

# A PINNs approach for the computation of eigenvalues in elliptic problems

Julián Fernández Bonder<sup>1,2,\*</sup> and Ariel Salort<sup>3,†</sup>

<sup>1</sup>*Departamento de Matemática, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires*

<sup>2</sup>*Instituto de Cálculo UBA-CONICET, Buenos Aires, Argentina*

<sup>3</sup>*Universidad CEU San Pablo, Urbanización Montepríncipe, s.n. 28668, Madrid, Spain*

In this paper, we propose a method for computing eigenvalues of elliptic problems using Deep Learning techniques. A key feature of our approach is that it is independent of the space dimension and can compute arbitrary eigenvalues without requiring the prior computation of lower ones. Moreover, the method can be easily adapted to handle nonlinear eigenvalue problems.

## INTRODUCTION

The impact of artificial intelligence (AI) on everyday life has become undeniable in recent years. From smartphones to image recognition, from medical diagnostics to self-driving cars, machine learning—and deep learning in particular—has transformed the way we interact with technology.

This influence has also extended into mathematics and physics, where machine learning techniques are being applied to analyze complex phenomena that were traditionally approached using classical analytical methods. Among these, problems governed by partial differential equations (PDEs) are especially prominent, due to their central role in modeling physical systems across many disciplines.

Reflecting this trend, a growing number of software libraries and frameworks have been developed as educational and research tools for solving PDE-based problems in computational science and engineering; see, for instance, [5–7] and references therein.

In parallel, neural network-based approaches for approximating solutions to PDEs have gained increasing attention. These include successful applications to problems such as the Burgers, Eikonal, and heat equations [8], as well as linear diffusion equations in complex two-dimensional geometries [9].

A natural extension of this idea is the use of neural networks (NNs) as general-purpose function approximators for solving differential equations. A prototypical example arises in quantum mechanics, where one seeks to solve the eigenvalue problem

$$-\Delta u + V(x)u = Eu, \quad \text{in } \mathbb{R}^n, \quad (1)$$

with a given potential function  $V: \mathbb{R}^n \rightarrow \mathbb{R}$  and eigenvalue parameter  $E \in \mathbb{R}$ . The idea of employing NNs to approximate solutions of such equations dates back to the 1990s [1], but the field gained significant momentum with the introduction of *Physics-Informed Neural Networks* (PINNs) in [2], which triggered a wave of research and applications (see also [3, 4]).

Most early PINN-based work focused on source problems, where the term  $Eu$  in (1) is replaced by a known forcing function  $f$ . The eigenvalue problem itself has

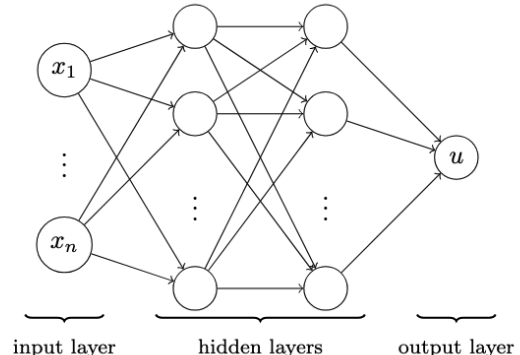


FIG. 1. A fully connected feedforward NN with an  $n$ -dimensional input and a 1-dimensional output.

been more recently addressed: first in one dimension [10, 11], and later in two dimensions [12]. Related approaches have also been proposed in [13], which include extensions to nonlinear eigenvalue problems.

In these works, the solution  $u$  is approximated using a fully connected feedforward neural network (see Fig. 1), with variations in the number of hidden layers and activation functions across implementations. To approximate the eigenvalue  $E$ , the authors minimize the *Rayleigh quotient*:

$$L(u) = \frac{\int |\nabla u|^2 + V(x)u^2 dx}{\int u^2 dx}.$$

Minimizing  $L(u)$  yields an approximation of the principal eigenvalue  $E_1$  and its associated eigenfunction  $u_1$ . Higher-order eigenpairs can be computed by iteratively minimizing the same functional while constraining the search to subspaces orthogonal to previously computed eigenfunctions.

