IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 October 2025, accepted 17 November 2025, date of publication 24 November 2025,
date of current version 2 December 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3636289

RESEARCH ARTICLE

A Physics-Informed Graph Neural Network for Computing Laplace–Beltrami Eigenfunctions on Manifolds

DAMIANA LAZZARO, SERENA MORIGI, AND PAOLO ZUZOLO
Department of Mathematics, University of Bologna, 40126 Bologna, Italy

Corresponding author: Serena Morigi (serena.morigi@unibo.it)

This work was supported by the Made in Italy–Circular and Sustainable (MICS) Extended Partnership and European Union Next-GenerationEU (PNRR) under Grant 1551.11-10-2022, PE00000004, CUP J33C2200295000. The work of Damiana Lazzaro and Serena Morigi was supported in part by PRIN2022_MORIGI, titled "Inverse Problems in the Imaging Sciences (IPIS)" 2022 ANC8HL, under Grant CUP J53D23003670006; by the National Group for Scientific Computation (GNCS-INDAM) through the Research Projects 2025, and in part by PRIN2022_PNRR_CRESCENTINI under Grant CUP J53D23014080001. The work of Paolo Zuzolo was supported by PNRR CN-HPC through European Commission, NextGeneration EU Program, under Grant CUP J33C22001170001.

ABSTRACT Eigenfunctions and the spectrum of the Laplace-Beltrami operator are fundamental tools in geometry processing, providing a robust framework for analyzing complex geometries. This paper introduces an incremental framework specifically designed for the computation of the leading eigenpairs of the Laplace-Beltrami operator of 2-manifolds represented by triangular surface meshes in $\mathbb{R}^3$. The proposed approach relies on a discrete physics-informed neural network (dPINN), which leverages Graph Neural Networks to effectively model the intrinsic geometry and topology of the manifold. dPINNs are ideally suited for learning on a triangular surface mesh in $\mathbb{R}^3$ because they leverage its discrete representation and numerical differentiation. Conversely, continuous PINNs (cPINNs) prove inadequate when the spatial domain (manifold) is not readily available for arbitrary sampling. Numerical experiments on several 2-manifolds in $\mathbb{R}^2/\mathbb{R}^3$ demonstrate the accuracy, robustness to geometric and topological variations, and generalization capabilities of the proposed neural network approach in case of 2-manifolds with parametric representations. The presented method aims to demonstrate how learning-based approaches can improve upon traditional numerical methods under adverse meshing conditions or domain discretization changes.

INDEX TERMS Laplace–Beltrami eigendecomposition, geometric Laplacian, physics-informed discrete neural network, graph neural networks.

I. INTRODUCTION

Many geometric objects of interest are naturally embedded in Euclidean space. They are Riemannian 2-manifolds embedded in $\mathbb{R}^3$ endowed with a Riemannian metric that is induced from the standard Euclidean metric of $\mathbb{R}^3$. Triangular meshes serve as discrete approximations of smooth 2-manifolds embedded in $\mathbb{R}^3$ by using piecewise planar surfaces. The finer the mesh (i.e., the more polygons used), the closer the approximation can be to the original smooth surface.

The Laplace-Beltrami operator (LBO) plays a central role in comprehending the geometric and topological characteristics of Riemannian manifolds. This operator, intrinsically linked to a Riemannian metric, encodes crucial geometric information about the manifold, including its spectral properties. Investigating these spectral properties and their geometric implications has been a focal point of substantial research endeavors in recent times [1], [2], [3], [4].

Similarly, for discrete surfaces represented by triangular meshes, the discrete LBO, also known as geometric or graph Laplacian, represents a fundamental geometric operator. It has found widespread use in spectral analysis

The associate editor coordinating the review of this manuscript and approving it for publication was Emanuele Crisostomi.

and various geometric processing tasks, including shape descriptors [1], [5], shape matching [2], partitioning [3], 3D feature extraction [6]. More recently spectral learning LBO methods have been introduced for 3D shape analysis. For example, in [7] a manifold learning scheme is presented using deep neural networks, while in [8] isospectralization is performed by matching a limited portion of a shape's LBO spectrum to another. Other approaches, such as Riemannian Metric Optimization on Surfaces (RMOS) in [9], propose an intrinsic surface mapping within the LBO embedding space, while in [10] a supervised LBONet is proposed to modify the LBO for specific geometric tasks. However, all these interesting approaches require the calculation, even if partial, of the eigenpairs of the LBO.

The core challenge thus becomes the efficient approximation of the LBO eigenvalues and eigenfunctions on an arbitrary topology manifold, by computing eigenpairs of the geometric Laplacian on its representative triangular surface mesh in $\mathbb{R}^3$.

This is exactly the aim that this paper's contribution fulfills.

Spectral decompositions on a discretized manifold can be approached in several ways, depending on how the problem is mathematically formulated, on how the continuous LBO is discretized, and on how the subsequent eigendecomposition is numerically computed. Finite Element Methods (FEM), spectral methods and Finite Difference Methods (FDM) can be applied to compute the LBO spectrum of a discretized manifold. The choice of the method depends heavily on the geometry of the manifold. For example, FEM is well-suited for complex geometries, however can be computationally expensive, especially when high accuracy is required. Moreover, FEM accuracy and convergence are strongly dependent on the quality and resolution of the mesh. FDM, on the other side, is easier to implement and highly efficient on regular grids, but it often struggles with irregular domains and complex boundary conditions, which can limit its accuracy and stability when the underlying geometry is nontrivial.

This work proposes a method for solving the Laplace-Beltrami Eigendecomposition Problem on 2-manifolds, which are represented by triangular surface meshes in $\mathbb{R}^3$. Specifically, we compute the leading eigenfunctions and corresponding eigenvalues, by utilizing an unsupervised Physics-Informed Neural Network (PINN) framework based on Graph Neural Networks (GNNs). PINNs, introduced by Raissi et al. [11], has emerged as an alternative for solving a wider class of partial differential equations (PDEs). However, to our knowledge, no PINN approaches have yet been presented to solve the specific proposed problem.

Solving the LBO eigendecomposition problem on discrete surfaces poses challenges for continuous PINNs. The inherent Euclidean nature of their solution space and the use of automatic differentiation are inadequate for handling complex geometries. In addition, the cPINN training process involves evaluating a large number of points arbitrarily located in the manifold. This poses a natural limit in the case the shape domain is described uniquely by a given triangular mesh of arbitrary topology.

In light of these limitations, our approach focuses on a discrete PINN framework tailored for Laplacian Eigendecomposition Problems (LEP) on discrete manifolds, named LEP-dPINN in the following. This framework employs numerical differentiation and confines evaluations to the mesh vertices.

The core of our dPINN proposal is the Generalized GNN, which is effective due to two key aspects. First, GNNs capture in a natural way the domain's topological structure, including mesh connectivity. Second, the convolutional operators in the GNN are derived by directly solving a Dirichlet energy driven variational formulation. According to the proposed Generalized GNNs in [12], physics-inspired convolution operators can be incorporated within a GNN framework through a learnable, energy-driven evolution process, addressing features such as smoothness, sparsity, or enhancement.

The contribution of this paper is twofold. We first formulate the LBO eigendecomposition problem in term of a constrained minimization problem involving the Rayleigh quotient. To solve this problem a numerical optimization technique is provided by leveraging an Projected Gradient Descent Method on Manifolds (M-PGD). This iterative process allows us to sequentially compute the leading LBO eigenpairs. Building on this incremental approach, we then introduce LEP-dPINN, a learning-based approach that leverages a graph-based neural network to compute successive eigenpairs, exploiting orthogonality with previously computed eigenfunctions.

Both methods compute the first K leading eigenpairs. In the numerical tests we considered $K = 10\%$ of the mesh's vertex number. While this might appear to be a limitation, focusing on the first 10% of the principal eigenfunctions of the LBO is generally sufficient for many practical applications. These leading eigenfunctions effectively capture the global and most significant structure of the surface under study, neglecting highly localized details or multiple levels of refinement. This is ideal for tasks such as coarse surface representation (e.g., for compression), low-resolution shape recognition, basic geometric modeling, and signal processing on surfaces, [13], [14].

A key advantage of LEP-dPINN, as shown in the numerical section, lies in its ability to solve the LEP on manifolds remaining agnostic to their discretization, while preserving good accuracy and providing runtime benefits. In contrast, both standard FEMs and numerical methods require recomputation of stiffness and mass matrices for each new discretization and are sensitive to mesh quality. LEP-dPINN offers a significant computational advantage: once trained, eigenpairs on unseen meshes can be easily obtained through a single forward pass of the network.

This enables to achieve independence on the domain discretization.

The paper is organized as follows. Section II highlights the benefits of dPINN over cPINN for LBO eigendecomposition on discrete manifolds. Section III introduces the continuous LBO and its discrete counterpart on triangular surface meshes. The incremental optimization solution is presented in section IV, and the dPINN-based framework is described in section V. Numerical experiments will be illustrated in section VI and section VII draws conclusions and discusses limitations.

II. CONTINUOUS VERSUS DISCRETE PINNs

Utilizing neural networks' capacity as universal function approximators, PINNs approach the PDE solution by a neural network, optimizing the parameters to minimize the strong form of the PDE residual via stochastic gradient descent. Beyond the strong PDE residual form, other physics-informed models focus on different forms of PDE residuals, such as the variational form (VPINNs) [15], and weak form (wPINNs) [16]. For a detailed introduction to PINN approaches, we refer the reader to De Ryck and Mishra [17].

PINNs present themselves as part of a paradigm for solving PDEs based on three key components: input spatio-temporal features, network architecture choice, and numerical derivatives computation, as depicted in the dashed boxes in Fig. 1. The first component depends on the given domain representation: either coordinates sampled from a continuous distribution, or given points often enriched with additional information such as vertices connectivity, whenever the domain is discrete. This process directly influences the neural network architecture choice, as it must be capable of capturing the inherent structure and complexities of the input features, adhering to their geometric and topological structure. The neural network then generates a function to be used as candidate solution to the given PDE: numerical derivatives are computed and integrated into the physics-informed loss function representing the PDE model. Generally speaking, the PINNs framework is then unsupervised since no groundtruth solution is available and the physics-informed loss function presents an unsupervised setting.

Based on these three key components PINNs can be further classified into two main frameworks: discrete PINNs and continuous PINNs.

The cPINNs is a mesh-free framework which works on spatial Euclidean domains, and thus a continuous domain representation is available for sampling from an adaptive or uniform distribution. They usually employ fully-connected neural networks since the input features do not exhibit any spatial relationships and the solution is then point-wise approximated, allowing for easy evaluation on new domain points. Derivatives are computed using Automatic Differentiation (AD), allowing for machine-precision computation of derivatives [11], [18], [19]. However, their point-wise formulation involves high training costs due to the extensive AD computations required across numerous points in high-dimensional spatio-temporal spaces [20].

In the dPINN framework, the domain is usually a discrete 2-manifold, represented by a given polygonal mesh, leading to employ convolutional neural networks or graph neural network to learn the spatio-temporal solution in the given collocation points. Consequently, derivatives are usually approximated using numerical differentiation (ND) schemes like finite differences [21], finite elements [15], [18] or Galerkin methods [16] and are evaluated on the mesh vertices.

The dependence of dPINNs on a predefined grid or mesh offers the advantage of a strong connection to traditional numerical methods, benefiting from their rich theoretical foundation, through analysis, and inherent robustness. Conversely, approximations of differential operators are grid-dependent and may introduce discretization errors, limiting flexibility, especially for complex geometries. Finer discretizations can significantly increase the approximation accuracy, but at the cost of higher computational demands. Finally, examples of coupled AD and ND approaches, are presented in [18] and [22].

III. LBO ON 2-MANIFOLDS AND ASSOCIATED GEOMETRIC LAPLACIAN ON 2D SURFACE MESHES

The LBO of a compact Riemannian manifold $\mathcal{M}$ is a second-order elliptic differential operator defined by the metric tensor of $\mathcal{M}$. Let f be a C^2 real-valued function defined on $\mathcal{M}$, then the *Laplace-Beltrami operator* $\Delta_{\mathcal{M}}$ applied to $f(x)$, for $x \in \mathcal{M}$, is defined as

$$\Delta_{\mathcal{M}} f(x) := div_{\mathcal{M}}(grad_{\mathcal{M}} f(x)), \tag{1}$$

where $grad_{\mathcal{M}}$ and $div_{\mathcal{M}}$ are the gradient and divergence on the manifold $\mathcal{M}$ with respect to its metric. Since these operators depend only on the metric of $\mathcal{M}$, the LBO is invariant to isometries of the manifold [23].

