

SpectONet: A Physics-Guided Spectral Deep Operator Network for Euler-Bernoulli Beam Dynamics

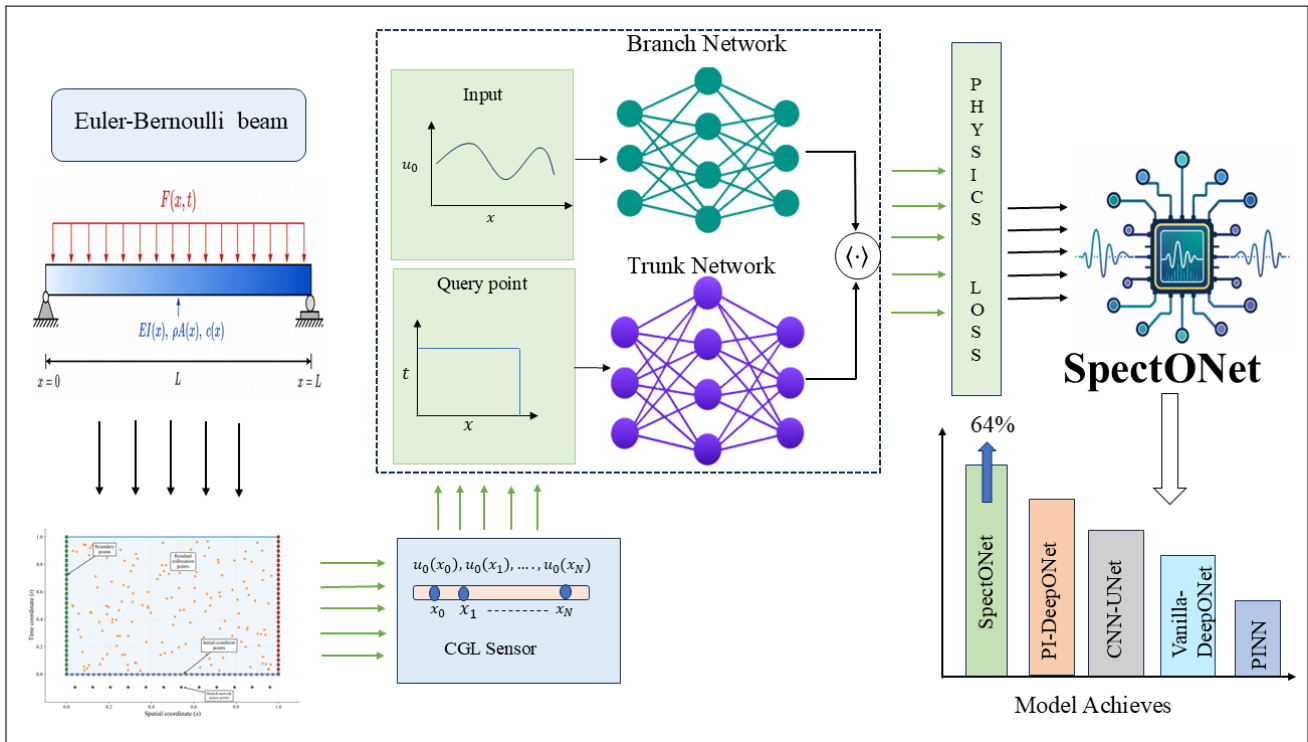
Shivani Saini^a, Ramesh Kumar Vats^a, Arup Kumar Sahoo^{*b}

^aDepartment of Mathematics and Scientific Computing,
National Institute of Technology Hamirpur, 177005, Himachal Pradesh, India

^bThe Hatter Department of Marine Technologies, Leon H. Charney School of Marine Sciences, University of Haifa, 3498838, Haifa, Israel

Abstract

This paper proposes a novel physics-guided spectral deep operator network, termed SpectONet, for solving Euler-Bernoulli beam (EBB) vibration problems. The proposed framework integrates the operator-learning capability of DeepONet with physics-informed constraints and Chebyshev-Gauss-Lobatto (CGL) sensor placement. Unlike conventional DeepONet frameworks, which commonly employ uniformly distributed sensors, SpectONet uses nonuniform spectral sensor locations with a higher concentration of points near the domain boundaries. This sampling strategy improves the finite-dimensional representation of boundary-sensitive structural responses while requiring only a limited number of branch-network inputs. The governing beam equation, together with the associated initial and boundary conditions, incorporated into the training objective to promote physically consistent and generalizable predictions. Numerical experiments on three synthetic EBB vibration problems and a real-world bridge vibration dataset demonstrate the effectiveness of the proposed framework. Comparisons with strong baselines such as, Vanilla DeepONet, PI-DeepONet, PINN, and CNN-UNet show that SpectONet consistently achieves lower prediction errors across all considered evaluation metrics. In particular, SpectONet achieves at least 64% improvement over the considered baseline models across the three synthetic problems and at least 37% for the real-world problems. These results demonstrate that SpectONet provides an accurate, computationally efficient, and physically consistent operator-learning framework for structural vibration analysis.



Graphical abstract

Keywords: Scientific Machine Learning; Neural Operator; Physics-Guided DeepONet; Chebyshev-Gauss-Lobatto Sensors; Euler-Bernoulli Beam; Structural Vibration.

*Corresponding author

Email addresses: shivanis.nith@gmail.com (Shivani Saini), rkstats@nith.ac.in (Ramesh Kumar Vats), arupnitr.jrfmath@gmail.com (Arup Kumar Sahoo*)

1. Introduction

Artificial intelligence (AI) has emerged as one of the fastest-growing computational technologies in 21st century, with successful applications in pattern recognition, natural language processing, computer vision, robotics, and scientific computing [1, 2]. Among the various AI approaches, artificial neural networks (ANNs) have gained considerable attention because of their ability to learn complex nonlinear relationships directly from available data. Owing to their strong approximation and generalization capabilities, ANNs have been increasingly employed for modelling dynamical problems governed by differential, integral, and polynomial equations [3, 4, 5].

A wide variety of real-world phenomena arising in structural mechanics, fluid dynamics, heat transfer, electrostatics, wave propagation, and mathematical physics can be described using ordinary differential equations (ODEs) and partial differential equations (PDEs). In structural mechanics, such equations are extensively employed to model the static and dynamic behaviour of advanced engineering structures, including multidirectional functionally graded nanostructures [6], composite beams [7], and functionally graded plates and shells [8]. The increasing complexity of these systems, together with variations in material properties, external loading, geometrical configurations, initial conditions (ICs), and boundary conditions (BCs), motivates the development of accurate, efficient, and flexible computational frameworks. In this context, ANN-based approaches provide a promising means of learning the underlying solution behaviour and establishing nonlinear mappings between physical inputs and the corresponding system responses.