This approach, while effective, presents two main limitations:

1. It is inherently sequential: to compute eigenvalues within a given range, one must first compute all lower ones, which can be inefficient.
2. Its extension to nonlinear problems (such as those involving the  $p$ -Laplace operator) is limited, since

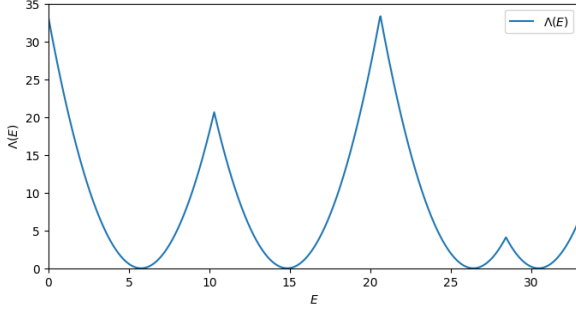


FIG. 2. The loss curve  $\Lambda(E)$  in 2-dimensional space with a confined potential in the unit disk.

orthogonality conditions between eigenfunctions are no longer available.

### OUR APPROACH

To overcome the drawbacks mentioned above, we propose a different approach to problem (1). We define the following loss function:

$$\Lambda(u, E) = \frac{\int (\Delta u - V(x)u + Eu)^2 dx}{\int u^2 dx}. \quad (2)$$

Given a fixed  $E \in \mathbb{R}$ , minimizing with respect to  $u$  yields a *loss curve*:

$$\Lambda(E) = \inf_u \Lambda(u, E). \quad (3)$$

A proof of the existence of a minimizer  $u_E$  for (3) is provided in Theorem 1 in the appendix.

It is relatively straightforward to derive a theoretical upper bound for the loss curve  $\Lambda(E)$ . Indeed, let  $u_k$  denote the  $k$ -th eigenfunction to (1), normalized such that  $\int u_k^2 dx = 1$ . Then we have:

$$\Lambda(E) \leq \int (\Delta u_k - V(x)u_k + Eu_k)^2 dx = (E_k - E)^2,$$

and consequently,

$$\Lambda(E) \leq \min_k (E_k - E)^2.$$

See Fig. 2, where this upper bound is plotted in the two-dimensional case with a confining potential given by  $V(x) = 0$  for  $|x| \leq 1$  and  $V(x) = \infty$  for  $|x| > 1$ . In this case the eigenvalues are known explicitly. The first four are:

$$E_1 = 5.7832, E_2 = 14.6819, E_3 = 26.3743, E_4 = 30.4713.$$

Our approach proceeds as follows:

We first fix the interval  $[E_*, E^*]$  where the eigenvalues are expected to lie. Then, we take a uniform partition of this interval:

$$E_* = E^1 < E^2 < \dots < E^j = E^*,$$

and for each  $E^i$  we train our neural network to minimize  $\Lambda(u, E^i)$ , obtaining a corresponding minimizer  $u^i$ .

Finally, we identify the minima of the loss curve  $\Lambda(E^i)$ . The values  $E^i$  at which these minima occur approximate the eigenvalues of (1), and the associated minimizers  $u^i$  approximate the corresponding eigenfunctions.

### IMPLEMENTATION DETAILS

We consider a fully connected feedforward neural network (as in Fig. 1) with two hidden layers and the hyperbolic tangent activation function,  $\tanh$ .

We begin by studying confining potentials as in the previous section:

$$V(x) = \begin{cases} 0 & \text{if } x \in \Omega \\ \infty & \text{if } x \notin \Omega \end{cases} \quad (4)$$

which is equivalent to solving the Helmholtz problem with homogeneous Dirichlet boundary condition:

$$\begin{cases} -\Delta u = Eu & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega. \end{cases} \quad (5)$$

To impose the boundary condition, we multiply the output of the NN by a boundary factor  $B(x)$  that is a smooth approximation to the distance to the boundary. To ensure normalization, we include a penalization term in the loss function to enforce  $\|u\|_2 = 1$ . The resulting loss is:

$$\bar{\Lambda}(u, E) = \int_{\Omega} (\Delta u + Eu)^2 dx + \mu \left( \int_{\Omega} u^2 dx - 1 \right)^2. \quad (6)$$

All integrals are approximated using a Monte Carlo sampling scheme.

As described in the previous section, we train the NN to minimize  $\bar{\Lambda}(u, E)$  for each value in a discrete partition  $E_* = E^1 < \dots < E^j = E^*$ . To improve efficiency, for each  $E^i$ , the network is initialized with the weights and biases obtained from training at  $E^{i-1}$ .

After training is complete, we obtain the discrete loss curve

$$\bar{\Lambda}(E^i) = \bar{\Lambda}(u^i, E^i)$$

and each local minimum below a given threshold  $\epsilon > 0$  is taken as an approximate eigenvalue for (5).

