda_0 = 1$. The results are shown in Figure 13 and Table 11. In our experiments, the inverse power method seemed to be sensitive to discretization and initialization. In particular, in the Legendre's equation, different values of h brought out different eigenpairs, where for $h = 0.03$ we received the fifth eigenvalue which converged to $\lambda = 29.12$. Our algorithm outperforms in part of the quantitative measures. Furthermore, the inverse power method finds only a *single* eigenpair, while the proposed method outputs the M smallest eigenpairs. Although it is possible to calculate more eigenpairs based on the previous one via orthogonality constraints, an undesired accumulated error may emerge.

Table 11: Legendre's equation: comparison to the Inverse Power Method where the true eigenvalue is 30. Note that in this method there was a manual normalization of the eigenfunctions, while in the proposed method the normalization was automatic. PSNR and MSE are referred to the eigenfunctions.

	λ (GT=30)	PSNR	RMSE
Inverse power method	29.12	17.51	1.77e-2
Proposed (ours)	28.98	**24.83**	**3.28e-3**

Acknowledgement

We thank Prof. Mataina Ben-Artzi for his help with pointing out for us the relevant references Evans (2010); Gilbarg and Trudinger (1998) concerning the error bounds in Section 8. We also thank Ron-El Azar for her help with the visualizations throughout this paper.