One of the earliest applications of ANNs to solve dynamical problems was presented by Lagaris et al. [9]. They developed a feedforward neural network framework for approximating the solutions of ODEs and PDEs while satisfying the prescribed ICs and BCs. Their work demonstrated the potential of ANNs as flexible and mesh-free computational tools for solving differential equations (DEs) problems. Since then, ANNs have been widely employed as surrogate models for complex nonlinear systems because they can provide rapid predictions after training [10]. However, conventional ANNs are primarily data-driven, typically require large labelled datasets, and may produce physically inconsistent solutions when the physical laws are not incorporated during model training.

To address these limitations, physics-informed neural networks (PINNs), first proposed by Raissi et al. [11], have emerged as a powerful mesh-free framework for solving forward and inverse DE problems. Unlike conventional data-driven neural networks, PINNs incorporate the DEs together with the associated ICs and BCs directly into the loss function. The required spatial and temporal derivatives are generally evaluated through automatic differentiation (AD), allowing the neural network to be trained without conventional mesh-based discretization. By embedding physical knowledge into the learning process, PINNs reduce their dependence on labelled data and encourage the predicted solutions to remain consistent with the underlying governing laws. Owing to these advantages, PINNs have been successfully applied to fluid dynamics [12, 13], wave propagation [14], structural vibration analysis [15], and solid mechanics [16].

Despite their significant advantages, conventional PINNs generally learn only a single solution corresponding to a fixed DE configuration rather than the underlying solution operator. Consequently, they cannot directly generalize to variations in governing parameter, forcing function, IC, or BC without retraining. This limitation restricts their computational efficiency and practical applicability in repeated-query and parameter-dependent problems.

Motivated by the need to learn entire families of solutions rather than individual solution instances, operator learning has recently emerged as a powerful paradigm for approximating mappings between infinite-dimensional function spaces. Unlike conventional learning approaches, which approximate the solution corresponding to a fixed set of governing parameters, IC, BC, and forcing functions, operator learning learns the mapping from input functions to the corresponding solution fields. Consequently, a single trained neural operator can directly predict solutions for previously unseen input functions, varying parameters, and different IC or BC without requiring retraining. Such frameworks are particularly attractive for many-query and real-time applications, where repeated evaluations of the solution operator are required. Owing to their remarkable approximation capabilities and near real-time inference, neural operators have attracted considerable attention in computational mechanics [17], and fluid dynamics [18]. These capabilities make neural operators particularly suitable for complex engineering systems that require rapid prediction, repeated evaluations, and real-time decision-making.

Among the various neural operator architectures developed in recent years, the deep op-

erator network (DeepONet), proposed by Lu et al. [19] and theoretically supported by the universal approximation theorem for nonlinear operators, has emerged as one of the most versatile frameworks. DeepONet consists of two subnetworks: a branch network that encodes the input function and a trunk network that represents the spatial and temporal coordinates. Through the interaction of these two subnetworks, DeepONet is capable of learning nonlinear operators associated with complex PDE systems. Several variants and extensions of neural operators have subsequently been proposed, including the Fourier neural operator [20], wavelet neural operator [21], and Laplace neural operator [22], convolution neural operator [23], and graph neural operator [24] each possessing distinct advantages depending on the application domain [25].

Despite their success, conventional DeepONet models generally require a large amount of paired input-output data generated from expensive numerical simulations or experimental measurements. The acquisition of such datasets becomes increasingly challenging and cost-effective for complex engineering systems. To alleviate this dependency on labeled data, Wang et al. [26] introduced the physics-informed deep operator network (PI-DeepONet), which embeds the governing equations and the associated ICs and BCs into the loss function. By incorporating physical constraints into the learning process, PI-DeepONet can achieve accurate predictions with significantly fewer training samples.

Nevertheless, several challenges remain unresolved in the existing PI-DeepONet frameworks. **First**, the accuracy of DeepONet strongly depends on the placement of sensor points used to discretize the input functions. Uniformly distributed sensors often fail to capture localized variations effectively, leading to reduced approximation accuracy and poor generalization. **Second**, conventional PI-DeepONet architectures treat sensor values independently and do not explicitly exploit the spectral characteristics of the underlying functional representation. **Third**, equidistant sensor locations generally require a large number of sampling points to accurately represent smooth solutions and forcing functions, thereby increasing the computational cost and training complexity.

In this regard, spectral approximation techniques based on orthogonal polynomials provide an alternative for function representation [27]. Among these techniques, Chebyshev-Gauss-Lobatto (CGL) points exhibit excellent interpolation properties, reducing interpolation errors near the domain boundaries while providing enhanced boundary resolution. Owing to these characteristics, CGL points have been widely employed in spectral methods because they offer high approximation accuracy with relatively few collocation points [28]. Despite their success in numerical analysis and spectral methods [29], the incorporation of CGL-based sensor placement into physics-informed operator learning frameworks has received limited attention. This observation motivates the development of a more effective operator learning framework that leverages the advantages of spectral sensor distributions while preserving the physical consistency provided by PI-DeepONet.

To fill these gaps, this work proposes a novel physics-guided spectral deep operator network (SpectONet) for solving Euler-Bernoulli Beam (EBB) vibration problems. The proposed framework combines the operator-learning capability of DeepONet with the physical constraints imposed by PI-DeepONet and employs CGL sensor locations to improve the discretization of input functions. The branch network receives the values of the forcing function or IC sampled at CGL points, whereas the trunk network encodes the spatio-temporal coordinates. Unlike conventional DeepONet models and baselines that rely on supervised solution data, the proposed algorithm is trained solely using the governing physics, BC, and IC, without requiring paired input-output solution fields. Despite the absence of supervised solution loss, the proposed framework achieves superior predictive accuracy through physics-constrained operator learning and orthogonal sensors.

The major contributions of this research are summarized as follows:

1. A novel SpectONet is proposed by integrating PI-DeepONet with optimized CGL sensor placement for solving EBB vibration problems.
2. Optimized non-uniform CGL sensor locations are incorporated into the branch network to improve the representation of input functions and prediction accuracy.
3. The proposed model eliminates the need for supervised solution data during training. Instead, learning is driven entirely by the governing PDE together with the associated IC and BC.
4. Extensive numerical experiments on three benchmark EBB vibration problems demonstrate that SpectONet consistently outperforms the baseline models, namely PI-DeepONet,

Vanilla-DeepONet, PINN, and CNN-UNet, in terms of prediction accuracy and robustness.

5. Finally, the model has been tested with a real-world sensor dataset collected from Z24 bridge in Switzerland, to validate our model.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents the preliminaries, including the mathematical formulation of the EBB vibration problems and the fundamentals of PINN and DeepONet. Section 4 describes the proposed SpectONet framework. Section 5 presents the numerical experiments, followed by a detailed discussion in Section 6. Section 7 validates the proposed framework using a real-world sensor dataset. Finally, Section 8 concludes the paper and outlines future research directions. For interested readers, the training algorithms of the baseline models considered in this study are provided in [Appendix A](#).

2. Related Work

This section reviews prior research on DeepONet, with particular emphasis on recent methodological advancements in PI-DeepONet and its applications to structural dynamics and vibration analysis.