The Laplace-Beltrami eigenvalue equation on a compact Riemannian manifold $\mathcal{M}$ without boundary reads as

$$-\Delta_{\mathcal{M}}\phi = \lambda\phi, \tag{2}$$

where $\lambda \in \mathbb{R}$ is an eigenvalue of $\Delta_{\mathcal{M}}$ and ϕ is the eigenfunction associated with λ. The eigenpairs of the Laplacian on a manifold $\mathcal{M}$ are all the pairs (λ_k, ϕ_k) that satisfy (2). Since the LBO is self-adjoint ($< \Delta_{\mathcal{M}} f, g >=< f, \Delta_{\mathcal{M}} g >$), its matrix representation, in finite-dimensional space, is a Hermitian matrix. Then its eigenvalues are real, and the eigenfunctions corresponding to different eigenvalues are orthogonal. The LBO $-\Delta_{\mathcal{M}}$ is positive semidefinite, therefore, all its eigenvalues $\lambda_k, k \geq 1$, are real and non-negative. For $\lambda_1 = 0$ its corresponding eigenfunction ϕ_1 is a constant function on $\mathcal{M}$, and $\phi_1 = 1/\sqrt{Vol(\mathcal{M})}$. By ordering the eigenvalues according to their increasing magnitude, $0 \leq \lambda_1 < \lambda_2 \leq \lambda_3 \leq \ldots$, the spectrum naturally defines a scale space. Generally speaking, as the value of the eigenvalue increases, its corresponding eigenfunction varies more rapidly over the manifold.

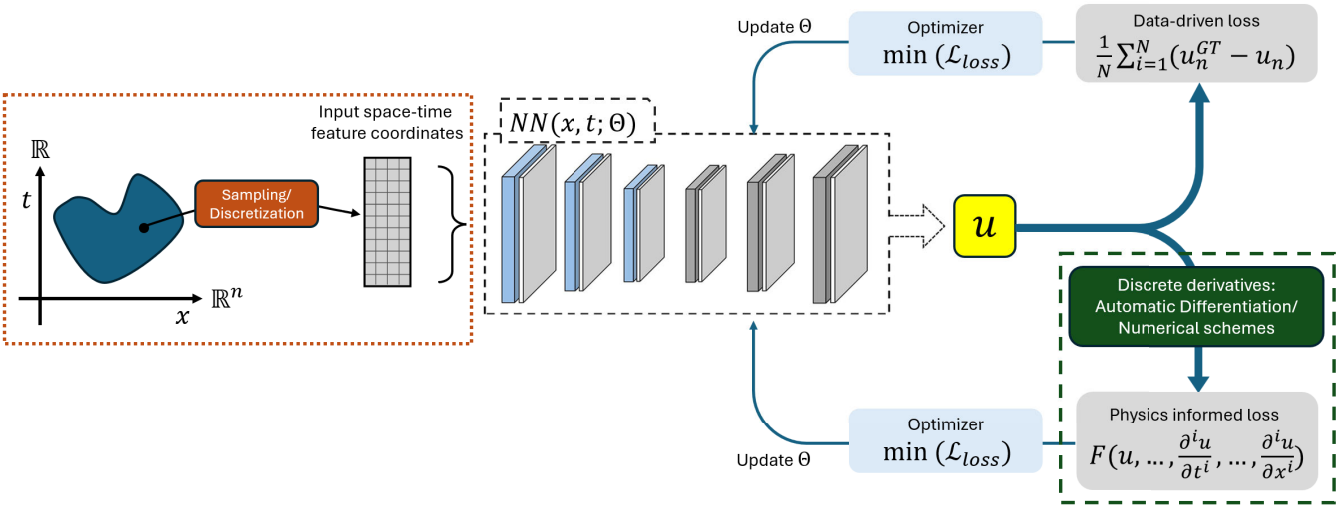


FIGURE 1. Schematic of a PINN architecture. The dashed boxes highlight the three key components of a PINN: input spatio-temporal domain and the sampling to extract input features, the network approximating the candidate solution, and the discrete derivatives used within the physics-informed loss.

For a manifold $\mathcal{M}$ with boundary $\partial\mathcal{M}$, equation (2) is usually paired with homogeneous Dirichlet or Neumann boundary conditions. In case of Dirichlet homogeneous boundary conditions $-\Delta_{\mathcal{M}}$ has its first eigenvalue $\lambda_1 > 0$, and thus is positive-definite.

The set of LBO eigenfunctions forms a complete orthonormal basis - called the manifold harmonic basis - for the Hilbert space of square-integrable functions on the manifold, denoted as $L^2(\mathcal{M})$. In this context, completeness means that any function in $L^2(\mathcal{M})$ can be arbitrarily well approximated by a finite linear combination of these eigenfunctions. In most practical applications, it suffices to consider only a set of $K \in \mathbb{N}$ eigenfunctions corresponding to the smallest eigenvalues.

The geometry processing of 3D objects in general involves the description of their shape as Riemannian 2-manifolds $\mathcal{M}$. These manifolds represent the boundary surfaces of the object, which can be discretized and represented as surface meshes in $\mathbb{R}^3$, here denoted by M. A triangular mesh $M = (X, E, T)$ consists of a set of vertices $X = \{p_i\}_{i=1}^{n_V} \in \mathbb{R}^{n_V \times 3}$, a set of edges $E = \{e_{i,j}\} \in \mathbb{N}^{n_E \times 2}$ and a set of planar triangular faces $T = \{\tau_i\}_{i=1}^{n_T} \in \mathbb{N}^{n_T \times 3}$. The set of vertices p_j connected to a vertex p_i via an edge is defined as the first ring: $N(i) = \{j \in \mathbb{N} : p_j \in X\ ,\ (i,j) \in E\}$.

Discretizations of the Laplace and Laplace-Beltrami operators on 2-manifolds $\mathcal{M}$ represented by polygonal meshes M should encode geometric and topological information and preserve desired properties.

Numerically, the LBO discretization leads to the construction of a symmetric Laplacian matrix $L \in \mathbb{R}^{n_V \times n_V}$. If f is the vector obtained by sampling the function f over the mesh M, then $\Delta f\ (p) \approx Lf|_p$ and letting p_i be one vertex of M, then

$$Lf\ (p_i) := \frac{1}{b_i} \sum_{j \in N(i)} w_{ij} \left[f\ (p_i) - f\ \left(p_j\right) \right], \tag{3}$$

where the scalar value $b_i > 0$ is associated to a vertex $p_i \in X$, and the scalar value $w_{I,j} > 0$ is the weight associated to the edge $e_{i,j}$, with $w_{I,j} = w_{j,i}$ for any edge $e_{i,j} \in E$. The Laplacian matrix L of a surface mesh in $\mathbb{R}^3$ is defined by elements $L_{i,j}$:

$$L_{i,j} = \begin{cases} w_{i,j} > 0 & e_{i,j} \in E \\ -\sum_{j \in N(i)} w_{i,j} & i = j \\ 0 & otherwise. \end{cases} \tag{4}$$

The literature defines various discrete Laplacians, each characterized by a specific definition of its weights. Simply setting $w_{i,j} = 1$ and $b_i = 1$ in (3) we get the standard *graph Laplacian*, [24]. The normalized version of the graph Laplacian $\bar{L}$ is obtained by normalization of the weights $\bar{w}_{i,j} = \frac{w_{i,j}}{\sum_{j \in N(i)} w_{ij}}$, which reads as

$$\bar{L} = I_{n_V} - W, \tag{5}$$

with W the adjacency matrix, and I_{n_V} the identity matrix.

Another popular choice regards the class of *geometric Laplacians* and, among them, the popular *cotangent Laplacian*, obtained by setting the weights in (3) as:

$$w_{i,j} = \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2}, \quad b_i = Area(i) \approx \sum_{j \in N(i)} |\tau_j|, \tag{6}$$

where α_{ij} and β_{ij} are the opposite angles to the shared edge $e_{i,j}$ and $|\tau_j|$ is the area of a triangle τ_j having vertex p_i among its vertices. The cotangent Laplacian incorporates the discrete manifold's geometry by considering areas and angles. For a description of several other geometric alternatives see [25]. The cotangent Laplacian characterizes the matrix L, defined as in (4), with weights in (6) and is expressed in matrix form as:

$$\bar{\bar{L}} = B^{-1} L, \tag{7}$$

where B is a diagonal matrix $B = diag(b_1, b_2, \ldots, b_{n_V})$.

The Laplacian eigenvalue problem in (2) is then formulated in a discrete setting as a generalized eigenvalue problem

$$L\phi = \lambda B\phi, \tag{8}$$

which admits non-negative eigenvalues $\lambda \in \mathbb{R}$ and a set of n_V orthonormal eigenvectors ϕ, where n_V is the number of vertices. In case of the graph Laplacian defined in (5), problem (8) reduces to $L\phi = \lambda\phi$. The spectra of the geometric Laplacian converge to the LBO spectra on a closed manifold as the max edge length of the mesh vanishes [26]. Since matrix L is symmetric, and matrix B is symmetric positive-definite, then the generalized eigenvectors corresponding to different generalized eigenvalues λ are orthogonal. However, the orthogonality is with respect to the scalar product induced by the matrix B

$$< \phi_i, \phi_j >_B = \phi_j^T B \phi_i. \tag{9}$$

The FEM-based solution to the eigenvalue problem (2) for manifolds $\mathcal{M}$ without boundary is obtained by reformulating it in integral form (weak form), as

$$< \Delta_{\mathcal{M}}\phi, \psi_i >_{L^2(\mathcal{M})} = -\lambda < \phi, \psi_i >_{L^2(\mathcal{M})} . \tag{10}$$

Considering piecewise-linear finite elements for both *test* and *basis functions*, with $\psi_i (i = 1, \ldots, n_V)$ such that $\psi_i(i) = 1$ and $\psi_i(j) = 0$ if $i \neq j$, problem (10) leads to the *generalized symmetric eigenvalue problem* of the form (8), where the stiffness matrix L is defined as in (4) with weights $w_{i,j}$ in (6) and, unlike the geometric Laplacian, the mass matrix B is defined by the elements

$$B_{i,j} := \begin{cases} \dfrac{|\tau_i| + |\tau_j|}{12} & (i,j) \quad \text{edge} \\ \dfrac{-\sum_{k \in N(i)} |\tau_k|}{6} & i = j \\ 0 & otherwise. \end{cases} \tag{11}$$

Then the FEM-based Laplacian, with B in (11), is defined as

$$\widehat{L} = B^{-1}L. \tag{12}$$

FEM-based Laplacians are in general more accurate than the geometric Laplacians. However, in the simplest case of piecewise linear finite element spaces and lumped diagonal mass B, FEM-based Laplacian reduces to the geometric Laplacian.

Standard numerical linear algebra techniques are then used to solve the (generalized) matrix eigenvalue problem (8). The package ARPACK (ARnoldi PACKage) relying on the Implicitly Restarted Arnoldi Method or, in the case of symmetric matrices, the corresponding variant of the Lanczos algorithm can compute a few eigenvalues and corresponding eigenvectors of large sparse or structured matrices. ARPACK is used by many popular numerical computing environments such as SciPy [27], Mathematica, GNU Octave and MATLAB to provide this functionality.

The generalized Rayleigh quotient $R : \mathbb{R}^{n_V} \setminus \{0\} \to \mathbb{R}$ characterizes the eigenpairs and is defined as

$$R(\phi) := \frac{\phi^T L\phi}{\phi^T B\phi}. \tag{13}$$

Notably, each eigenvector ϕ_k corresponds to a critical point of the generalized Rayleigh quotient $R(\phi)$, with the eigenvalue λ_k being the value $R(\phi_k)$.

The eigenvectors of the Laplacian depend on the structure of the matrix L. Since the graph Laplacian $\bar{L}$ is constructed based purely on the topology of the mesh (the adjacency of vertices), then the eigenvectors of a scaled mesh's Laplacian will be the same as those of the original mesh (possibly with sign flips or reordering due to the non-uniqueness of eigenvectors). However, since the geometric Laplacian incorporates geometric information (like edge lengths, angles, or face areas) then scaling the mesh will change the Laplacian matrix $\bar{\bar{L}}$, and therefore, the eigenvectors of the scaled mesh's Laplacian will generally be different from those of the original mesh.

Proposition 1: Let $\mathcal{M}$ and $\mathcal{M}_s$ be a given compact 2-manifold and its scaled version $\mathcal{M}_s = s\mathcal{M}$, with scaling factor $s > 0$. If (λ, ϕ) is an eigenpair of $\mathcal{M}$ then $(\frac{1}{s^2}\lambda, \frac{1}{s}\phi)$ is an eigenpair of $\mathcal{M}_s$.