Fig. 3 shows the loss curve  $\bar{\Lambda}(E^i)$  in the case where  $\Omega$  is the unit disk in  $\mathbb{R}^2$ . The computed eigenvalues are accurate up to the resolution of the discretized  $E^i$ -grid. Naturally, once an approximate eigenvalue  $E^{i_0}$  is detected, the method can be refined locally by using a finer grid in the interval  $[E^{i_0-1}, E^{i_0+1}]$  to improve precision.

The associated computed eigenfunctions are displayed in Fig. 4

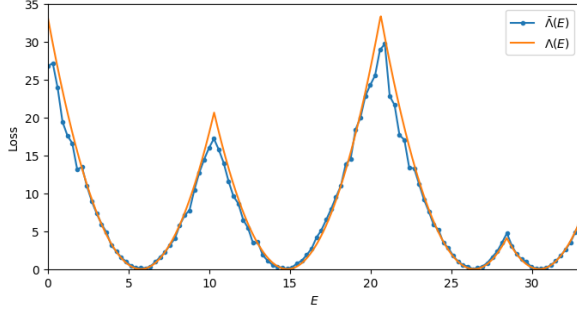


FIG. 3. The loss curve  $\tilde{\Lambda}(E)$  in two-dimensional space with a confined potential in the unit disk compared with the theoretical upper bound.

### FURTHER EXAMPLES

In this section, we illustrate the flexibility and robustness of our method by applying it to a variety of settings beyond the two-dimensional unit disk with a confining potential. These examples include the computation of higher-order eigenvalues, problems in higher spatial dimensions, domains with different geometries, and more general potential functions.

#### Higher eigenvalues

Our method enables the computation of higher-order eigenvalues without relying on sequential orthogonalization procedures. By choosing an appropriate interval  $[E_*, E^*]$  and refining the discretization grid, we can directly recover eigenvalues that are far from the principal one.

One technical point that must be considered when computing higher eigenvalues is the appropriate choice of the hyperparameter  $\mu$  in the loss function (6). For a randomly initialized NN output  $u$ , the first term in the loss is typically of the order

$$\int_{\Omega} (\Delta u + Eu)^2 dx \sim c_1 E^2 + c_2,$$

while the normalization penalty term is of the order

$$\mu \left( \int_{\Omega} u^2 dx - 1 \right)^2 \sim c_3 \mu.$$

Hence, if  $E^2 \gg \mu$ , the loss is dominated by the first term, and the network tends to minimize it by driving  $u \approx 0$ , which is undesirable. To maintain a proper balance between the two terms and ensure meaningful training, it is necessary to scale  $\mu \propto E^2$ .

To assess the accuracy of our method, we computed the eigenvalues of problem (5) in the two-dimensional unit square  $[0, 1] \times [0, 1]$  that lie within the interval [44, 55]. In

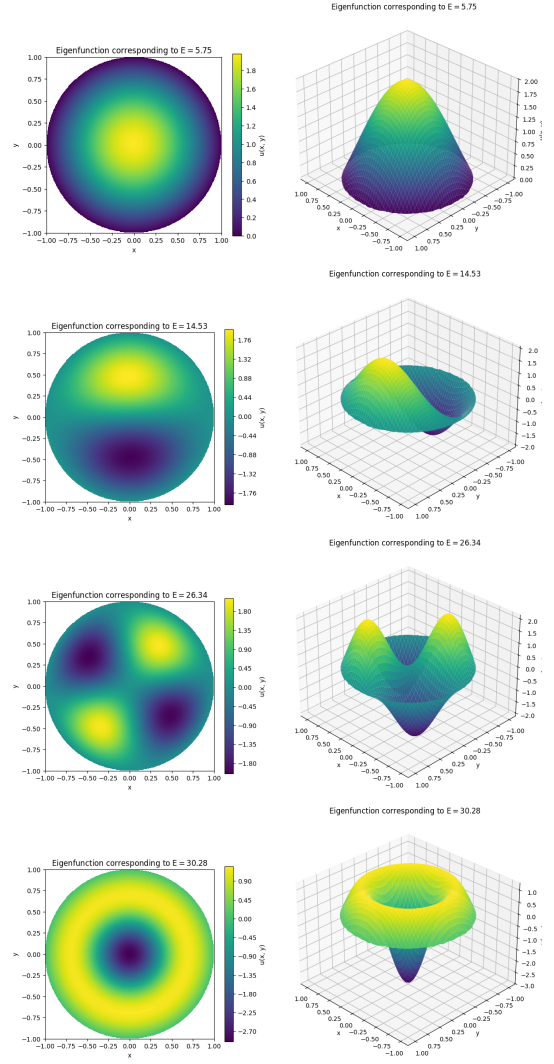


FIG. 4. The first four computed eigenfunctions of (5) in the 2-dimensional unit disk.

this range, problem (5) has one eigenvalue, denoted  $E_{(1)}$ , with value  $E_{(1)} = 49.348$ . See Fig. 5 for the computed loss curve in this interval, together with the theoretical upper bound.