In a pioneering work, Goswami et al. [30] highlighted the potential of physics-informed operator learning by integrating governing physical laws into neural operator architectures, thereby improving predictive accuracy and generalization for forward and inverse PDE problems. Building upon this foundation, several studies have proposed enhanced neural operator frameworks to improve robustness, scalability, computational efficiency, and predictive performance. Mandl et al. [31] proposed separable PI-DeepONet (Sep-PI-DeepONet), a computationally efficient extension of PI-DeepONet for solving high-dimensional PDEs. The framework employed separable network architectures and forward-mode AD to alleviate the curse of dimensionality and significantly reduce computational costs compared with the PI-DeepONet framework. Cong et al. [32] introduced a causality-aware segmental training strategy for PI-DeepONet, which decomposes the temporal domain and adaptively weights residual losses. The proposed approach substantially accelerated convergence and enhanced predictive accuracy for time-dependent PDEs. Similarly, Michałowska et al. [33] combined DeepONet with recurrent neural networks to enhance the accuracy and stability of long-term simulations of dynamical systems. Their two-stage framework employs DeepONet for operator learning and an recurrent neural network with a moving-window strategy to capture temporal dependencies and enhance long-term prediction accuracy. Yang et al. [34] introduced DD-DeepONet, a domain decomposition framework that learns subdomain-wise operators for efficient PDE solution on complex geometries. The proposed method reduced training difficulty and computational cost while enabling rapid prediction under varying geometrical and physical conditions. Karumuri et al. [35] developed physics-informed latent neural operator (PI-Latent-NO), which integrates latent-space learning with physics-based constraints for time-dependent parametric PDEs. The framework employed coupled DeepONet architectures to achieve accurate, scalable, and data-efficient predictions while significantly reducing computational costs. More recently Sarthak and Srinivasacharya [36] proposed a PI-DeepONet framework for long-time simulation of nonlinear PDEs using temporal domain decomposition. By recursively applying the learned operator over successive time subdomains and employing Gaussian random field sampling, the proposed approach improved long-time prediction stability and accelerated convergence. For further literature, Table 1 summarizes recent neural operator variants, their key innovations, and application domains.

In addition to these advancements, recent studies have increasingly explored the application of DeepONet and physics-informed operator learning to engineering problems, particularly in structural mechanics and vibration analysis. Lu et al. [42] employed a deep neural operator framework to predict the transient response of interpenetrating phase composites subjected to dynamic loading, demonstrating the effectiveness of operator learning for structural mechanics problems. More recently, research has increasingly focused on structural dynamics and vibration analysis, motivated by the growing demand for accurate and computationally efficient surrogate models. Van Delden et al. [43] proposed an operator learning-based framework for structural vibration prediction in harmonically excited plate structures. The proposed model effectively captures vibration responses across varying geometries and excitation conditions, outperforming DeepONet and Fourier neural operator on a large-scale vibration benchmark. Ahmed et al. [44] proposed a PI-DeepONet framework with stiffness-based loss functions for real-time structural response prediction. By incorporating equilibrium and energy conservation principles through structural stiffness matrices, the proposed

Table 1. Representative neural operator variants and their key features.

Variant	Application Area	Features
VS-WNO [37]	Computational mechanics	Integrates variable spiking neurons into a wavelet neural operator to reduce computational energy while maintaining accurate PDE operator learning.
VB-DeepONet [38]	Computational mechanics	Diffusion-reaction, gravity pendulum, and advection-diffusion problems.
DNO [39]	Variable geometries	Employs smooth diffeomorphic mappings to transform variable domains into a common reference domain before neural operator learning.
Geom-DeepONet [40]	Parameterized 3D geometries	Combines point-cloud geometry encoding, signed distance functions, sinusoidal representation network, and DeepONet-based feature fusion for full-field prediction.
Phase-Field DeepONet [41]	Phase-field dynamics	Combines PI-DeepONet, minimizing movement schemes, and energy-based optimization to learn explicit time-stepping operators governed by free-energy functionals.

approach achieved accurate displacement and rotation predictions with significantly reduced training time. The study demonstrated the potential of physics-informed operator learning for efficient structural analysis and surrogate modeling. He et al. [45] extended the DeepONet architecture by proposing a sequential DeepONet (S-DeepONet) for predicting transient vector-valued solution fields. The proposed framework simultaneously predicted multiple output components over time and demonstrated high accuracy and computational efficiency for transient fluid flow and path-dependent plasticity problems. Liu et al. [46] presented a preliminary study on vibration analysis using DeepONet for a single-degree-of-freedom (SDOF) vibration system. The authors demonstrated that DeepONet can effectively learn and predict structural vibration responses under various loading conditions. Their results highlighted the potential of operator learning frameworks for structural health monitoring (SHM) and dynamic system analysis. Similarly, Wan et al. [47] introduced DeepVIVONet, a DeepONet-based framework for dynamic reconstruction and forecasting of vortex-induced vibrations using sparse spatio-temporal measurements. The proposed model demonstrated accurate vibration prediction, transfer-learning capabilities, and effective sensor placement optimization for marine riser systems. Tian et al. [48] proposed a PI-DeepONet framework for the bending analysis of stiffened panels by combining mixed-variable formulations with physics-informed operator learning. The proposed approach achieved real-time full-field response prediction under varying loading conditions and demonstrated high accuracy for complex thin-walled structural systems.

Overall, the literature studies demonstrate the growing capability of neural operator frameworks for scientific computing and engineering applications. Building upon these advances, the next section introduces the benchmark EBB vibration problems considered in this work.

3. Preliminaries

This section provides the mathematical and theoretical background necessary for the proposed methodology. It first presents the problem statement, followed by an overview of the fundamental principles of PINNs and DeepONet.

3.1. Problems Statement

The transverse vibration of an Euler-Bernoulli beam (EBB) can be written as [49]

$$\rho(x) \frac{\partial^2 u(x, t)}{\partial t^2} + c(x) \frac{\partial u(x, t)}{\partial t} + \frac{\partial^2}{\partial x^2} \left[EI(x) \frac{\partial^2 u(x, t)}{\partial x^2} \right] = F(x, t), \quad x \in \Omega, \quad t \in [0, T], \quad (1)$$

where $u(x, t)$ denotes the transverse displacement of the beam, $\rho(x)$ is the mass density per unit length, $c(x)$ is the damping coefficient, $EI(x)$ is the spatially varying flexural rigidity, and $F(x, t)$ denotes the external distributed load. For variable flexural rigidity, the operator $\frac{\partial^2}{\partial x^2} \left(EI(x) \frac{\partial^2 u}{\partial x^2} \right)$ naturally accounts for the additional spatial derivative terms arising from the variation of $EI(x)$.