Proof: First we recall that if one 2-manifold is a scaled version of another by a factor of s, the area of the scaled manifold is s^2 times the area of the original manifold. Let $x : \Omega \subset \mathbb{R}^2 \to \mathcal{M} \subset \mathbb{R}^3$ be a local smooth parameterization of $\mathcal{M}$, and g its induced metric from x (Ω). Then $g_s = s^2 g$ is the induced metric of $\mathcal{M}_s$. Using the formula for the surface area of a parameterized manifold $\mathcal{M}$, which reads as $Area = \int_\Omega \sqrt{\det g}\, d\Omega$, we have

$$\begin{aligned} Area_s &= \int_\Omega \sqrt{\det g_s}\, d\Omega = \int_\Omega \sqrt{\det(s^2 g)}\, d\Omega \\ &= \int_\Omega \sqrt{s^4 \det g}\, d\Omega = s^2\, Area \end{aligned} \tag{14}$$

Now let us consider the generalized eigenvalue problem (8) on $\mathcal{M}$, with geometric Laplacian (7) and area matrix B in (6), then on $\mathcal{M}_s$ we have

$$L\phi_s = \lambda_s B_s \phi_s = \lambda_s (s^2 B)\phi_s = (s^2\lambda_s) B\phi_s. \tag{15}$$

It follows that $\lambda_s = \frac{\lambda}{s^2}$. Moreover, by orthonormality of ϕ and ϕ_s, we have $1 =< \phi_s, \phi_s >_{s^2 B} = s^2 (\phi_s)^T B\phi_s$ and $1 =< \phi, \phi >_B = \phi^T B\phi$, then $\phi_s = \frac{\phi}{s}$. □

IV. INCREMENTAL OPTIMIZATION SOLUTION OF THE GENERALIZED EIGENVALUE PROBLEM

The problem of simultaneously estimating multiple eigenvectors $\phi_1, \ldots, \phi_k$ of the geometric Laplacian can be formulated as an optimization problem under linear orthogonality constraints, defining a matrix $\Phi^{(k)} \in \mathbb{R}^{n_V \times k}$ whose columns are estimates of the first k eigenfunctions ϕ_l, $l = 1, \ldots, k$, associated with the k smallest eigenvalues of the geometric

Laplacian operator L. In formula,

$$\Phi^{(k)} = \underset{\Phi\in\mathbb{R}^{n_V\times k}}{\arg\min}\ R(\Phi) \quad \text{subject to} \quad <\Phi,\Phi>_B = I_k, \tag{16}$$

where I_k is the identity matrix in $\mathbb{R}^{k\times k}$ and $R(\Phi)$ is the Rayleigh quotient defined in (13). The optimization problem in (16) is too computationally demanding to be solved efficiently. To address this issue, we introduce an incremental scheme which is based on a known result in spectral geometry. The result states that the first non-zero eigenvalue, λ_2, is the minimum value of the Rayleigh quotient over all functions orthogonal to the constant function (the eigenfunction for $\lambda_1 = 0$). Subsequent eigenvalues, $\lambda_3, \lambda_4, \ldots$, are found by minimizing the Rayleigh quotient over function subspaces that are orthogonal to the preceding eigenfunctions [28]. Therefore, the numerical difficulty of solving (16) can be partially reduced by an incremental scheme which computes a new eigenfunction ϕ_{k+1} - i.e., add a new, $(k+1)$-th column to matrix $\Phi^{(k)}$ - orthogonal to the set of already computed orthogonal eigenfunctions $\phi_1, \ldots, \phi_k$ - i.e., the given matrix $\Phi^{(k)}$. Starting from the first, known constant eigenfunction, problem (16) is thus transformed into the following sequence of simpler problems:

$$\phi_{k+1} \in \underset{\phi\in\mathbb{R}^{n_V}}{\arg\min}\ R(\phi) \quad \text{s.t. } \phi\in\mathcal{O}_2^{(k)}, \quad k=1,2,\ldots \tag{17}$$

$$\text{with} \quad \mathcal{O}_2^{(k)} := \left\{\phi : \langle\phi,\phi_l\rangle_B = 0, \quad \forall l = 1,\ldots,k\right\}, \tag{18}$$

where the manifold constraint represents the k-th orthogonality constraint manifold. For any $k \in \{1,2,...\}$, the optimization problem (17) admits nontrivial solution ($R(\phi)$ is not defined in $\phi = 0_{n_V}$).

We compute approximate solutions to any k-th optimization problem in (17) by means of the Projected Gradient Descent on manifold (M-PGD) iterative algorithm with constraint manifold $\mathcal{O}_2^{(k)}$ defined in (18). Generalizing the projected gradient descent method from Euclidean spaces to Riemannian manifolds requires us to use the Riemannian gradient as the search direction and the projection to move between points on the manifold.

The M-PGD algorithm iterates over $\ell = 0,1,\ldots$ which we refer as *inner* iterations of the overall incremental approach proposed in (17)-(18) to distinguish them from the *outer* ones, with index $k = 1,2,\ldots$, over the different eigenfunctions to compute. For any $k = 1,2,\ldots$, we compute the $(k+1)$-th eigenfunction ϕ_{k+1} by applying the M-PGD algorithm to the solution of problem (17)-(18). The function $R(\phi)$ in (13) is continuously differentiable on $\mathbb{R}^{n_V}$ and bounded below by zero, and its gradient is given by

$$\nabla R(\phi) = \frac{2}{\phi^T B\phi}(L\phi - R(\phi)B\phi). \tag{19}$$

At each ℓ-th M-PGD iteration, we first compute the gradient $g^{(\ell)} \in \mathbb{R}^{n_V}$ of the cost function R by means of the formula in (19). Then, we compute the orthogonal projection of vector $g^{(\ell)}$ onto the linear subspace $\mathcal{O}_2^{(k)}$ denoted by $d^{(\ell)} = \text{proj}_{\mathcal{O}_2^{(k)}}(g^{(\ell)})$. The columns of matrix $\Phi^{(k)}$ are orthogonal by construction. However, their Euclidean norm is not unitary, then let consider a B-normalized form as follows:

$$\Phi^{(k)} = \left(\frac{\phi_1}{\|\sqrt{B}\phi_1\|_2},\ldots,\frac{\phi_k}{\|\sqrt{B}\phi_k\|_2}\right). \tag{20}$$

Now the matrix defining the orthogonal projection onto the linear subspace $\mathcal{O}_2^{(k)}$, is defined in terms of the B-inner product as $P = I_{n_V} - \Phi^{(k)}(\Phi^{(k)})^T B \in \mathbb{R}^{n_V\times n_V}$, and the orthogonal projection of vector $g^{(\ell)}$ is given by

$$\begin{aligned} d^{(\ell)} &= \text{proj}_{\mathcal{O}_2^{(k)}}\left(g^{(\ell)}\right) \\ &= P\,g^{(\ell)} \\ &= g^{(\ell)} - \sum_{i=1}^{k}\Phi_i^{(k)}\left[\left(\Phi_i^{(k)}\right)^{\mathrm{T}} B_{i,i}\,g^{(\ell)}\right] \end{aligned} \tag{21}$$

where $\Phi_i^{(k)} \in \mathbb{R}^{n_V}$ denotes the column vector of $\Phi^{(k)}$ in (20). We note that all vectors $\Phi_i^{(k)}$ do not change along the M-PGD iterations, hence they can be computed once for all before starting to iterate. The orthogonal projection of any vector in $\mathbb{R}^{n_V}$ on the linear constraint $O_2^{(k)}$ in (18) not only admits an explicit solution, but has computational complexity $O(kn_V)$, where commonly $k \ll n_V$.

The main computational steps of the M-PGD method are summarized in Algorithm 1. The M-PGD is initialized with

Algorithm 1 M-PGD Algorithm Solving (17)-(18)

1. `Input:`
   $\phi_1,\ldots,\phi_k \in \mathbb{R}^{n_V}$ eigenfunctions in (20)
   step-size $\alpha > 0$
2. `Output:`
   ϕ_{k+1} new (B-orthogonal) eigenfunction
3. `Initialize:` $\ell = 0$, $t^{(0)} = t_0$,
4. `While (22) are not satisfied`
5. • `compute the gradient` $g^{(\ell)}$ `by (19)`
6. • `compute the projected gradient` $d^{(\ell)}$ `by (21)`
7. • `update` $t^{(\ell+1)} = t^{(\ell)} - \alpha d^{(\ell)}$
8. • `update` $\ell = \ell + 1$
9. `end while`
10. $\phi_{k+1} = t^{(\ell)}$

a random vector $t_0 \in \mathbb{R}^{n_V}$ and the iterations are stopped as soon as one of the two following conditions is fulfilled:

$$\frac{\|t^{(\ell+1)} - t^{(\ell)}\|_2}{\|t^{(\ell)}\|_2} < \text{Tol}, \qquad \ell > \text{Nits}. \tag{22}$$

The M-PGD method allows us to avoid degeneracy issues when multiple eigenfunctions share the same eigenvalue. Indeed, after obtaining each eigenvector, the method projects subsequent iterates to be orthogonal to all previously computed eigenvectors. This ensures convergence to independent eigenvectors, each within the degenerate eigenspace, [28].

Being the cost function $R(\phi)$ in (17) continuously differentiable and lower bounded, the convergence of the M-PGD algorithm to critical points of the problem is guaranteed by standard arguments when α belongs to a range of step-sizes that lead to sufficient decrease. The values for the step-size α in the reported experiments have been selected by trial and error.

V. dPINNs FOR GEOMETRIC LAPLACIAN EIGENDECOMPOSITION

Following the incremental strategy detailed in section IV, we propose LEP-dPINN to address the Laplacian eigenvalue problem (2) on a surface mesh M in $\mathbb{R}^3$ (an approximation of a compact 2-manifold $\mathcal{M}$). This framework leverages Generalized GNNs [12] due to their specialized capability for processing data on triangular meshes.

The network predicts the K eigenpairs $\{\phi_k, \lambda_k\}_{k=1}^K$ one at a time incrementally. In each pass, a new eigenfunction is predicted, ensuring its orthogonality by incorporating the eigenfunctions already computed. Specifically, at the $(k+1)$th iteration, the Generalized GNN predicts the solution to the optimization problem (17)-(18). It achieves this by exploiting special convolutional graph layers designed to minimize the Dirichlet energy over the subspace orthogonal to the span of the first k eigenfunctions, $\text{span}\{\phi_1, \ldots, \phi_k\}^\perp$, as detailed in section V-B.

The LEP-dPINN framework is designed as discrete PINNs over continuous PINNs because they are inherently suited for our problem: learning on a triangular surface mesh M. They directly leverage this discrete representation and numerical differentiation. In contrast, cPINNs are not ideal for problems where the spatial domain (2-manifold $\mathcal{M}$) is not readily available for arbitrary sampling. Although they can be adapted to this purpose by incorporating interpolation techniques, such as mesh parameterization or local patching on fixed point clouds, the approach is indirect and less efficient. Furthermore, the Euclidean space in which cPINNs operate and compute derivatives is ill-suited for LBO eigendecomposition on discrete manifolds.

A scheme of the LEP-dPINN architecture is depicted in Fig. 2. In analogy to the scheme of a generic PINN architecture, illustrated in Figure 1, the LEP-dPINN architecture is based on three key components: input spatial features, network architecture choice, and numerical derivatives computation, as depicted in the dashed boxes in Fig. 2. The input consists of the structure of the triangular mesh $M = (X, E, T)$ that approximates the 2-manifold $\mathcal{M}$. Topological information, encoded in the adjacency matrix W, are extracted from the set T of faces, enabling the network to learn directly from the mesh connectivity T and geometry X. The Generalized GNN then generates a function to be used as candidate eigenfunction solution $\widehat{\phi}$ to the given eigendecomposition PDE model ($L\widehat{\phi} = R(\widehat{\phi})B\widehat{\phi}$). Numerical derivatives are computed and integrated into the physics-informed loss function which enforces the physics of the Laplacian Eigenvalue problem by minimizing the residual of the equation $L\widehat{\phi} - \lambda B\widehat{\phi} = 0$, promoting the orthogonality conditions in (18) such that $\widehat{\Phi} \in \mathcal{O}_2$, and imposing suitable boundary conditions.

In the following a detailed description of the training procedure is given in section V-A, and the GGNN is illustrated in section V-B.

A. PHYSICS-INFORMED LOSS FUNCTION AND TRAINING

The central idea of physics-informed learning for our setting is to design a loss function $\mathcal{L}$ such that the minimizer

$$\Theta^* = \arg\min_{\Theta} \mathcal{L}(\Theta; \widehat{\phi}, \ldots)$$

gives rise to a model that predicts a good approximation of the Laplacian eigenfunction and, consequently, of the associated eigenvalue.