#### Higher dimensions

The proposed framework extends naturally to higher spatial dimensions, with minimal changes required in the implementation. As an example, we consider the unit ball in  $\mathbb{R}^d$ , with  $d = 3, 4$  where the eigenvalue problem becomes:

$$\begin{cases} -\Delta u = Eu & \text{in } B^d \\ u = 0 & \text{on } \partial B^d. \end{cases}$$

Despite the increased computational cost, the method remains effective, as shown in Fig. 6.

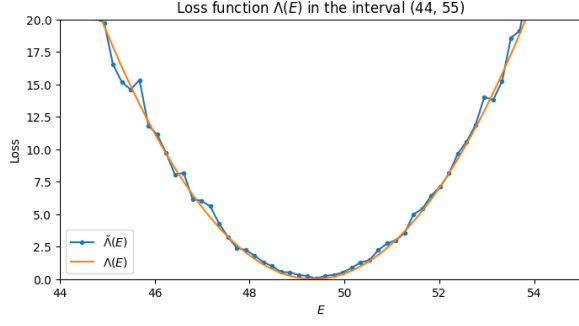


FIG. 5. The loss curve  $\tilde{\Lambda}(E)$  in two-dimensional space in the unit square computed in the interval  $[44, 55]$ .

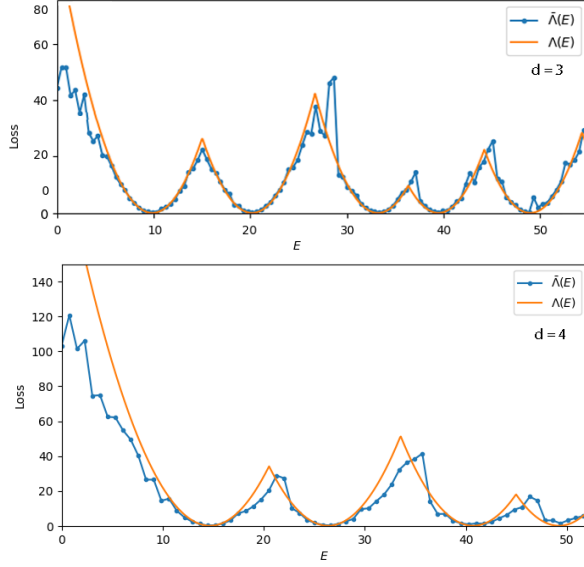


FIG. 6. The loss curve  $\tilde{\Lambda}(E)$  in three-dimensional space (above) and in the four-dimensional space (below) in the unit ball compared with the theoretical upper bound.

### Different geometries

We also explore domains with more complex geometries, such as squares, annuli, or triangles. The boundary factor  $B(x)$  can be adapted to these shapes, ensuring that the Dirichlet condition is enforced. In Fig. 7, we show eigenfunctions obtained in a square, in Fig. 8 the annular region is considered and in Fig. 9 the results in a triangle are shown.

### Different potentials

Beyond hard-wall confinement, our approach accommodates general potential functions  $V(x)$ , including smooth wells and step-like profiles. As an example, we consider a harmonic potential  $V(x) = \frac{\omega^2}{2}|x|^2$  for differ-