To obtain an unique solution, the following ICs are prescribed:

$$u(x, 0) = u_0(x), \quad x \in \Omega, \quad (2)$$

$$\frac{\partial u}{\partial t}(x, 0) = v_0(x), \quad x \in \Omega, \quad (3)$$

where $u_0(x)$ and $v_0(x)$ represent the initial displacement and velocity distributions, respectively. The BCs of the considered EBB problems depend on the beam configuration. In general form, they can be expressed as:

$$\mathcal{B}_x[u(x, t)] = h(x, t), \quad x \in \partial\Omega, \quad t \in [0, T], \quad (4)$$

where $\mathcal{B}_x[\cdot]$ denotes the differential boundary operator and $h(x, t)$ represents the specified boundary data.

Eq. (1) provides a unified framework for describing a wide range of structural vibration phenomena encountered in EBB dynamics. By appropriately specifying the external loading, BCs, and material parameters, several classical beam vibration models can be obtained as special cases. In the present study, three representative classes of EBB vibration problems are considered, namely, the forced undamped beam, the undamped simply supported beam, and the damped beam. Their corresponding physical configurations are illustrated in Figure 1.

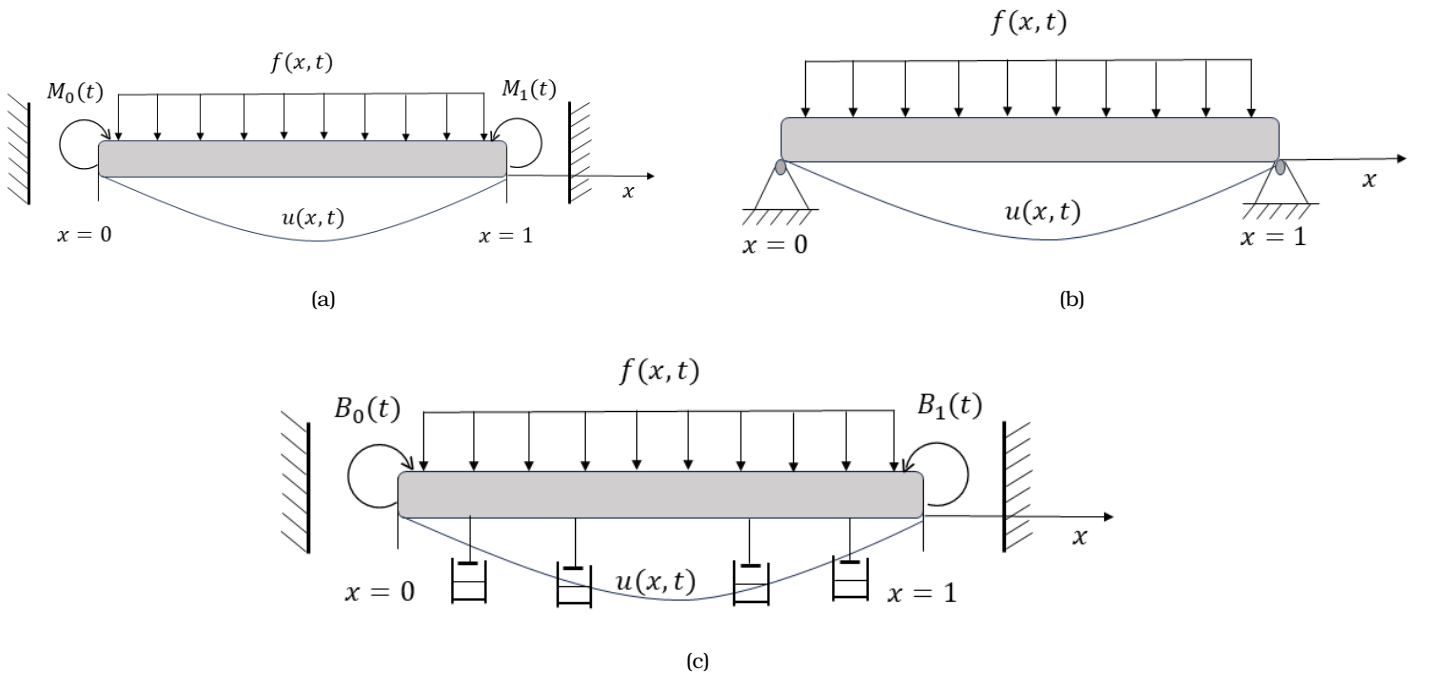


Figure 1. Schematic illustrations of the three representative EBB vibration problems investigated in this work, (a) forced undamped beam, (b) undamped simply supported beam, and (c) damped variable-coefficient beam.

(a) The forced undamped vibration of an EBB, illustrated in Figure 1a, is governed by the following fourth-order PDE [50]:

$$\rho(x) \frac{\partial^2 u(x, t)}{\partial t^2} + \frac{\partial^2}{\partial x^2} \left[EI(x) \frac{\partial^2 u(x, t)}{\partial x^2} \right] = F(x, t), \quad x \in \Omega, \quad t \in [0, T], \quad (5)$$

The corresponding ICs and BCs are

$$u(x, 0) = u_0(x), \quad \frac{\partial u}{\partial t}(x, 0) = v_0(x), \quad x \in \Omega, \quad (6)$$

$$u(0, t) = 0, \quad u(1, t) = 0, \quad (7)$$

$$\frac{\partial^2 u}{\partial x^2}(0, t) = M_0(t), \quad \frac{\partial^2 u}{\partial x^2}(1, t) = M_1(t), \quad (8)$$

where $M_0(t)$ and $M_1(t)$ represent the bending moments at left and right ends of the beam, respectively.

- (b) The transverse undamped vibration of a simply supported EBB, illustrated in Figure 1b, is written as [50]:

$$\rho(x) \frac{\partial^2 u(x, t)}{\partial t^2} + \frac{\partial^2}{\partial x^2} \left[EI(x) \frac{\partial^2 u(x, t)}{\partial x^2} \right] = F(x, t), \quad x \in \Omega, \quad t \in [0, T]. \quad (9)$$

The ICs and simply supported BCs are expressed as

$$u(x, 0) = u_0(x), \quad \frac{\partial u}{\partial t}(x, 0) = v_0(x), \quad x \in \Omega, \quad (10)$$

$$u(0, t) = 0, \quad u(1, t) = 0, \quad (11)$$

$$\frac{\partial^2 u}{\partial x^2}(0, t) = 0, \quad \frac{\partial^2 u}{\partial x^2}(1, t) = 0, \quad t \in [0, T]. \quad (12)$$

Where the beam is simply supported at both ends, so that the transverse displacement and bending moments are zero at $x = 0$ and $x = 1$.

- (c) The transverse damped vibration of a variable-coefficient EBB, illustrated in Figure 1c, is governed by [7]:

$$\rho(x) \frac{\partial^2 u(x, t)}{\partial t^2} + c(x) \frac{\partial u(x, t)}{\partial t} + \frac{\partial^2}{\partial x^2} \left[EI(x) \frac{\partial^2 u(x, t)}{\partial x^2} \right] = F(x, t), \quad x \in \Omega, \quad t \in [0, T]. \quad (13)$$

The corresponding ICs and BCs are

$$u(x, 0) = u_0(x), \quad \frac{\partial u}{\partial t}(x, 0) = v_0(x), \quad x \in \Omega, \quad (14)$$

$$u(0, t) = 0, \quad u(1, t) = 0, \quad (15)$$

$$\frac{\partial^2 u}{\partial x^2}(0, t) = B_0(t), \quad \frac{\partial^2 u}{\partial x^2}(1, t) = B_1(t), \quad (16)$$

where $B_0(t)$ and $B_1(t)$ denote time-dependent boundary moment functions.