We denote the network's prediction of the $(k+1)$-th eigenfunction as $\widehat{\phi} = GGNN(X; W, \Theta)$, where Θ represents the trainable weights of the $(k+1)$-th block. The physics-informed loss function $\mathcal{L}$ enforces solutions of the Laplacian Eigendecomposition problem and relevant constraints, and reads as

$$\begin{aligned}\mathcal{L}(\Theta; \widehat{\phi}, \phi_1, \ldots, \phi_k) = \alpha_1 \mathcal{L}_{bc}(\Theta; \widehat{\phi}_b) \\ + \alpha_2 \mathcal{L}_{eigen}(\Theta; \widehat{\phi}) \\ + \alpha_3 \mathcal{L}_{ortho}(\Theta; \widehat{\phi}, \phi_1, \ldots, \phi_k)\end{aligned} \quad (23)$$

where the parameters $\alpha_i \geq 0$, $i = 1, 2, 3$, ϕ_b is the ϕ function evaluated at the mesh boundary, if any, and

$$\mathcal{L}_{bc}(\Theta; z) = ||z_b||_F^2, \quad (24)$$

$$\mathcal{L}_{eigen}(\Theta; z) = ||Lz - (z^T Lz)Bz||_F^2, \quad (25)$$

$$\mathcal{L}_{ortho}(\Theta; z) = ||z^T B[\phi_1 | \ldots | \phi_k]||_F^2. \quad (26)$$

More specifically, the formulation of $\mathcal{L}$ in (23) incorporates three key components:

- $\mathcal{L}_{bc}$ Boundary Condition Loss (24): when dealing with manifolds with boundaries, this term enforces Dirichlet boundary conditions. This ensures that the predicted eigenfunction $\widehat{\phi}$ evaluated at the boundary vertices ($z_b = \widehat{\phi}_b$) adheres to the prescribed boundary behavior.
- $\mathcal{L}_{eigen}$ Eigenvalue Equation Loss (25): enforces the physics of the Laplacian Eigenvalue problem, by minimizing the residual of the equation $L\widehat{\phi} - \lambda B\widehat{\phi} = 0$. The eigenvalue λ associated to $\widehat{\phi}$ is computed by the Rayleigh quotient $\lambda = R(\cdot)$ defined in (13) and L and B define $\bar{L}$ as in (7).
- $\mathcal{L}_{ortho}$ Orthogonality Loss (26): this term penalizes the projection of the current eigenfunction $\widehat{\phi}$ onto the span of previously computed eigenfunctions $\{\phi_1, \ldots, \phi_k\}$, enforcing B-orthogonality with respect to the preceding approximated eigenfunctions. For the first eigenfunction ($k = 1$), this term is omitted (i.e., $\alpha_3 = 0$).

Given the predictions ϕ_i, for $i = 1, \ldots, k$, of the preceding k eigenvectors, where each $\phi_i = GGNN(X, W; \Theta)$ is obtained from a distinct, pre-trained GGNN, the training

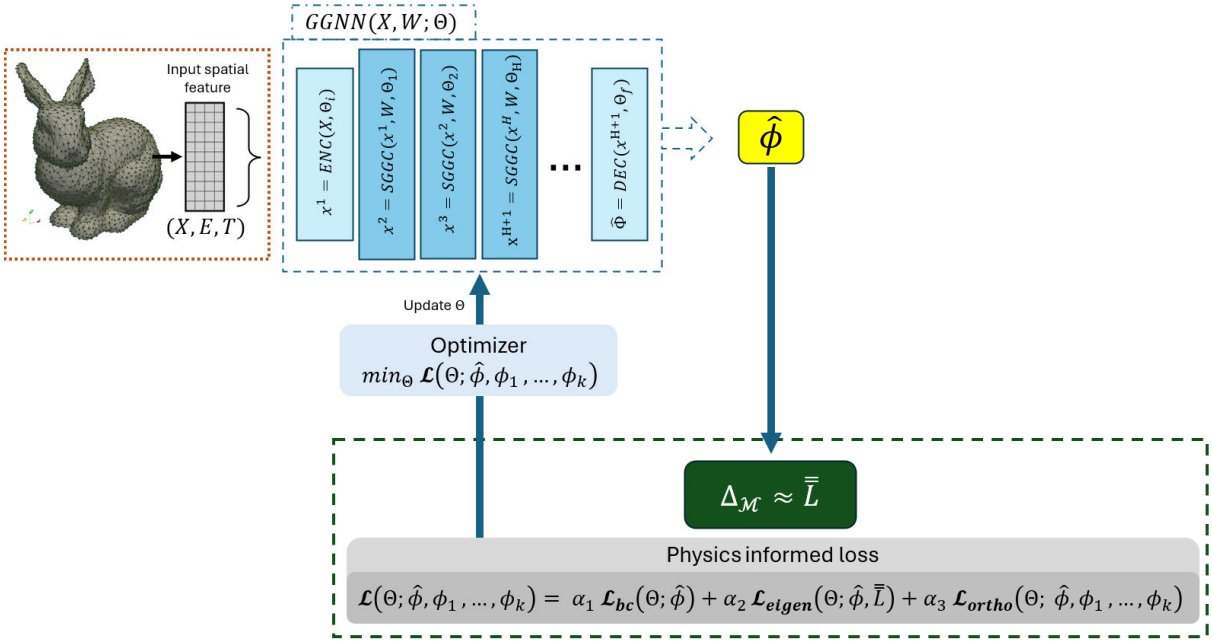


FIGURE 2. Schematic of LEP-dPINN architecture. The dashed boxes highlight the three key components of the PINN: input spatial domain (left), the GGNN approximating the candidate $\hat{\phi}$ solution (middle), and the discrete derivatives used within the physics-informed loss (bottom).

procedure to derive the model weights $\Theta^{(k+1)}$ involves the following steps:

1) Initialize the hyperparameters of the GGNN model: number of hidden layers H, number of hidden features h_f for each layer, learning rate l_r.
2) Forward-pass: the prediction $\tilde{\phi} = GGNN(X, W; \Theta)$ is normalized to have unit length $\hat{\phi} = \frac{\tilde{\phi}}{\|\tilde{\phi}\|_B}$, according to the B-norm (norm induced by the positive definite matrix B), see (9). While this may seem unnecessary, it helps in the selection of the hyperparameters.
3) Loss computation: numerical differentiation allows for computing the geometric Laplacian $\bar{\bar{L}}$. The three weighted contributions of the loss function $\mathcal{L}$ in (23) are then assembled.
4) Backward-pass: the gradient of the loss function with respect to the trainable weights Θ is computed, and the network parameters Θ are updated by leveraging on the ADAM optimizer.

Unlike the training procedure, in inference the only input required is the mesh data (vertex X and adjacency matrix W).

B. GENERALIZED GRAPH NEURAL NETWORK

This section details the neural network architecture at the core of our proposed LEP-dPINN, which is based on the Generalized GNN, introduced in [12].

GNNs represent a class of neural networks designed to operate on graph-structured data [29], [30], [31], effectively capturing the dependencies among nodes through message-passing mechanisms. Advanced researches have extended GNN applications to polygonal meshes, enabling their use in various domains, including PINNs [16], [22], [32].

The GGNN has been designed to represent functions on surface meshes $M = (X, E, T)$. In particular $GGNN: \mathbb{R}^{n_{in}} \to \mathbb{R}^{n_{out}}$ maps an input feature space of dimension $n_{in} = n_V \times 3$ to an output feature space of dimension $n_{out} = n_V$, parameterized by weights Θ. GGNN is a deep architecture composed of $H + 2$ functions (layers):

$$GGNN(X, W; \Theta) = DEC_{\Theta_{\text{out}}} \circ SGGC_{\Theta_H} \circ \cdots \circ SGGC_{\Theta_1} \circ ENC_{\Theta_{\text{in}}}(X; \Theta_{\text{in}}) \quad (27)$$

where $\Theta = \{\Theta_{in}, \Theta_1, \ldots, \Theta_H, \Theta_{out}\}$ are trainable parameters, and W is the adjacency matrix associated with M and used in (5) to define $\bar{L}$. Consequently, the GGNN leverages both extrinsic geometric information, given by X, and topological information, obtained by the adjacency matrix W.

The first *encoding layer* ($ENC_{\Theta_{in}}$) projects the low-dimensional input features into a higher-dimensional hidden space:

$$ENC_{\Theta_{in}}: \mathbb{R}^{n_{in}} \to \mathbb{R}^{n_V \times h_f}, \qquad x \mapsto \sin(x w_{in}^T + b_{in}), \quad (28)$$

where h_f denotes the number of hidden features of each layer. Here, $\Theta_{in} = \{w_{in}, b_{in}\}$ are trainable parameters with $w_{in} \in \mathbb{R}^{h_f \times 3}$ and $b_{in} \in \mathbb{R}^{n_V \times h_f}$. The sinusoidal activation function ($\sin(\cdot)$) in (28), and in the subsequent hidden layers, is strategically chosen to enhance input gradient variability and prevent gradient vanishing thereby promoting better network convergence and the capture of complex function behavior, see [33].

The core of the GGNN architecture consists of H *hidden layers*, denoted as $SGGC_{\Theta_j}$, for $j = 1, \dots, H$. Among the spectral-based convolution operators proposed in [12], we employ a slight modification of the *Smoothing Generalized Graph Convolution* (SGGC), defined as:

$$SGGC_{\Theta_j} : \mathbb{R}^{n_V \times h_f} \to \mathbb{R}^{n_V \times h_f}, \qquad x \mapsto \sin(x^I),$$

where x^I is obtained starting from $x^0 = x$, by iterating

$$x^{i+1} = x^i - [(x^i w_{j,1}^T) + 2((I - W)x^i w_{j,2}^T)], \quad i = 0, \dots \tag{29}$$

where $\Theta_j = \{w_{j,1}, w_{j,2}\}$ are trainable weights with $w_{j,1}, w_{j,2} \in \mathbb{R}^{h_f \times h_f}$. The first update term in (29) serves as feature mixture and involves $w_{j,1}$, while the second term, which involves $w_{j,2}$, spatially combines the data. This SGGC layer adapts the approach from [12] by substituting the geometric Laplacian with the unnormalized graph Laplacian $\bar{L} = I_{n_V} - W$, in (5), and by employing the $\sin(\cdot)$ nonlinear activation function. This variant eliminates the need to compute stiffness and mass matrices during inference, allowing us instead to rely solely on topological information while embedding domain-specific knowledge into the network weights through training.

The defined convolutional operator $SGGC_{\Theta_j}$ produces x^I which approximates the minimizer of the following energy function

$$E(x; w_{j,1}, w_{j,2}) := \|x\, w_{j,1}\|_F^2 + \frac{1}{2} tr(w_{j,2}^T x^T \bar{L} x\, w_{j,2}). \tag{30}$$

This energy is the sum of a smoothing term, in Frobenius norm, with the (weighted) discrete Dirichlet energy and is strictly related to the functional in (17). In fact the Rayleigh quotient minimized in (17) is the normalization of the Dirichlet energy by the total energy of the function. As shown in [12], the minimizer x^I is numerically computed by applying the gradient descent method to the functional $E(x; w_{j,1}, w_{j,2})$ in (30), namely, by iterating (29). This provides a rationale on how SGGC convolution effectively enforces the Laplacian eigendecomposition.

Finally, the *decoding layer* ($DEC_{\Theta_{out}}$), maps the high-dimensional hidden features back to the desired output space:

$$DEC_{\Theta_{out}} : \mathbb{R}^{n_V \times h_f} \to \mathbb{R}^{n_{out}}, \qquad x \mapsto x w_{out}^T + b_{out},$$

where $\Theta_{out} = \{w_{out}, b_{out}\}$ are trainable parameters with $w_{out} \in \mathbb{R}^{1 \times h_f}$ and $b_{out} \in R^{n_{out}}$. This linear layer facilitates the extraction of task-specific information, embedding the learned high-level representations into the appropriate output dimensionality, such as the predicted eigenfunction values at each vertex.

VI. NUMERICAL RESULTS

To assess the performance of the proposed eigendecomposition methods we present numerical experiments on two distinct training dataset types: multi-instance and single-instance. *Multi-instance* datasets are applicable when the 2-manifold-domain of the LBO possesses an analytic or parametric representation. This allows us to easily generate numerous instances from a single 2-manifold representation by simply varying the position or the number of vertices of the mesh approximating it. Conversely, the term *single-instance* refers instead to cases where one aims to determine the LBO eigenfunctions of a 2-manifold-domain for which only a single, free-form polygonal mesh representation is available. Consequently, the neural network cannot be trained using multiple meshes representative of the same object. We point out that the latter case is certainly the most frequent in applications.

The GGNN architecture in the LEP-dPINN is equipped with $H = 4$ SGGC layers, $h_f = 128$ hidden features and the learning rate is initially set to $l_r = 1.0 \times 10^{-3}$, and then decreased by a factor of 1×10^{-1} at epochs 20, 50, and 100, for a total of 250 epochs and batch size 10. Unless otherwise specified, we used the same hyperparameters for all examples. We tuned the hyperparameters α's in the loss function (23) by trial-and-error, while keeping α_3 much larger with respect to the values α_1 and α_2 to preserve the orthogonality constraints. However, we observed that their values are relatively insensitive to the shape of the input surface meshes M. Our code was implemented in Python 3.12.2, using PyTorch 2.5.1, and run on a machine equipped with an 8Gb Nvidia Quadro P4000. Given the available computational power, we decided to have the LEP-dPINN predict a set of the first K eigenfunctions, where K is 10% of the mesh's vertex number.