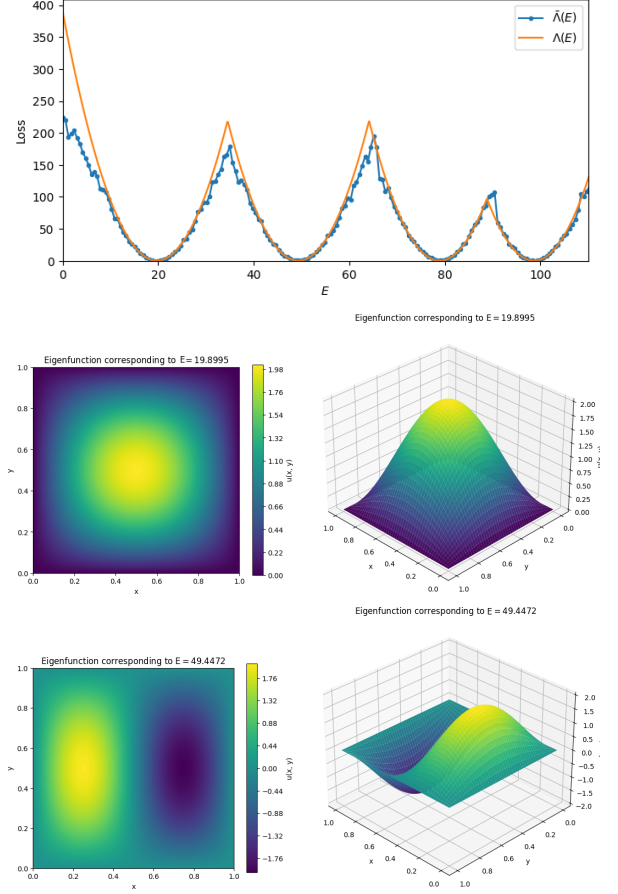


FIG. 7. The loss curve  $\tilde{\Lambda}(E)$  in two-dimensional space in the unit square compared with the theoretical upper bound. Below it is displayed the first two eigenfunction.

ent values of  $\omega$ , illustrating the method's applicability to physically relevant models. Fig. 10 displays loss curve and the first eigenfunction for  $\omega = 1$  and Fig. 11 for  $\omega = 10$ .

### NONLINEAR PROBLEMS

Our method can also be extended to nonlinear eigenvalue problems. As an illustrative example, we consider the eigenvalue problem for the  $p$ -Laplacian in the two-dimensional unit disk:

$$\begin{cases} -\nabla \cdot (|\nabla u|^{p-2} \nabla u) = E|u|^{p-2}u & \text{in } D, \\ u = 0 & \text{on } \partial D, \end{cases} \quad (7)$$

where  $D \subset \mathbb{R}^2$  is the unit disk and  $p > 1$ .

To approximate the solution using our framework, we modify the loss function accordingly. For a fixed  $E$ , we

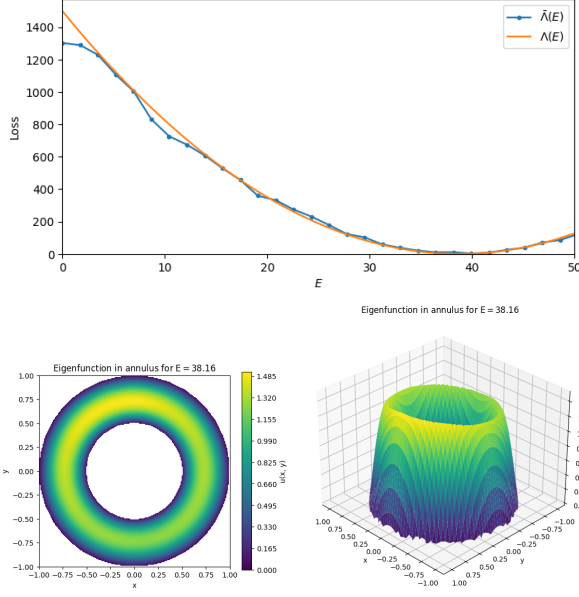


FIG. 8. The loss curve  $\bar{\Lambda}(E)$  in two-dimensional space in the unit annulus of radii 0.5 and 1 compared with the theoretical upper bound. Below it is displayed the first eigenfunction.

define the following loss:

$$\Lambda_p(u, E) = \int (\Delta_p u + E|u|^{p-2}u)^2 dx + \mu \left( \int |u|^p dx - 1 \right)^2,$$

where  $\Delta_p u = \nabla \cdot (|\nabla u|^{p-2} \nabla u)$ .

Here, the normalization condition is adapted to the natural scaling of the  $p$ -Laplacian:  $\|u\|_p = 1$ . As in the linear case, the output of the neural network is multiplied by a boundary factor to enforce homogeneous Dirichlet boundary conditions.

Since the nonlinear problem lacks an orthogonality structure among eigenfunctions, traditional sequential methods based on orthogonal projections are not directly applicable. Our approach, however, remains valid and does not rely on any such structure, which makes it suitable for problems like (7). As a result, we are still able to identify the first two eigenvalues as the values of  $E$  for which the loss function attains a sufficiently small minimum.

Figure 12 shows the loss curve and the corresponding eigenfunction computed for the case  $p = 2.2$ .