3.2. Physics-informed Neural Networks

PINNs introduced by Raissi et al. [11], are a class of deep learning models for solving forward and inverse problems governed by PDEs. Unlike conventional data-driven neural networks, PINNs incorporate the governing PDE together with the associated IC and BC into the loss function, thereby embedding physical laws directly into the training process. The resulting loss function consists of a data loss, which enforces the prescribed IC and BC, and a physics loss, which minimizes the residual of the governing PDE [51]. To formulate the PINN framework, consider the following general PDE defined over the spatial domain Ω and temporal domain $\mathcal{T}$:

$$\mathcal{N}(u(\mathbf{x}, t; \boldsymbol{\lambda})) = f(\mathbf{x}, t), \quad (\mathbf{x}, t) \in \Omega \times \mathcal{T}, \quad (17)$$

where $\mathcal{N}(\cdot)$ denotes a nonlinear differential operator, $u(\mathbf{x}, t)$ is the unknown solution, $\boldsymbol{\lambda}$ represents the physical parameters of the system, and $f(\mathbf{x}, t)$ denotes the source term. To ensure the uniqueness of the solution, Eq. (17) is supplemented with appropriate IC and BC,

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}), \quad \mathbf{x} \in \Omega, \quad (18)$$

$$\mathcal{B}(u) = h(\mathbf{x}, t), \quad (\mathbf{x}, t) \in \partial\Omega \times \mathcal{T}. \quad (19)$$

where $\mathcal{B}(\cdot)$ denotes the boundary operator, while $u_0(\mathbf{x})$ and $h(\mathbf{x}, t)$ represent the prescribed IC and BC, respectively. The unknown solution is approximated by a deep neural network $u_\theta(\mathbf{x}, t)$, where θ denotes the trainable network parameters. Using AD, the residual of the governing PDE is defined as

$$R(\mathbf{x}, t; \theta) = \mathcal{N}(u_\theta(\mathbf{x}, t; \boldsymbol{\lambda})) - f(\mathbf{x}, t). \quad (20)$$

The trainable parameters are obtained through the minimization of the following composite loss function,

$$\mathcal{L}(\theta) = \lambda_r \mathcal{L}_{\text{PDE}} + \lambda_i \mathcal{L}_{\text{IC}} + \lambda_b \mathcal{L}_{\text{BC}}, \quad (21)$$

where λ_r , λ_i , and λ_b denote the weighting coefficients associated with PDE residual, ICs, and BCs losses, respectively. The individual loss terms are given by

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_r} \sum_{i=1}^{N_r} |R(\mathbf{x}_i^r, t_i^r; \theta)|^2, \quad (22)$$

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_0} \sum_{i=1}^{N_0} |u_\theta(\mathbf{x}_i^0, 0) - u_0(\mathbf{x}_i^0)|^2, \quad (23)$$

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_b} \sum_{i=1}^{N_b} |\mathcal{B}(u_\theta(\mathbf{x}_i^b, t_i^b)) - h(\mathbf{x}_i^b, t_i^b)|^2, \quad (24)$$

where N_r , N_i , and N_b denote the numbers of interior, initial, and boundary collocation points, respectively. The optimal network parameters are obtained by solving

$$\theta^* = \arg \min_{\theta} \mathcal{L}(\theta). \quad (25)$$

Although PINNs have demonstrated remarkable success in solving a broad range of PDEs, they remain solution-specific, requiring retraining whenever the governing equation, physical parameters, forcing function, or IC and BC change. Furthermore, the computational cost can become significant for complex and many-query problems. These limitations have motivated the development of operator learning approaches, such as DeepONet, which learn mappings between function spaces rather than individual PDE solutions.

3.3. Deep Operator Networks

DeepONet, proposed by Lu et al. [19], are a class of neural operator architectures designed to approximate learn nonlinear mappings between infinite-dimensional function spaces. Unlike conventional neural networks, which approximate mappings between finite-dimensional vectors, DeepONet directly approximates nonlinear operators. The input function may represent quantities such as an IC, BC, source term, material property, or domain. Owing to the universal approximation theorem (Theorem 1) for operators, DeepONet has emerged as an effective framework for solving PDEs, surrogate modeling, inverse problems, and various SciML applications. For the PDE defined in Eq. (17), the objective of DeepONet is to learn the nonlinear solution operator

$$\mathcal{G} : u_0(\mathbf{x}) \mapsto u(\mathbf{x}, t),$$

where $u_0(\mathbf{x})$ denotes the IC given in Eq. (18), and $u(\mathbf{x}, t)$ represents the corresponding solution. The trained DeepONet approximates the solution operator as $\hat{\mathcal{G}}_\theta(u_0)(\mathbf{x}, t)$, where $\theta = (W, b)$ denotes the trainable weights and biases. Its branch network encodes the sampled input function, while the trunk network encodes the spatial-temporal coordinates.

Since the branch network operates on finite-dimensional inputs, the continuous input function is first discretized by sampling it at a finite set of prescribed sensor locations, denoted by $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$. The resulting discretized input is represented as

$$u_0 = [u_0(\mathbf{x}_1), u_0(\mathbf{x}_2), \dots, u_0(\mathbf{x}_m)]^T, \quad (26)$$

which serves as the input feature vector for the branch network. The trunk network receives the spatial-temporal coordinates $(\mathbf{x}, t) \in \Omega \times \mathcal{T}$. The branch and trunk networks generate the latent feature vectors $\mathbf{b} = [\hat{b}_1, \hat{b}_2, \dots, \hat{b}_p]^T$, and $\mathbf{t} = [\hat{t}_1, \hat{t}_2, \dots, \hat{t}_p]^T$, respectively, where both vectors

have the same latent dimension p . Since both feature vectors have the same dimension, the DeepONet approximation is obtained through their inner product,

$$\widehat{\mathcal{G}}_{\theta}(u_0)(\mathbf{x}, t) = \sum_{k=1}^p \widehat{b}_k(u_0) \widehat{t}_k(\mathbf{x}, t), \quad (27)$$

where $\widehat{b}_k(\mathbf{g})$ denotes the latent feature extracted by the branch network from the discretized input function, whereas $\widehat{t}_k(\mathbf{x}, t)$ represents the basis function generated by the trunk network corresponding to spatial-temporal coordinates.