The following sections detail our experimental setup and results. Specifically, section VI-A introduces the 2D and 3D datasets generated. We then present results for meshes discretizing a domain in $\mathbb{R}^2$ in section VI-B, followed by results for surface meshes in $\mathbb{R}^3$ with (section VI-C) and without (section VI-D) parametric representations. Finally, in section VI-E we discuss an ablation study on key hyperparameters.

For the quantitative validation of the network performance, we report the Mean Squared Error (MSE) between each analytic reference eigenfunction ϕ_{GT} and its approximated eigenfunction ϕ_* as

$$\mathrm{MSE}(\phi_{GT}, \phi_*) = \frac{1}{n_V} \sum_{i=1}^{n_V} \big(\phi_{\mathrm{GT}}(i) - \phi_*(i)\big)^2. \tag{31}$$

where ϕ_* is either ϕ_{FEM}, ϕ_{MPGD} or ϕ (network prediction). We do not adjust for sign flips or for rotations within degenerate eigenspaces; we manually select corresponding eigenfunctions to compute the MSE metric.

A. DATASETS

A multi-instance dataset enables different samplings of a given 2-manifold representation.

The first dataset is sampled on a 2D `square` domain $\Omega = [0, \pi]^2$ and consists of triangular mesh discretizations whose geometry and topology vary across samples. Only

10% of the vertices lies on the boundary of each mesh. Vertex locations are randomly generated, allowing for samples with highly variable mesh quality, potentially including triangles with areas close to zero. The training set contains 200 mesh samples, each with $n_V = 144$ vertices. In the following, we refer to this dataset as the *square-training-dataset*. In a similar way, we generate two test datasets:

- *square-test-dataset-A*: 8 mesh samples with an increasing number of vertices $n_V = \{144, 162, 236, 324, 468, 700, 932, 1164\}$;
- *square-test-dataset-B*: 20 mesh samples with a fixed number of vertices, $n_V = 932$, randomly located over the square.

The second dataset consists of triangular mesh discretizations of a `sphere` in $\mathbb{R}^3$, with radius 1. The training dataset consists of 200 mesh samples, each with $n_V = 162$ vertices and $n_T = 320$ triangles, with varying vertex positions. In the following, we refer to this dataset as the *sphere-training-dataset*. Then, we generate a test dataset, *sphere-test-dataset*, comprising 3 mesh samples with an increasing number of vertices in the range set $n_V = \{252, 752, 1502\}$.

The third dataset, named *vase-training-dataset*, contains 100 samples each of $n_V = 500$ vertices of a `vase` in $\mathbb{R}^3$, which has a parametric representation. The test dataset, *vase-test-dataset*, includes 2 mesh samples with an increasing number of vertices, $n_V = \{520, 2280\}$.

In the single-instance case we consider discrete 2-manifolds without boundary, represented by triangular meshes $M(n_V, n_T)$ approximating `bunny1`(102,199), `bunny2`(502,1000) `horse`(152,300) and `torus`(100,200) shapes, without an associated parameterization. Consequently, it is not straightforward to generate multiple meshes that approximate the same object. In this case the network's parameters are optimized to predict the eigenfunctions solely for that specific object.

B. EXAMPLE 1 - MESHES IN $\mathbb{R}^2$

This example assesses the performance of the proposed LEP-dPINN model by comparing the predicted Laplacian eigenpairs on the square 2D domain $\Omega = [0, \pi]^2$ with both analytical eigenpairs and results from FEM computations. We trained the model on the *square-training-dataset* and perform comparisons using samples from the *square-test-dataset-A* and *square-test-dataset-B*.

Laplacian eigendecomposition on Ω, considering Dirichlet homogeneous boundary conditions, is known and given by, for $n, m = 1, 2, 3, \ldots$

$$\phi_{n,m}(x, y) = \frac{2}{\pi} \sin(nx) \sin(my), \quad \lambda_{n,m} = n^2 + m^2. \quad (32)$$

Fig. 3 shows the predicted solutions for the first five eigenpairs $(n, m) = (1, 1), (1, 2), (2, 1), (2, 2), (1, 3)$ of the 2D Laplacian eigenvalue problem on the test sample from *square-test-dataset-A* with $n_V = 324$, ordered by ascending eigenvalue. As for this particular sample, we in general observe that the LEP-dPINN performs better than FEM on samples exhibiting low-quality mesh discretization. However, for high-quality mesh discretizations the accuracy of the two methods is comparable.

TABLE 1. RMSE results for the LBO eigendecomposition on 2D square domain: (left panel) LEP-dPINN estimated the K eigenpairs on the test sample from *square-test-dataset-A* with $n_V = 144$; (right panel) results from Table 2 in [19].

	Mean RMSE over efs.		
	LEP-dPINN		[19]
K	training	testing	
1	1.07×10^{-5}	5.92×10^{-5}	3.2×10^{-4}
2	6.09×10^{-5}	9.81×10^{-5}	9.1×10^{-4}
3	5.32×10^{-4}	5.96×10^{-4}	2.0×10^{-3}
4	7.55×10^{-4}	9.02×10^{-4}	1.0×10^{-3}

Next, we examine whether LEP-dPINN maintains robustness and invariance with respect to both mesh resolution and geometry through targeted stress tests. The selected test datasets include meshes with higher vertex counts than those in the training dataset, as well as widely varying element quality, including triangles of near-zero areas. The latter test intentionally challenges the model in regimes where linear FEM is known to exhibit limitations.

In Fig. 4, we illustrate the estimated solution of the fifth eigenfunction, $(m, n) = (3, 1)$ in (32), on several test samples from *square-test-dataset-A*. As the number of mesh vertices increases, the meshes undergo topological and geometric changes, resulting in variations in triangle area and formation of smaller, degenerate triangles. It is noteworthy that, despite the slight improved accuracy of the eigenvalues computed by FEM, the MSE(ϕ_{GT}, ϕ) consistently remains lower than the MSE(ϕ_{GT}, ϕ_{FEM}) not only for the small subset here reported, but even, in general, for all the different mesh resolutions examined. This consistent trend strongly indicates that our proposal exhibits superior stability with respect to domain discretization and, crucially, is less susceptible to mesh quality variations that can compromise FEM accuracy.

To further highlight this aspect, we conducted tests on *square-test-dataset-B*. This dataset features a fixed vertex count and randomized mesh geometry, allowing for a direct assessment of sensitivity to geometric variations. On this dataset, we obtain average MSE values $avg(MSE(\phi_{GT}, \phi)) = 6.81 \times 10^{-4}$ and $avg(MSE(\phi_{GT}, \phi_{FEM})) = 2.23 \times 10^{-2}$, alongside average eigenvalues $avg(\lambda_{3,1}) = 9.74$ versus $avg(\lambda_{3,1,FEM}) = 9.84$ in case of FEM-computed eigenvector. These results reinforce our conclusion that LEP-dPINN offers a more stable and reliable approach for Laplacian eigendecomposition, especially when mesh quality and geometric variability are concerns, providing a significant advantage over traditional FEM methods, which are more susceptible to variations that can negatively affect their accuracy.

As further demonstration of the LEP-dPINN robustness to mesh quality, we now consider the test mesh sample with

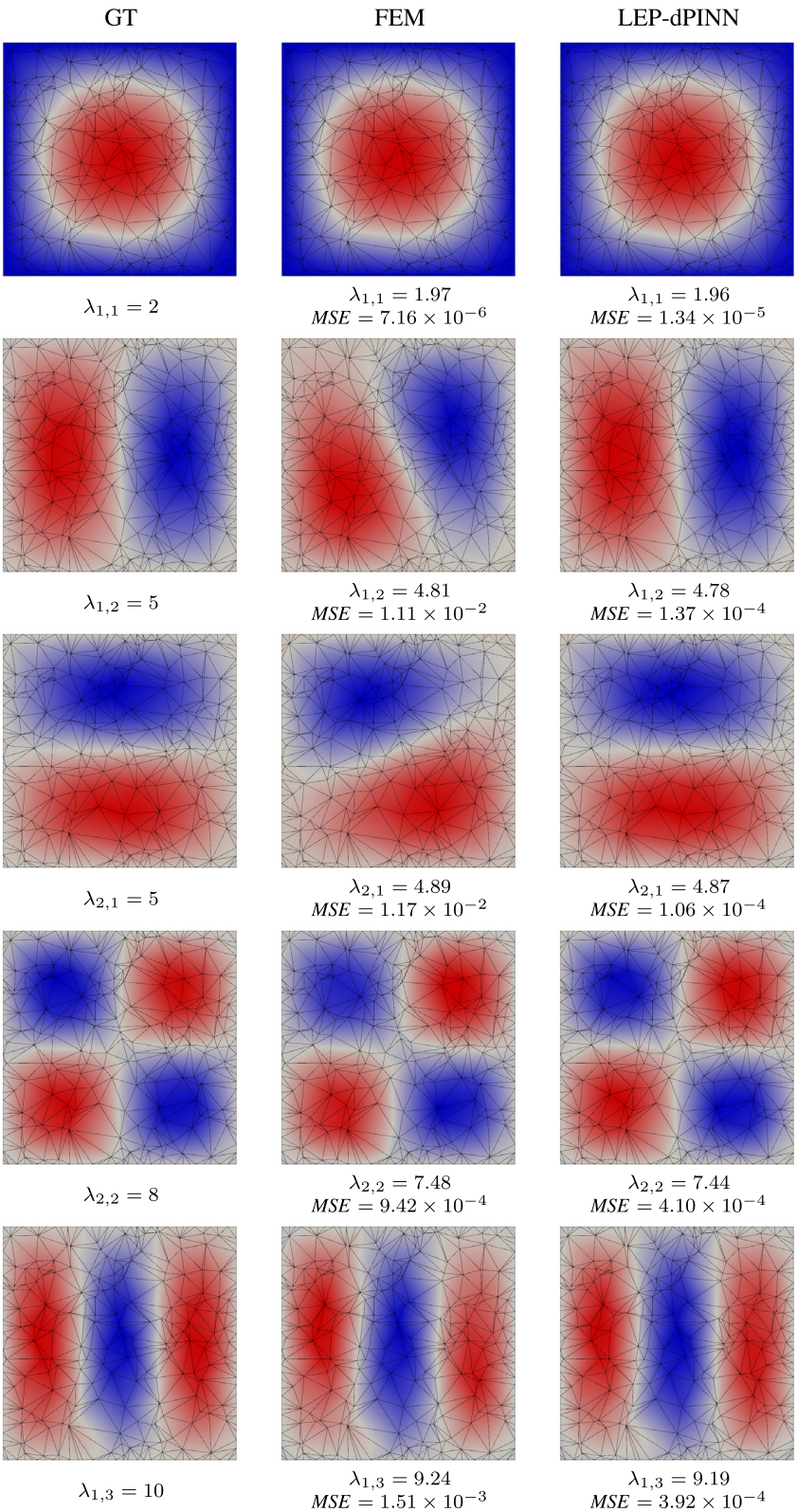


FIGURE 3. Example 1. Estimate of the first five eigenpairs on the test sample from *square-test-dataset-A* with $n_V = 324$. Ground truth (ϕ_{GT}) first column, ϕ_{FEM} second column, and ϕ computed by LEP-dPINN, third column. For each method, the corresponding eigenvalues $((n, m) = (1, 1), (1, 2), (2, 1), (2, 2), (1, 3))$ are reported, together with MSE with respect to ϕ_{GT}.