#### CODE AVAILABILITY

The Python code used in this work is openly accessible at the following URL:

<https://github.com/amsalort/PINN-eigenvalues>

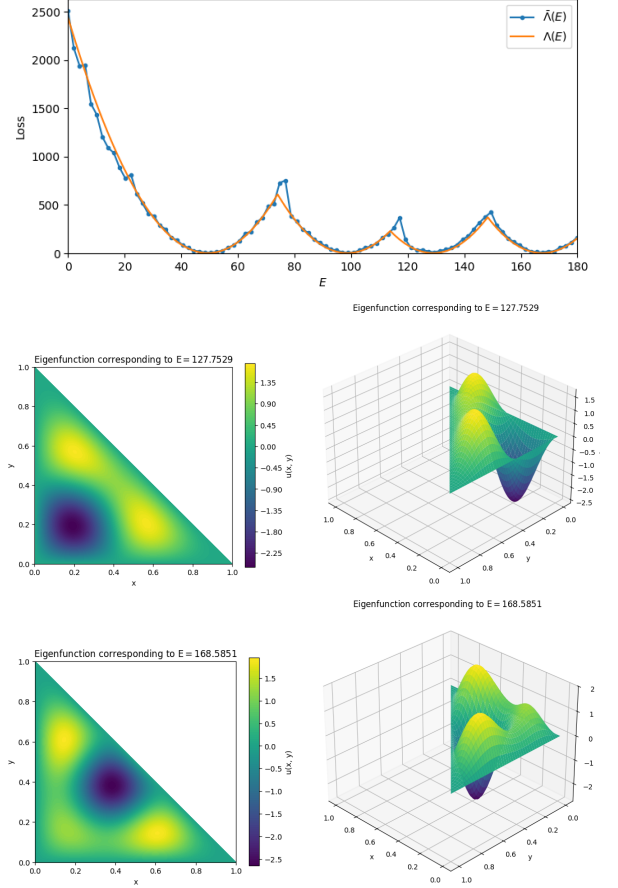


FIG. 9. The loss curve  $\bar{\Lambda}(E)$  in two-dimensional space in the triangle of vertices  $(0,0)$ ,  $(0,1)$ ,  $(1,0)$  compared with the theoretical upper bound and the third and fourth computed eigenfunctions.

#### CONCLUSIONS

The method proposed in this work provides a flexible and effective framework for computing eigenvalues and eigenfunctions of differential operators using neural networks. Unlike classical approaches, it does not rely on the Rayleigh quotient or require orthogonality conditions between eigenfunctions. As a result, it is particularly well-suited for computing higher eigenvalues and for addressing nonlinear problems where such structures are no longer available.

Moreover, the approach naturally extends to irregular domains and high-dimensional settings, where traditional mesh-based methods become increasingly difficult to apply. Its non-sequential nature allows for the identification of multiple eigenvalues without the need to compute the entire spectrum in order. These features make it a robust and broadly applicable alternative for spectral problems in both linear and nonlinear contexts.

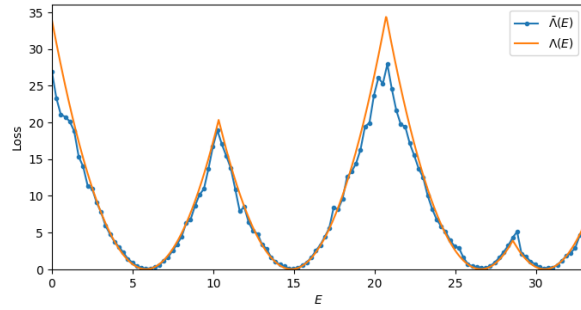
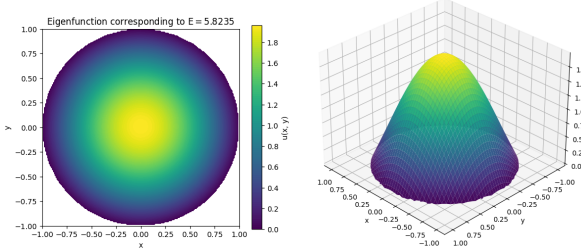
Eigenfunction corresponding to  $E = 5.8235$ 

FIG. 10. The loss curve  $\bar{\Lambda}(E)$  in two-dimensional disk with harmonic potential  $V = \frac{\omega^2}{2}|x|^2$  and  $\omega = 1$  with the first computed eigenfunction.