Theorem 1 (Universal approximation theorem for operators [19]). *Let σ be a continuous non-polynomial activation function. Let X be a Banach space, and let $K_1 \subset X$ and $K_2 \subset \mathbb{R}^d$ be compact sets. Assume that V is a compact subset of $C(K_1)$ and that $\mathcal{G} : V \rightarrow C(K_2)$ is a continuous nonlinear operator. Then, for every $\varepsilon > 0$, there exist positive integers m , n , and p , together with appropriate constants, such that*

$$\left| \mathcal{G}(u)(y) - \sum_{k=1}^p \left(\sum_{i=1}^n c_i^k \sigma \left(\sum_{j=1}^m \xi_{ij}^k u(x_j) + \theta_i^k \right) \right) \sigma(w_k \cdot y + \zeta_k) \right| < \varepsilon,$$

for all $u \in V$ and $y \in K_2$.

The above theorem establishes the mathematical foundation for DeepONet by demonstrating its ability to approximate a broad class of continuous nonlinear operators with arbitrary accuracy. Furthermore, the theorem remains valid when the Banach spaces $C(K_1)$ and $C(K_2)$ are replaced by $L^q(K_1)$ and $L^r(K_2)$, respectively, with $q, r \geq 1$.

The trainable parameters of DeepONet are obtained by minimizing the mean squared error (MSE) loss. Let $u^{(i)}(\mathbf{x}, t)$ denote the exact solution corresponding to the input function $u_0^{(i)}(\mathbf{x})$. The loss function is defined as

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N w_i \left| u^{(i)}(\mathbf{x}, t) - \widehat{\mathcal{G}}_{\theta}(u_0^{(i)})(\mathbf{x}, t) \right|^2, \quad (28)$$

where N denotes the total number of training samples and w_i is the weighting coefficient associated with the i^{th} training sample.

Although DeepONet is capable of accurately learning nonlinear operators, its original formulation does not explicitly incorporate the PDE and associated physical constraints into the learning process. Consequently, its performance may deteriorate when training data are limited or when strong physical consistency is required. These limitations motivate the development of physics-guided operator learning frameworks that embed governing equations into the optimization process, leading to improved accuracy, enhanced data efficiency, and physically consistent predictions.

4. Proposed Model

The SpectONet architecture shown in Figure 3 consists of two interconnected multilayer perceptron (MLP): a branch network and a trunk network. For the generalized EBB vibration model given in Eq. (1), the objective is to learn the solution operator

$$\mathcal{G} : u_0 \mapsto u, \quad \mathcal{G}(u_0)(x, t) = u(x, t), \quad (29)$$

where $u_0(x)$ denotes the initial displacement function, and $u(x, t)$ is the corresponding spatiotemporal displacement field. In SpectONet, the branch network encodes $u_0(x)$ through its CGL sensor representation, while the trunk network encodes the spatio-temporal coordinates (x, t) at which the beam displacement is evaluated. The learned representations from the branch and trunk networks are combined using an inner-product operation to obtain the final neural network prediction.

4.1. Branch Network with CGL Sensor Representation

The branch network is responsible for learning the dependence of the solution on the input function $u_0(x)$. Since $u_0(x)$ is a continuous function, it cannot be directly supplied to a neural network. Therefore, it is first converted into a finite-dimensional vector by evaluating it at a set of CGL sensor locations. The CGL nodes are derived from the extrema of the Chebyshev polynomials of the first kind, defined as [52]

$$T_N(\xi) = \cos(N \arccos \xi), \quad \xi \in [-1, 1]. \quad (30)$$

The extrema of $T_N(\xi)$, or equivalently the roots of

$$(1 - \xi^2)T'_N(\xi) = 0, \quad (31)$$

define the CGL nodes on the reference interval $[-1, 1]$ as

$$\xi_j = -\cos\left(\frac{j\pi}{N}\right), \quad j = 0, 1, \dots, N. \quad (32)$$

Since the EBB problem is considered over the physical domain $\Omega = [0, 1]$, the reference CGL nodes are mapped to the physical domain as

$$x_j = \frac{\xi_j + 1}{2} = \frac{1}{2} \left[1 - \cos\left(\frac{j\pi}{N}\right) \right], \quad j = 0, 1, \dots, N. \quad (33)$$

Using these CGL sensor locations, the continuous input function $u_0(x)$ is represented by the sensor vector

$$\mathbf{u}_0 = [u_0(x_0), u_0(x_1), \dots, u_0(x_N)]^T \in \mathbb{R}^{N+1}. \quad (34)$$

This CGL-based sensor vector is supplied to the branch network as the finite-dimensional representation of the input function.

Unlike uniformly distributed sensors, the CGL sensors naturally cluster near both boundaries of the computational domain and remain relatively sparse in the interior. This boundary clustering can be observed by setting $\theta_j = j\pi/N$. For $j \ll N$, using the approximation $\cos \theta = 1 - \theta^2/2 + \mathcal{O}(\theta^4)$, the physical CGL node satisfies

$$x_j = \frac{1 - \cos \theta_j}{2} \approx \frac{\pi^2 j^2}{4N^2} = \mathcal{O}\left(\frac{j^2}{N^2}\right), \quad j \ll N. \quad (35)$$

Consequently, the spacing between consecutive sensors behaves as

$$\Delta x_j = x_{j+1} - x_j \approx \frac{\pi^2}{4N^2}(2j+1) = \mathcal{O}\left(\frac{j}{N^2}\right), \quad j \ll N. \quad (36)$$

Thus, the CGL sensor spacing becomes very small near the boundaries, providing higher spatial resolution in boundary-dominated regions of EBB vibration problems. The resulting CGL-based sensor vector $\mathbf{u}_0$ is then supplied to the branch network, which maps this finite-dimensional representation into the latent operator space through a MLP [19]. The hidden layers of the branch network are defined as

$$\mathbf{h}_b^{(l)} = \phi_b \left(\mathbf{W}_b^{(l)} \mathbf{h}_b^{(l-1)} + \mathbf{c}_b^{(l)} \right), \quad l = 1, 2, \dots, L_b, \quad (37)$$

where

$$\mathbf{h}_b^{(0)} = [u_0(x_0), u_0(x_1), \dots, u_0(x_N)]^T. \quad (38)$$

Here, $\mathbf{W}_b^{(l)}$ and $\mathbf{c}_b^{(l)}$ denote the trainable weight matrix and bias vector of the l -th branch layer, respectively, and $\phi_b(\cdot)$ is the nonlinear activation function used in the branch network. The output of the final hidden layer is projected to the branch output vector

$$\mathcal{B}_{\theta_b}(\mathbf{u}_0) = [\hat{b}_1(\mathbf{u}_0), \hat{b}_2(\mathbf{u}_0), \dots, \hat{b}_q(\mathbf{u}_0)]^T \in \mathbb{R}^q. \quad (39)$$

Here, $\mathcal{B}_{\theta_b}$ is the branch network parameterized by θ_b , q denotes the output dimension of the branch network, and each component $\hat{b}_k(\mathbf{u}_0)$, $k = 1, 2, \dots, q$, is a learned scalar feature obtained from the complete CGL sensor vector $\mathbf{u}_0 \in \mathbb{R}^{N+1}$. Thus, the branch network converts the CGL sensor representation of the input function into a learned feature vector.