$n_V = 162$ vertices irregularly located in the square domain Ω, see Figure 5, right panel. In Figure 5 (left panel) we report a graph of the MSE(ϕ_{GT},ϕ_i) for $\phi_i, i = 1, .., K$, with $K = 16$, together with a plot of the relative errors on the

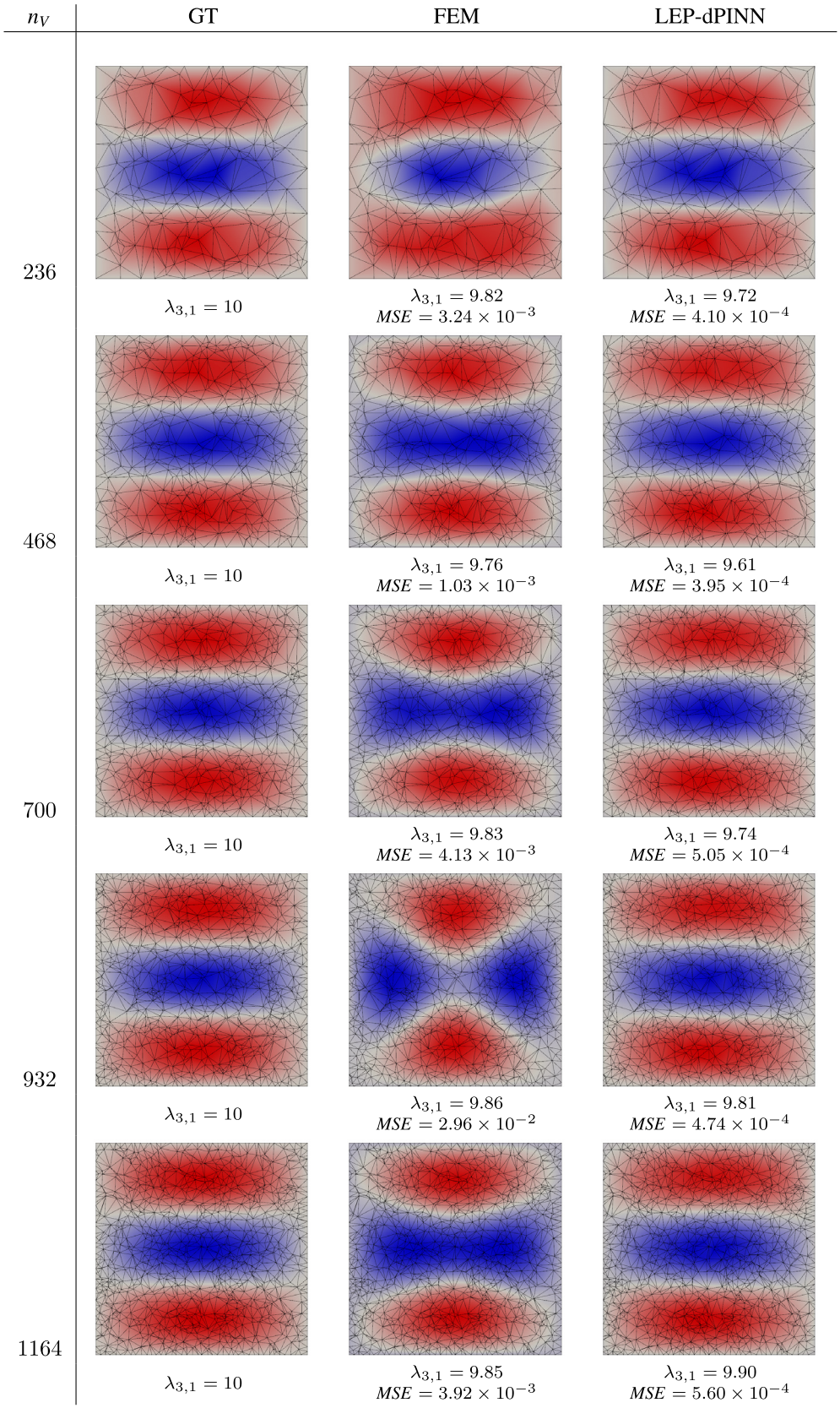


FIGURE 4. Example 1. Estimate of the sixth eigenpairs for increasing mesh resolution - number of vertices n_V. Ground truth (ϕ_{GT}) first column, ϕ_{FEM} second column, and ϕ computed by LEP-dPINN third column. For each method, the corresponding eigenvalues ($\lambda_{3,1}$) are reported together with the MSE values with respect to ϕ_{GT}.

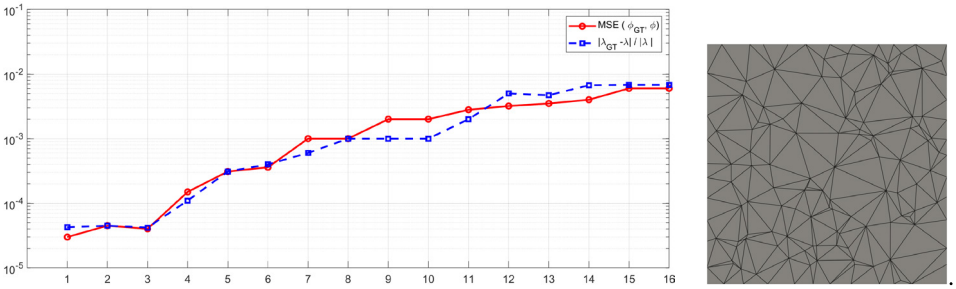


FIGURE 5. First 16 eigenpairs: (left) MSE(ϕ_{GT}, ϕ) eigenfunctions (dashed blu plot) and relative errors on the eigenvalues (solid red plot); (right) test sample with $n_V = 162$ vertices.

corresponding eigenvalues. Despite the non-regularity of the mesh the error values remain very low and stabilize as the number of predicted eigenpairs increases.

Furthermore, in the previous stress tests, we demonstrated the computational advantage of the proposed learning-based approach in scenarios where, for instance, one is required to compute an eigenpair for an unseen mesh discretization of the square 2D domain. In such cases, the FEM pipeline requires recomputing the stiffness and area matrices for the new mesh, followed by the solution of a linear system to recover the full set of eigenpairs—even if only a single eigenpair is of interest. In contrast, LEP-dPINN requires only a forward pass through the already trained network, using the vertex coordinates and the adjacency matrix of the unseen mesh, thereby offering a significant computational speed-up.

Finally, we compare with the results by Ben-Shaul et al. [19], where the authors present a cPINN-type approach for the Laplacian eigendecomposition on 2D square domain. Ben-Shaul et al. [19] use a SIREN [34] network (7 layers each with 100 neurons) to compute the first four eigenpairs. Their method involves training simultaneously four networks, one for each eigenpair, using a single loss function. In Table 1, we report the Root Mean Square Errors (RMSE), averaged over the number K of computed eigenfunctions, for both the LEP-dPINN and the approach by Ben-Shaul et al. [19]. This highlights a key advantage: the proposed LEP-dPINN approach not only learns from the training data but also effectively predicts accurate eigenfunctions on unseen mesh geometries, showcasing generalization ability.

C. EXAMPLE 2 - PARAMETRIC MESHES IN $\mathbb{R}^3$

This example focuses on evaluating the robustness of the LEP-dPINN model to variations in the approximation accuracy of the underlying mesh in $\mathbb{R}^3$, specifically by using various mesh discretizations that approximate parametric surfaces with increasing accuracy. The parametric representation allows us to use the multi-instance approach for the LEP-dPINN.

At this aim, we consider first a regular triangulation of the unit sphere without extraordinary vertices, thereby minimizing the influence of geometric irregularities. In this case, the analytical eigenfunctions of the Laplacian on the unit sphere S^2 embedded in $\mathbb{R}^3$ are known as *Spherical*

TABLE 2. MSE(ϕ_{GT}, ϕ) values for the ablation study. Evaluation of LEP-dPINN architectural components for an LBO eigenpair prediction.

layer	activation	h_f	MSE(ϕ_{GT}, ϕ)
SGGC	sin	**128**	$\mathbf{5.07 \times 10^{-6}}$
GAT	-	-	1.14×10^{-2}
GCN	-	-	2.23×10^{-3}
SGGC	tanh	128	2.55×10^{-1}
-	ReLU	-	3.71×10^{-3}
SGGC	sin	256	4.14×10^{-5}
-	-	512	6.68×10^{-3}
-	-	1024	1.21×10^{-1}

TABLE 3. *MSE*(ϕ_{GT}, ϕ) values for different upper limits for the first layer $[0, \gamma_1]$ and subsequent hidden layers $[0, \gamma_2]$. The bold MSE value corresponds to the baseline.

$[0,\gamma_1]$ \ $[0,\gamma_2]$	$[0, 0.5]$	$[0, 0.005]$
$[0, 1]$	5.08×10^{-3}	$\mathbf{5.07 \times 10^{-6}}$
$[0, 5]$	6.16×10^{-2}	1.15×10^{-2}
$[0, 10]$	6.18×10^{-2}	1.36×10^{-1}

Harmonics and are defined by the following equations:

$$\phi_{m,n}(\theta, \phi) = NP_n^m(\cos(\theta))e^{im\phi}, \tag{33}$$

where N is a normalization constant, $(\theta, \phi) \in [0, \pi] \times [0, 2\pi[$ are the spherical coordinates P_n^m are the associated Legendre polynomials [35], and $\lambda_{m,n} = n(n+1)$, are their corresponding eigenvalues, for $-m \leq n \leq m$, $m = 0, 1, 2, 3, \ldots$. The spherical harmonics have been computed by using the function `scipy.special.sph_harm_y` from the Python library SciPy [27].

In Fig. 6, we illustrate the LEP-dPINN predictions for the test samples in the *sphere-test-dataset* with increasing resolution, by training on the multi-instance dataset *sphere-training-dataset* ($n_V = 162$). We show comparisons with results obtained by the analytical spherical harmonics (33), the FEM, and the variational M-PGD method. Each plot reports the MSE value, comparing either the LEP-dPINN, FEM or M-PGD results to the analytical ground truth in (33).

Notably, Figure 6 demonstrates strong generalization capabilities: trained on samples with only $n_V = 162$ vertices, the LEP-dPINN still yields satisfactory results on test samples with a significantly increased vertex count, up to $n_V = 1502$.

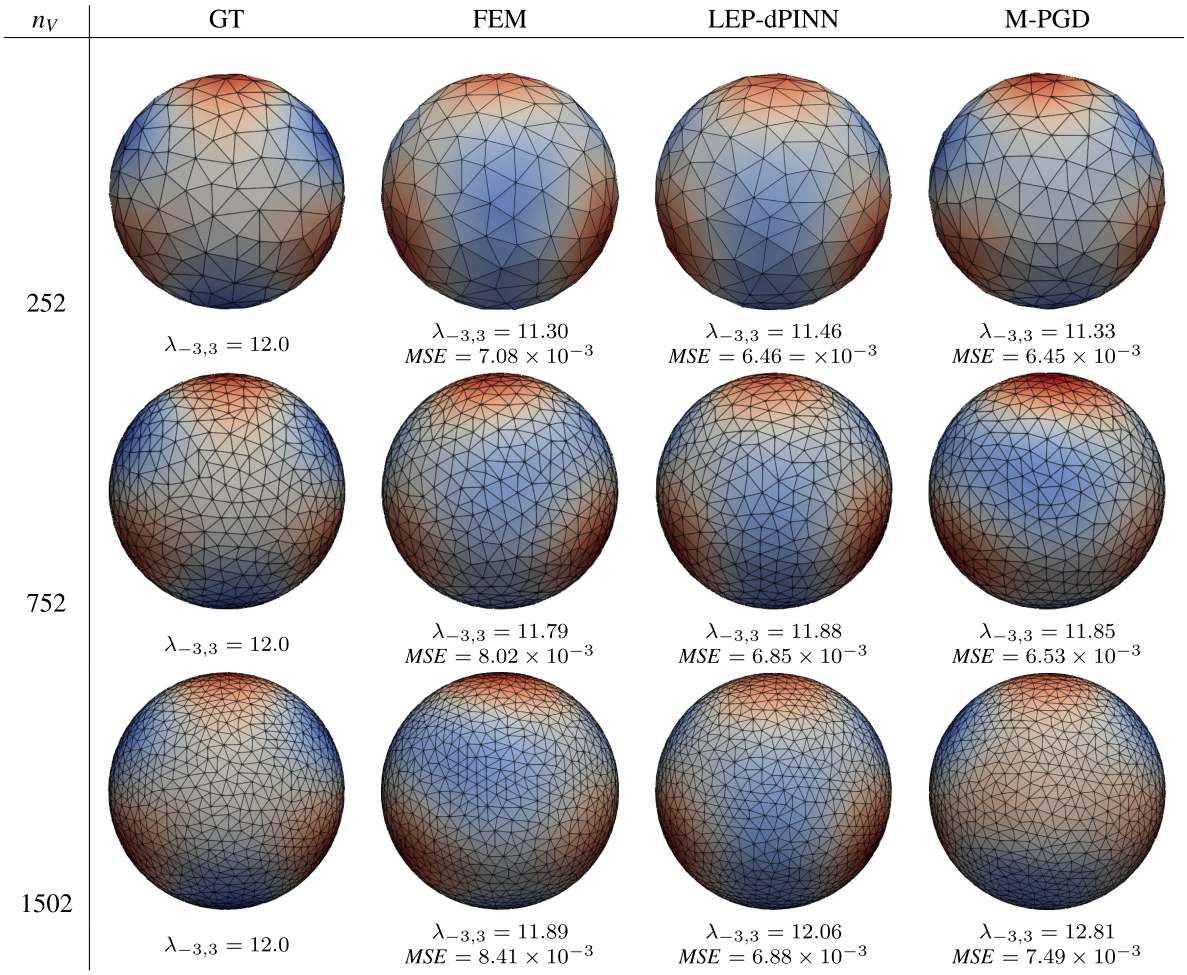


FIGURE 6. Example 2: robustness to mesh resolution for the `sphere` shape. Comparison of the ninth eigenfunction ϕ_{GT} with the computed ones ϕ_{FEM}, ϕ_{MPGD} and ϕ - obtained by LEP-dPINN- for different mesh resolutions (varying n_V). The corresponding eigenvalue and MSE are reported below.