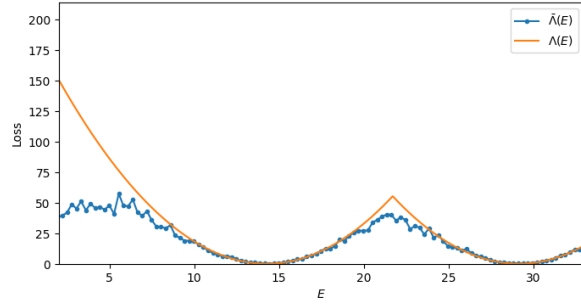
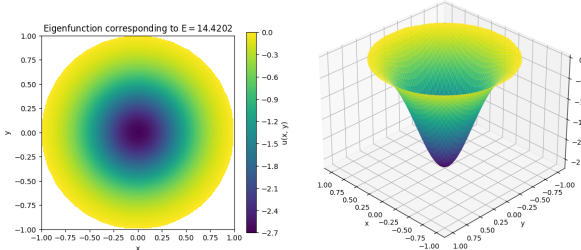
Eigenfunction corresponding to  $E = 14.4202$ 

FIG. 11. The loss curve  $\bar{\Lambda}(E)$  in two-dimensional disk with harmonic potential  $V = \frac{\omega^2}{2}|x|^2$  and  $\omega = 100$  with the first computed eigenfunction.

### ACKNOWLEDGMENTS

This work was partially supported by ANPCyT under grant PICT 2019-3837 and by CONICET under grant PIP 11220150100032CO.

JFB thanks Pablo Groisman for encouraging him to teach a course on the Mathematical Foundations of Deep

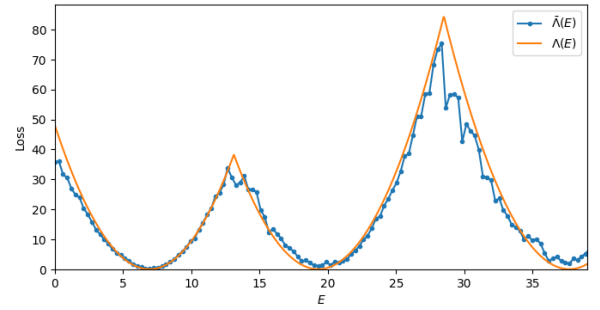
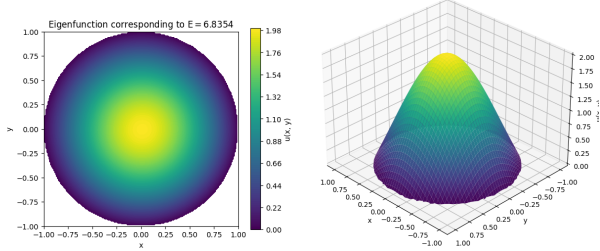
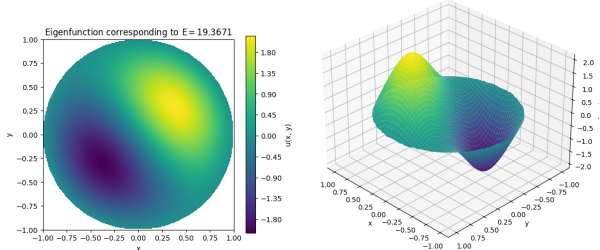
Eigenfunction corresponding to  $E = 6.8354$ Eigenfunction corresponding to  $E = 19.3671$ 

FIG. 12. The loss curve  $\bar{\Lambda}(E)$  in two-dimensional space for the p-Laplacian with  $p=2.2$  in the unit disk compared with the theoretical upper bound and the first two computed eigenfunctions.

Learning in the fall semester of 2025, which inspired the author to pursue this problem.

\* jfbonder@dm.uba.ar

† ariel.salort@ceu.es

- [1] I. E. Lagaris, A. Likas, and D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE Transactions on Neural Networks* **9**, 987 (1998).
- [2] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, arXiv preprint arXiv:1711.10561 (2017).
- [3] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics* **378**, 686 (2019).
- [4] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, Physics-informed machine learn-

- ing, *Nature Reviews Physics* **3**, 422 (2021).
- [5] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, DeepXDE: a deep learning library for solving differential equations, *SIAM Rev.* **63**, 208 (2021).
- [6] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman, *Universal differential equations for scientific machine learning* (2021), arXiv:2001.04385 [cs.LG].
- [7] C. Rackauckas, M. Innes, Y. Ma, J. Bettencourt, L. White, and V. Dixit, *Diffeqflux.jl - a julia library for neural differential equations* (2019), arXiv:1902.02376 [cs.LG].
- [8] J. Blechschmidt and O. G. Ernst, Three ways to solve partial differential equations with neural network—a review, *GAMM-Mitt.* **44**, Paper No. e202100006, 29 (2021).
- [9] J. Berg and K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, *Neurocomputing* **317**, 28–41 (2018).
- [10] H. Jin, M. Mattheakis, and P. Protopapas, Unsupervised neural networks for quantum eigenvalue problems, in *Proceedings of the 2020 NeurIPS Workshop on Machine Learning and the Physical Sciences* (Vancouver, Canada, 2020) NeurIPS Workshop.
- [11] H. Jin, M. Mattheakis, and P. Protopapas, Physics-informed neural networks for quantum eigenvalue problems, in *Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2022).
- [12] E. G. Holliday, J. F. Lindner, and W. L. Ditto, Solving two-dimensional quantum eigenvalue problems using physics-informed machine learning (2023), arXiv:2302.01413 [physics.comp-ph].
- [13] C. Rowan, J. Evans, K. Maute, and A. Doostan, Solving engineering eigenvalue problems with neural networks using the rayleigh quotient (2025), arXiv:2506.04375 [math.NA].