4.2. Trunk Network

The trunk network learns the dependence of the solution on the spatio-temporal coordinates. It takes the coordinate pair (x, t) as input and maps it into the same q -dimensional output space as the branch network. Let $\mathcal{T}_{\theta_t}$ denote the trunk network with trainable parameters θ_t . For an L_t -layer fully connected trunk network, the hidden layers are defined as

$$\mathbf{h}_t^{(l)} = \phi_t \left(\mathbf{W}_t^{(l)} \mathbf{h}_t^{(l-1)} + \mathbf{c}_t^{(l)} \right), \quad l = 1, 2, \dots, L_t, \quad (40)$$

where the input layer is given by

$$\mathbf{h}_t^{(0)} = [x, t]^T. \quad (41)$$

Here, $\mathbf{W}_t^{(l)}$ and $\mathbf{c}_t^{(l)}$ denote the trainable weight matrix and bias vector of the l -th trunk layer, respectively, and $\phi_t(\cdot)$ denotes the nonlinear activation function used in the trunk network. The output of the final hidden layer is projected to the trunk output vector

$$\mathcal{T}_{\theta_t}(x, t) = [\hat{t}_1(x, t), \hat{t}_2(x, t), \dots, \hat{t}_q(x, t)]^T \in \mathbb{R}^q. \quad (42)$$

Here, q denotes the output dimension of the trunk network, and each component $\hat{t}_k(x, t)$, $k = 1, 2, \dots, q$, is a learned coordinate-dependent feature associated with the spatio-temporal input (x, t) .

After obtaining the branch and trunk output vectors, SpectONet combines them through an inner-product operation. The branch network provides the learned features associated with the input function, while the trunk network provides the learned coordinate-dependent features. The learned operator $\hat{\mathcal{G}}_{\theta}$ approximates the exact solution operator $\mathcal{G}$, and its output is expressed as

$$\hat{\mathcal{G}}_{\theta}(u_0)(x, t) = \mathcal{B}_{\theta_b}(\mathbf{u}_0) \cdot \mathcal{T}_{\theta_t}(x, t) = \sum_{k=1}^q \hat{b}_k(\mathbf{u}_0) \hat{t}_k(x, t). \quad (43)$$

4.3. Physics-Informed Residual and Loss Function

To enforce the governing physics, automatic differentiation (AD) [53] is employed to compute the required temporal and spatial derivatives of the predicted operator output $\hat{\mathcal{G}}_{\theta}(u_0^{(i)})(x, t)$ [26]. During training, the branch network receives a collection of sampled initial displacement functions $\{u_0^{(i)}\}_{i=1}^{N_f}$, where N_f denotes the number of training input functions used to learn the solution operator. For the i -th sampled initial displacement function, the network prediction is given by

$$\hat{\mathcal{G}}_{\theta}(u_0^{(i)})(x, t) \approx \mathcal{G}(u_0^{(i)})(x, t) = u^{(i)}(x, t),$$

where $u^{(i)}(x, t)$ denotes the exact solution corresponding to the sampled initial displacement function $u_0^{(i)}(x)$. For each sampled input function, the governing equation is enforced at interior collocation points, while the corresponding ICs and BCs are imposed at the respective training points. Accordingly, the physics-informed residual corresponding to Eq. (1) is defined as

$$\begin{aligned} R_{\theta}(u_0^{(i)})(x, t) = & \rho(x) \frac{\partial^2 \hat{\mathcal{G}}_{\theta}(u_0^{(i)})(x, t)}{\partial t^2} + c(x) \frac{\partial \hat{\mathcal{G}}_{\theta}(u_0^{(i)})(x, t)}{\partial t} \\ & + \frac{\partial^2}{\partial x^2} \left[EI(x) \frac{\partial^2 \hat{\mathcal{G}}_{\theta}(u_0^{(i)})(x, t)}{\partial x^2} \right] - F(x, t), \end{aligned} \quad (44)$$

where R_{θ} measures the extent to which the predicted solution satisfies the governing EBB equation. The corresponding loss functions are given by

$$\mathcal{L}_{PDE} = \frac{1}{N_f N_r} \sum_{i=1}^{N_f} \sum_{j=1}^{N_r} \left| R_{\theta}(u_0^{(i)})(x_j^r, t_j^r) \right|^2, \quad (45)$$

$$\mathcal{L}_{BC} = \frac{1}{N_f N_b} \sum_{i=1}^{N_f} \sum_{j=1}^{N_b} \left| \mathcal{B}_x[\hat{\mathcal{G}}_{\theta}(u_0^{(i)})(x_j^b, t_j^b)] - h(x_j^b, t_j^b) \right|^2, \quad (46)$$

$$\begin{aligned} \mathcal{L}_{IC} = & \frac{1}{N_f N_0} \sum_{i=1}^{N_f} \sum_{j=1}^{N_0} \left| \hat{\mathcal{G}}_{\theta}(u_0^{(i)})(x_j^0, 0) - u_0^{(k)}(x_j^0) \right|^2 \\ & + \frac{1}{N_f N_0} \sum_{i=1}^{N_f} \sum_{j=1}^{N_0} \left| \frac{\partial}{\partial t} \hat{\mathcal{G}}_{\theta}(u_0^{(i)})(x_j^0, 0) - v_0^{(i)}(x_j^0) \right|^2, \end{aligned} \quad (47)$$

where, $\mathcal{L}_{PDE}$, $\mathcal{L}_{BC}$, and $\mathcal{L}_{IC}$ denote the PDE residual, BC, and ICs losses, respectively. Moreover, N_f denotes the number of sampled input functions, while N_r , N_b , and N_0 represent the numbers of interior collocation points, boundary points, and initial points associated with each sampled input function.

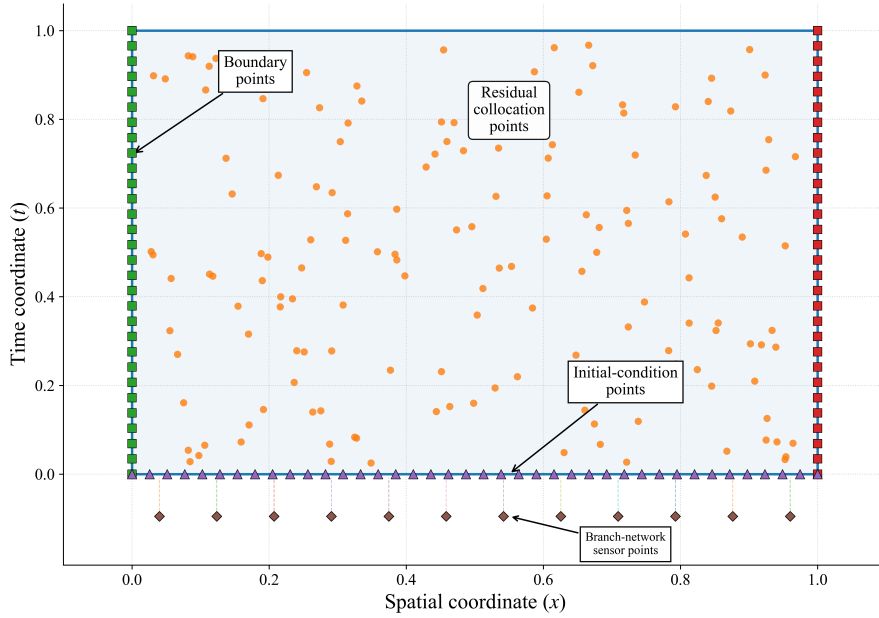


Figure 2. Training domain in the space-time plane for SpectONet. The interior collocation points are sampled inside the domain, while the boundary points are taken along $x = 0$ and $x = 1$, and the initial points are sampled at $t = 0$.