Similar results can be observed in case of generic free form parametric surfaces. We show in Figure 7 the LEP-dPINN predictions for the test samples in the *vase-test-dataset* with increasing resolution, $n_V = 520$ - first row - $n_V = 2280$ - second row - obtained by training the network on the multi-instance dataset *vase-training-dataset* with $n_V = 500$.

The predictions obtained by LEP-dPINN can be thus considered discretization-independent since LEP-dPINN takes as input, in the inference stage, only vertices' coordinates and the adjacency matrix of the unseen mesh. Differently, both the FEM and the M-PGD method need the computation of the stiffness and mass matrix of the unseen mesh. This inherent capability to generalize across varying discretizations, without the burden of recomputing domain-specific matrices, underscores a key advantage of LEP-dPINN in approximating continuous fields on discrete manifolds, moving beyond the limitations of traditional discretization-dependent methods like FEM and M-PGD.

D. EXAMPLE 3 – NON-PARAMETRIC MESHES IN $\mathbb{R}^3$

In this example, we apply LEP-dPINN to compute the leading geometric Laplacian eigenfunctions of free-form meshes `bunny1`, `horse`, and `torus`, for which it is assumed that a parametric representation is not available.

Therefore, LEP-dPINN can only be applied in single-instance mode, with the same network hyperparameters as for the previous examples, except for using $h_f = 256$ hidden features and training for a larger number of epochs, approximately 2000 epochs.

Fig. 8 illustrates four eigenpairs approximated by using FEM, M-PGD and LEP-dPINN approaches, ordered by ascending eigenvalue. For each eigenfunction, the associated eigenvalue is reported. We set the value of the step-size α for the M-PGD method to 0.001 for the `bunny1` and `torus` shapes, while to 0.0001 for the `horse` shapes.

Upon visual inspection of Fig. 8, and considering the associated eigenvalues, we can observe that for these free-form domains in $\mathbb{R}^3$, LEP-dPINN is able to compute eigenpairs with a level of accuracy that closely approximates, and is qualitatively comparable to, that achieved by the well-established FEM. This indicates the strong potential of LEP-dPINNs for accurate and efficient spectral analysis despite this extreme lack of data.

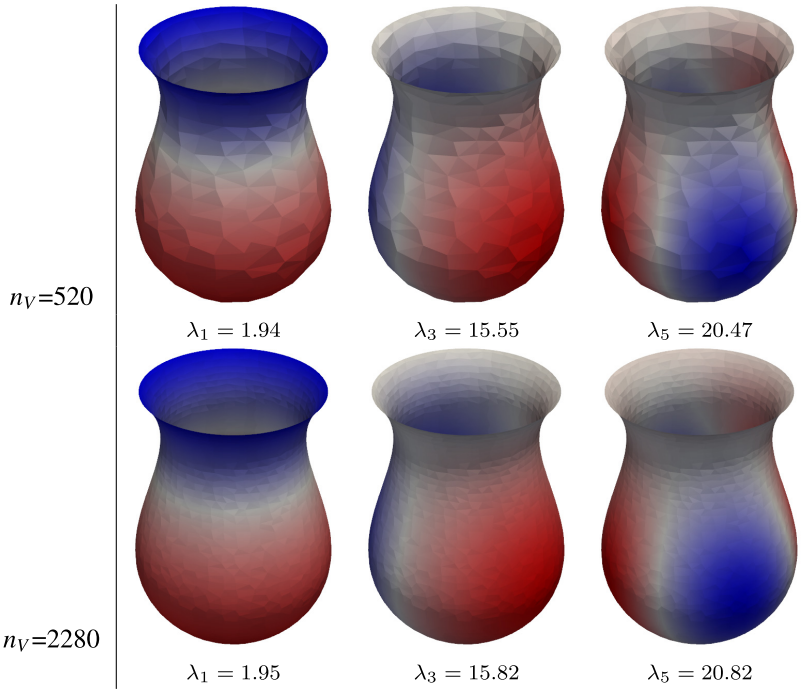


FIGURE 7. Example 2: robustness to mesh resolution for the `vase` shape. First row: three eigenfunctions on the test sample with $n_V = 520$ vertices. Second row: corresponding eigenfunctions predicted by LEP-dPINN on a refined test mesh with $n_V = 2280$ vertices.

It is well known that the LBO eigenfunctions on a 2-manifold embedded in $\mathbb{R}^3$ are invariant under isometric transformations of the manifold. By considering as network input features the length of the mesh edges, the proposed LEP-dPINN is made transformation invariant, i.e. it learns mesh-based eigenfunctions that are inherently invariant to isometric transformations such as rotation and translation of the input meshes. However, the eigenfunctions are not invariant under scaling of the manifold $\mathcal{M}$. The rescaling according to the result in Prop. 1 allows us to obtain scale invariant eigenfunctions.

In Fig. 9, we show qualitative results for testing the computation of the fifth eigenfunction associated with the `bunny2` shape. The small panels on each mesh indicates the affine transformations the sample mesh undergoes: $\mathbf{T}$ indicates a translation, $\mathbf{R}\circ\mathbf{T}$ the composition of a translation with a rotation and $\mathbf{S}\circ\mathbf{T}$ the composition of a translation with a scaling, by a scaling factor of 2. By leveraging such an intrinsic property the eigenfunction computation is invariant to the pose of the shape in the three-dimensional space.

E. ABLATION STUDY

To validate the specific design choices of the GGNN within our LEP-dPINN framework and understand the contribution of its key components, we perform an ablation study. We systematically investigate the impact of altering the core graph convolutional layer, the activation function, and hyperparameters such as hidden feature dimensionality and weight initialization. The results are reported as MSE values in Tables 2 and 3. A dash symbol indicates that the corresponding component is identical to that in the previous row, and bold fields indicate the baseline.

The study focuses on computing a single eigenpair of the LBO for the test sample from *square-test-dataset-A* with $n_V = 324$, after training the network on the *square-training-dataset*. Our baseline model consists of 4 hidden SGGC layers, $h_f = 128$ hidden features, and uses the $\sin(\cdot)$ activation. It corresponds to the first row of Table 2. The baseline weights are initialized as follows: encoding layer weights from a uniform distribution over the interval $[0, 1]$, hidden layers' weights from a uniform distribution over the interval $[0, 0.005]$, and decoding layer weights using Kaiming initialization.

The first relevant architectural design choice is related to the kind of *hidden layers*. The SGGC layer was selected based on the premise that it incorporates an inductive bias suitable for the task. As the LBO eigenpairs are critical points of the Rayleigh quotient (13), which represents an energy functional (16), the SGGC layer's formulation [12] is designed to promote features that naturally align with minimizing such energy functionals and extracting eigencouples corresponding to the leading eigenvalues in the spectrum. To test this hypothesis, we replaced the SGGC layers in our baseline model with standard Graph Convolutional Network (GCN) layers [36] and Graph Attention Network (GAT) layers [37]. As shown in the first three rows of Table 2, the performance, measured in terms of MSE values, degrades drastically when using GCN or GAT layers.

PINNs face significant challenges related to *activation functions*. Commonly used functions such as `tanh` and

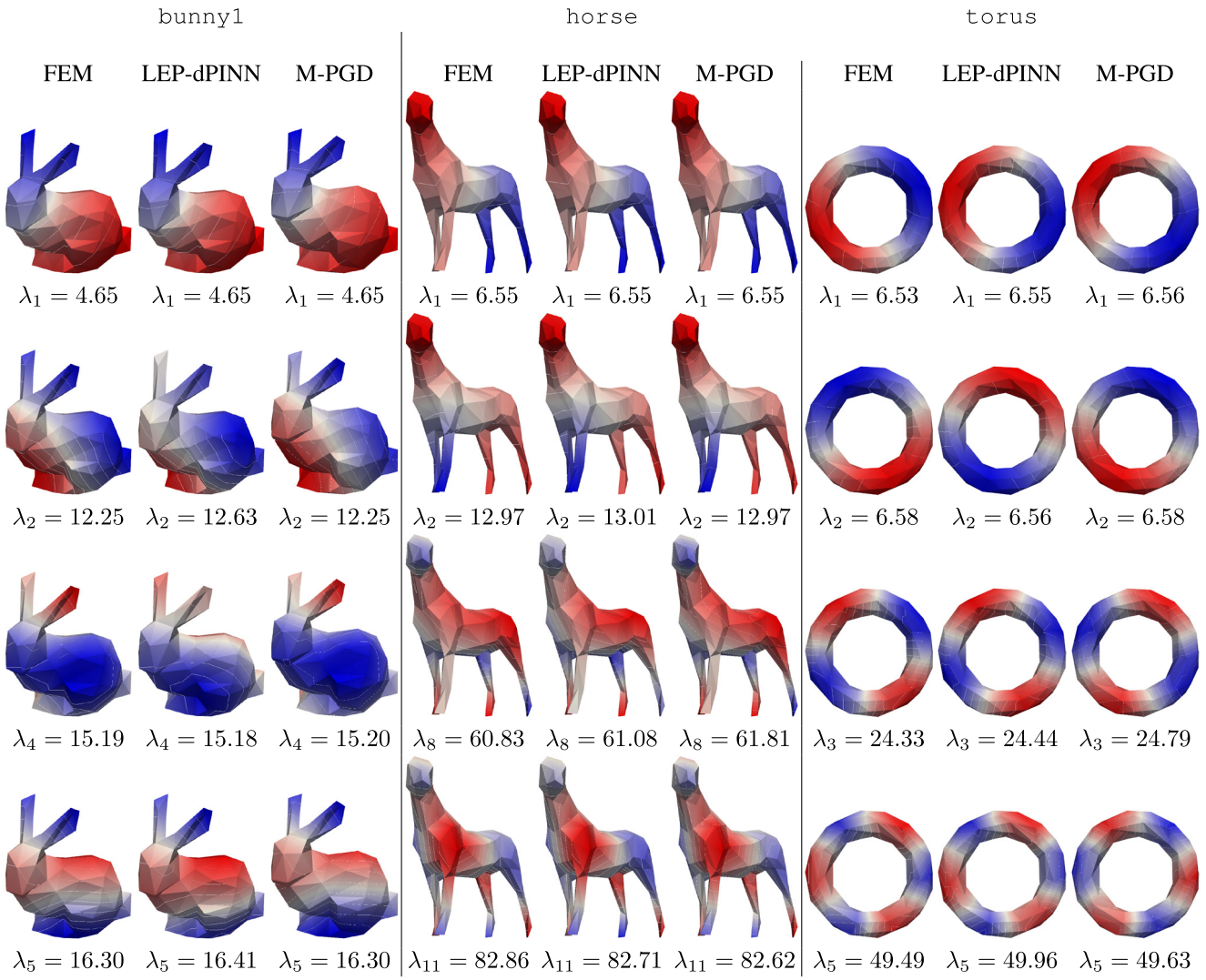


FIGURE 8. Example 3: comparison of the leading eigenfunctions for different 3D shapes. For each shape (`bunny1`, `horse`, `torus`), eigenfunctions computed ϕ_{FEM}, ϕ_{MPGD} and ϕ - obtained by LEP-dPINN- are shown alongside with the corresponding eigenvalues.

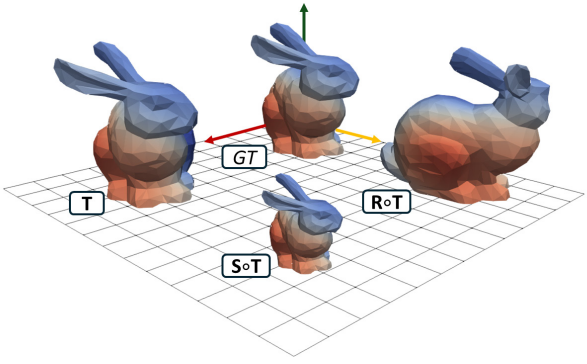


FIGURE 9. Example 3. Invariance to pose: fifth eigenfunction of the `bunny2(503, 1000)` mesh computed for different poses:T indicates translation, R∘T composition of translation with rotation and S∘T composition of translation with scaling, by a scaling factor of 2.

`ReLU` tend to saturate, contributing to the vanishing gradient problem and limiting the network's ability to adapt during training [38]. On the other hand, sinusoidal activation functions have demonstrated significant advantages in various works [18], [19], [34]. They enhance input gradient variability and maintain higher gradient magnitudes during both initialization and training, which helps prevent the network from stagnating in suboptimal solutions. The effectiveness of the `sin` activation is highlighted in Table 2. Comparing MSE values in the first row of the table (baseline) with those in the fourth and fifth rows, corresponding to the `tanh` and `ReLU` activation, there is an MSE improvement of several orders of magnitude.

Another design choice we investigated is the *number of hidden features* h_f. An insufficient number of hidden features might lack the capacity to adequately approximate the target function, whereas an excessive number can lead to increased optimization challenges without necessarily obtaining optimal results. Moreover, we highlight that for more complex geometries a higher number of hidden features is required to gain better performance. This trade-off is explored empirically in Table 2 where the MSE value for

$h_f = 128$ (first row) is several orders of magnitude lower than those for $h_f = 256, 512, 1024$ (last three rows). This suggests that $h_f = 128$ features are sufficient for optimal representation.