### The minimization problem

In this appendix, we show that the minimization problem leading to the loss curve  $\Lambda(E)$  defined in (3) admits a solution. In order to keep things simple, we consider the case in which  $V(x)$  is a confining potential of the form (4) with  $\Omega \subset \mathbb{R}^n$  bounded.

To this end, we must specify the function space over which the infimum is taken. The natural choice is the Sobolev space  $H^2(\Omega) \cap H_0^1(\Omega)$ . The main result of this section reads as follows:

**Theorem 1.** *Let  $V(x)$  be as in (4) with  $\Omega \subset \mathbb{R}^n$  bounded. Given  $E \geq 0$ , there exists  $u_E \in H^2(\Omega) \cap H_0^1(\Omega)$  such that*

$$\Lambda(E) = \Lambda(u_E, E) = \inf_{u \in H^2(\Omega) \cap H_0^1(\Omega)} \Lambda(u, E),$$

where  $\Lambda(u, E)$  is given in (2).

*Proof.* The result follows from the *direct method in the Calculus of Variations*. Fix  $E \geq 0$ , and consider a minimizing sequence  $\{u_j\}_{j \in \mathbb{N}} \subset H^2(\Omega) \cap H_0^1(\Omega)$  such that  $\|u_j\|_2 = 1$  and

$$\Lambda(E) = \lim_{j \rightarrow \infty} \Lambda(u_j, E).$$

In particular, the sequence  $\Lambda(u_j, E)$  is bounded and non-negative:

$$0 \leq \Lambda(u_j, E) \leq C, \quad (8)$$

for some constant  $C > 0$ .

Now observe that

$$\begin{aligned} \Lambda(u_j, E) &= \int_{\Omega} (\Delta u_j + E u_j)^2 dx \\ &= \int_{\Omega} (\Delta u_j)^2 dx + 2E \int_{\Omega} u_j \Delta u_j dx + E^2. \end{aligned}$$

Using the inequality  $ab \leq \epsilon a^2 + \frac{b^2}{4\epsilon}$  for any  $a, b > 0$  and  $\epsilon > 0$  we estimate:

$$\int_{\Omega} |u_j| |\Delta u_j| dx \leq \epsilon \int_{\Omega} (\Delta u_j)^2 dx + \frac{1}{4\epsilon} \int_{\Omega} u_j^2 dx.$$

Taking  $\epsilon = 1/4E$ , we obtain

$$\Lambda(u_j, E) \geq \frac{1}{2} \int_{\Omega} (\Delta u_j)^2 dx + E^2 - E. \quad (9)$$

Combining (8), (9) and the normalization  $\|u_j\|_2 = 1$  we conclude that the sequence  $\{u_j\}_{j \in \mathbb{N}}$  is bounded in  $H^2(\Omega) \cap H_0^1(\Omega)$ .

Therefore, up to a subsequence, we have  $u_j \rightharpoonup u_E$  weakly in  $H^2(\Omega) \cap H_0^1(\Omega)$  for some  $u_E$  in that space.

Since the functional

$$u \mapsto \int_{\Omega} (\Delta u + E u)^2 dx$$

is convex in  $u$ , it is weakly lower semicontinuous. Thus,

$$\int_{\Omega} (\Delta u_E + E u_E)^2 dx \leq \liminf_{j \rightarrow \infty} \int_{\Omega} (\Delta u_j + E u_j)^2 dx.$$

Moreover, the immersion  $H^2(\Omega) \cap H_0^1(\Omega) \hookrightarrow L^2(\Omega)$  is compact, so

$$\int_{\Omega} u_E^2 dx = \lim_{j \rightarrow \infty} \int_{\Omega} u_j^2 dx = 1.$$

Hence,  $u_E$  is admissible and attains the minimum.  $\square$