Finally, we carefully tuned the *weight initialization ranges* for the encoding (first) and subsequent hidden (SGGC) layers. For instance, accurately representing eigenpairs associated with larger eigenvalues typically requires initializing weights from progressively wider uniform distribution. Novello et al. [33] observed that the first layer's weights affect the number of harmonic frequencies characterizing the neural representation power while the weights of the hidden layers primarily control the amplitudes associated with these frequencies, providing a mechanism to bound them. Inspired by these observations, we explored the interplay between the weight initialization ranges from a uniform distribution $[0, \gamma_*]$, used for the first layer versus the hidden layers. Table 3 shows the resulting $MSE(\phi_{GT}, \phi)$ values for various γ_* both for the first layer, $[0, \gamma_1]$, and the hidden layers, $[0, \gamma_2]$ (applied to all hidden layers). The results suggest a trade-off: wider initialization ranges for the first layer introduce more high-frequency components, which may require narrower ranges (smaller γ_2) for the hidden layers to appropriately attenuate their amplitudes and achieve optimal performance.

VII. CONCLUSION

In this paper, we introduced LEP-dPINN, a physics-informed discrete Graph Neural Network framework. It is designed to compute Laplace-Beltrami eigenpairs on 2-manifolds represented by triangular meshes.

Our approach successfully leverages the strengths of GNNs to learn directly from mesh geometry and topology, addressing some of the computational burdens and mesh-dependency issues associated with classical numerical methods like FEM and M-PGD.

Numerical experiments were conducted on a variety of surface meshes. These meshes discretized domains both in $\mathbb{R}^2$ and in $\mathbb{R}^3$ and included analytically defined shapes as well as surfaces lacking parametric formulations. The results of these experiments consistently demonstrated the efficacy of the LEP-dPINN framework. Ablation studies further highlighted the importance of specific architectural choices, such as the use of SGGC layers and sinusoidal activation functions, for achieving optimal performance.

The method achieved accurate approximations of eigenfunctions and eigenvalues, comparable to, and in some cases exceeding, the performance of FEM-based solutions, particularly under variations in mesh resolution and quality. In the case the domain of the LBO eigendecomposition has a parametric representation, the proposed LEP-dPINN approach presents a significant practical advantage. Specifically, the ability to generalize to unseen mesh discretizations without recomputing large matrices: the approach is well-suited for scaling LBO decomposition to domains with significantly higher vertex counts, with minimal additional computational effort. Without a parametric representation, the method maintains good accuracy but suffers from poor generalization. Because it is also computationally expensive, we were restricted to processing low-resolution meshes. In our setup, a separate network block is trained for each eigenfunction ϕ_k, $k = 1, \ldots, K$, which is considerably less expensive than training a single monolithic model with K output channels.

However, like many deep learning approaches, the performance of LEP-dPINN remains sensitive to hyperparameter settings, including network depth, feature dimensionality, and weight initialization. Additionally, we observed that achieving good performance often requires a relatively deep network, even on meshes with as few vertices. We attribute this to the inherent difficulty of learning high-dimensional orthogonal vectors.

Future research could focus on several promising directions. Defining a more efficient strategy to gain orthogonality may enable the design of lighter and more easily tunable models. Developing methods to exploit precomputed weights for the prediction of the subsequent LEP-dPINN iteration (eigenfunction), such as following a fine-tuning procedure of the already trained network blocks, could significantly enhance efficiency. Exploring adaptive or automated hyperparameter optimization techniques could alleviate the tuning burden. Further investigation into the theoretical convergence properties and error bounds of such GNN-based PDE solvers on discrete manifolds would also provide valuable insights and strengthen the method's foundations. Finally, increasing the computational power will allow the processing of larger dimension meshes.

REFERENCES

[1] O. Litany, E. Rodolà, A. M. Bronstein, and M. M. Bronstein, "Fully spectral partial shape matching," *Comput. Graph. Forum*, vol. 36, no. 2, pp. 247–258, May 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13123

[2] E. Rodolà, L. Cosmo, M. M. Bronstein, A. Torsello, and D. Cremers, "Partial functional correspondence," *Comput. Graph. Forum*, vol. 36, no. 1, pp. 222–236, Jan. 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12797

[3] M. Huska, D. Lazzaro, and S. Morigi, "Shape partitioning via L_p compressed modes," *J. Math. Imag. Vis.*, vol. 60, no. 7, pp. 1111–1131, Sep. 2018, doi: 10.1007/s10851-018-0799-8.

[4] A. Wetzler, Y. Aflalo, A. Dubrovina, and R. Kimmel, "The laplace-beltrami operator: A ubiquitous tool for image and shape processing," in *Mathematical Morphology and Its Applications to Signal and Image Processing*. Cham, Switzerland: Springer, 2013, pp. 302–316.

[5] M. Reuter, F.-E. Wolter, and N. Peinecke, "Laplace–Beltrami spectra as 'shape-DNA' of surfaces and solids," *Computer-Aided Design*, vol. 38, no. 4, pp. 342–366, Apr. 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010448505001867

[6] R. Song, Y. Liu, R. R. Martin, and P. L. Rosin, "Mesh saliency via spectral processing," *ACM Trans. Graph.*, vol. 33, no. 1, pp. 1–17, Feb. 2014, doi: 10.1145/2530691.

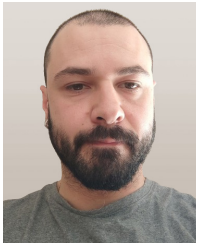
[7] G. Pai, A. Bronstein, R. Talmon, and R. Kimmel, "Deep isometric maps," *Image Vis. Comput.*, vol. 123, Jul. 2022, Art. no. 104461. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885622000907

[8] L. Cosmo, M. Panine, A. Rampini, M. Ovsjanikov, M. M. Bronstein, and R. Rodola, "Isospectralization, or how to hear shape, style, and correspondence," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7521–7530.

[9] J. K. Gahm and Y. Shi, "Riemannian metric optimization on surfaces (RMOS) for intrinsic brain mapping in the Laplace–Beltrami embedding space," *Med. Image Anal.*, vol. 46, pp. 189–201, May 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841518300847
[10] O. Yigit and R. C. Wilson, "LBONet: Supervised spectral descriptors for shape analysis," 2024, *arXiv:2411.08272*.
[11] 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," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999118307125
[12] D. Lazzaro, S. Morigi, and P. Zuzolo, "Learning intrinsic shape representations via spectral mesh convolutions," *Neurocomputing*, vol. 598, Sep. 2024, Art. no. 128152. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231224009238
[13] H. Zhang, O. Van Kaick, and R. Dyer, "Spectral mesh processing," *Comput. Graph. Forum*, vol. 29, no. 6, pp. 1865–1894, Sep. 2010.
[14] F. Zhu, S. Li, and G. Wang, "Example-based materials in Laplace–Beltrami shape space," *Comput. Graph. Forum*, vol. 34, no. 1, pp. 36–46, Feb. 2015.
[15] F. S. Costabal, S. Pezzuto, and P. Perdikaris, "Δ-PINNs: Physics-informed neural networks on complex geometries," *Eng. Appl. Artif. Intell.*, vol. 127, Jan. 2024, Art. no. 107324. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197623015087
[16] H. Gao, M. J. Zahr, and J.-X. Wang, "Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems," *Comput. Methods Appl. Mech. Eng.*, vol. 390, Feb. 2022, Art. no. 114502.
[17] T. De Ryck and S. Mishra, "Numerical analysis of physics-informed neural networks and related models in physics-informed machine learning," *Acta Numerica*, vol. 33, pp. 633–713, Jul. 2024.
[18] P.-H. Chiu, J. C. Wong, C. Ooi, M. H. Dao, and Y.-S. Ong, "CAN-PINN: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method," *Comput. Methods Appl. Mech. Eng.*, vol. 395, May 2022, Art. no. 114909. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782522001906
[19] I. Ben-Shaul, L. Bar, D. Fishelov, and N. Sochen, "Deep learning solution of the eigenvalue problem for differential operators," *Neural Comput.*, vol. 35, no. 6, pp. 1100–1134, May 2023.
[20] L. Sun, H. Gao, S. Pan, and J.-X. Wang, "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data," *Comput. Methods Appl. Mech. Eng.*, vol. 361, Apr. 2020, Art. no. 112732. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S004578251930622X
[21] R. Zhang, Y. Liu, and H. Sun, "Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling," *Eng. Struct.*, vol. 215, Jul. 2020, Art. no. 110704. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141029619345080
[22] J.-Z. Peng, N. Aubry, Y.-B. Li, M. Mei, Z.-H. Chen, and W.-T. Wu, "Physics-informed graph convolutional neural network for modeling geometry-adaptive steady-state natural convection," *Int. J. Heat Mass Transf.*, vol. 216, Dec. 2023, Art. no. 124593. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S001793102300738X
[23] I. Chavel, *Eigenvalues in Riemannian Geometry* (Pure and Applied Mathematics), vol. 115. New York, NY, USA: Academic, 1984.
[24] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, Dec. 2007, doi: 10.1007/s11222-007-9033-z.
[25] M. Reuter, S. Biasotti, D. Giorgi, G. Patanè, and M. Spagnuolo, "Discrete Laplace–Beltrami operators for shape analysis and segmentation," *Comput. Graph.*, vol. 33, no. 3, pp. 381–390, Jun. 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849309000272
[26] D. Burago, S. Ivanov, and Y. Kurylev, "A graph discretization of the Laplace–Beltrami operator," *J. Spectral Theory*, vol. 4, no. 4, pp. 675–714, Jan. 2015.
[27] P. Virtanen et al., "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020.
[28] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2011. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611970739
[29] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural. Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2008.
[30] L. Wu, P. Cui, J. Pei, L. Zhao, and X. Guo, "Graph neural networks: Foundation, frontiers and applications," in *Proc. 29th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2023, pp. 5831–5832, doi: 10.1145/3580305.3599560.
[31] Y. Shen, H. Fu, Z. Du, X. Chen, E. Burnaev, D. Zorin, K. Zhou, and Y. Zheng, "GCN-denoiser: Mesh denoising with graph convolutional networks," *ACM Trans. Graph.*, vol. 41, no. 1, pp. 1–14, Feb. 2022.
[32] K. Faghih Niresi, H. Bissig, H. Baumann, and O. Fink, "Physics-enhanced graph neural networks for soft sensing in industrial Internet of Things," *IEEE Internet Things J.*, vol. 11, no. 21, pp. 34978–34990, Nov. 2024, doi: 10.1109/JIOT.2024.3434732.
[33] T. Novello, D. Aldana, A. Araujo, and L. Velho, "Tuning the frequencies: Robust training for sinusoidal neural networks," 2024, *arXiv:2407.21121*.
[34] V. Sitzmann, J. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 7462–7473.
[35] G. B. Arfken, H. J. Weber, and F. E. Harris, "Chapter 9-partial differential equations," in *Mathematical Methods for Physicists*. New York, NY, USA: Academic, 2013, ch. 9, pp. 401–445.
[36] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn.*, vol. 4, 2017, pp. 2713–2726.
[37] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lió, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn.*, vol 4, 2018, pp. 2920–2931.
[38] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. 30th Int. Conf. Mach. Learn.*, 2012, pp. 1310–1318. [Online]. Available: https://proceedings.mlr.press/v28/pascanu13.html

DAMIANA LAZZARO received the B.S. degree (summa cum laude) in mathematics from the University of Messina, Italy, in 1988, and the Ph.D. degree in computational mathematics from the University of Naples Federico II, Italy, in 1995. She is currently an Associate Professor of numerical analysis with the Department of Mathematics, University of Bologna, Italy. Her major research interests include wavelet and multiwavelet theory and applications, signal and image processing, inverse problems, and compressed sensing. She is an Associate Editor of *International Journal of Computer Mathematics*.

SERENA MORIGI received the Ph.D. degree in applied mathematics from the University of Padova, Italy, in 1996. She is currently a Full Professor of numerical analysis with the Department of Mathematics, University of Bologna, Italy. In 1997, she was a Postdoctoral Fellow with Vanderbilt University, Nashville, TN, USA. She has coordinated/participated in several national and international research projects. Her research interests include discrete ill-posed problems and numerical methods for image processing, geometric modeling, surface reconstruction, and geometry processing. She is an Associate Editor of international journals in numerical analysis and image processing.



PAOLO ZUZOLO received the M.S. degree in mathematics from the University of Bologna, Bologna, Italy, in 2020, where he is currently pursuing the Ph.D. degree in mathematics. After completing his master's degree, he was a Researcher with the Visualization Information Technology Laboratory, CINECA, Bologna. His research interests include numerical methods for inverse problems in image and geometry processing and graph neural networks.