The overall training objective is obtained by combining the residual, ICs, and BCs losses according to Eq. (21). The optimal network parameters are obtained by solving Eq. (25).

$$\theta^* = \arg \min_{\theta} \mathcal{L}(\theta). \quad (48)$$

The principal novelty of the proposed framework lies in the seamless integration of the CGL sensor representation with a physics guided operator learning architecture. In contrast to conventional PI-DeepONet models that rely on uniformly distributed or randomly selected sensors, the proposed SpectONet employs CGL sensors to achieve a more accurate representation of input functions and improved operator-learning performance.

Consequently, the proposed framework unifies mathematically motivated sensor placement, operator learning, and physics-guided training within a single architecture, resulting in improved accuracy and computational efficiency for EBB vibration analysis.

Algorithm 1: Training Procedure of the Proposed SpectONet Framework

- 1 Initialize branch parameters θ_b , trunk parameters θ_t , and set $\theta = \{\theta_b, \theta_t\}$;
 - 2 Construct the CGL sensor locations $x_j = \frac{1}{2} \left(1 - \cos \left(\frac{j\pi}{N} \right) \right)$, $j = 0, 1, \dots, N$;
 - 3 Form the sensor vector $\mathbf{u}_0 = [u_0(x_0), u_0(x_1), \dots, u_0(x_N)]^T$;
 - 4 Generate interior collocation points (x_i^r, t_i^r) , boundary points (x_i^b, t_i^b) , and initial points $(x_i^0, 0)$;
 - 5 Set convergence threshold ε ;
 - 6 **for** each training epoch **do**
 - 7 **for** each mini-batch of ICs $\mathbf{u}_0^{(m)}$ **do**
 - 8 Compute branch features using Eq. (39)
 - 9 Compute trunk features using Eq. (42)
 - 10 Predict the solution in Eq. (43)
 - 11 Compute the PDE residual $R_\theta(u_0)(x, t)$ from Eq. (44) using AD;
 - 12 Compute the residual loss $\mathcal{L}_r$ using Eq. (45);
 - 13 Compute the BC loss $\mathcal{L}_b$ using Eq. (46);
 - 14 Compute the IC loss $\mathcal{L}_i$ using Eq. (47);
 - 15 Compute the total loss according to Eq. (21);
 - 16 Update θ using the Adam optimizer;
 - 17 Refine the network parameters using the L-BFGS optimizer;
 - 18 **if** $\mathcal{L}(\theta) < \varepsilon$ **then**
 - 19 **return** Trained SpectONet model;
 - 20 **end**
 - 21 **end**
 - 22 **end**
 - 23 **return** Trained SpectONet model;
-

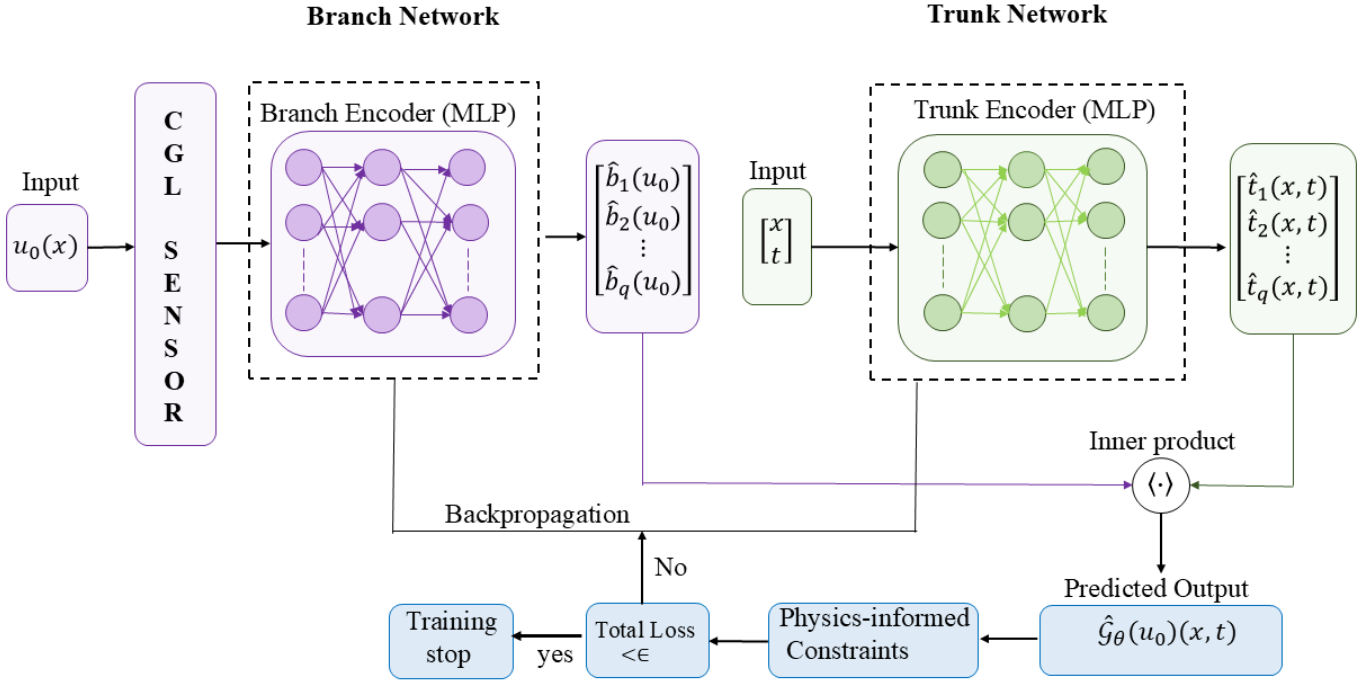


Figure 3. Overall architecture and training workflow of the SpectONet framework for solving EBB vibration problems.

5. Numerical Experiments

This section presents the numerical experiments performed to evaluate the proposed SpectONet framework for solving EBB vibration problems. The performance of the proposed model is compared with four deep learning baseline models, namely PI-DeepONet [32], Vanilla DeepONet [19], PINNs [11], and CNN-UNet [54], in terms of prediction accuracy, convergence behavior, and error distribution. A detailed description of the architectures, training configurations, and implementation settings of these baseline models are provided in [Appendix A](#). Three benchmark problems are considered, namely Problem 1 (P1): forced undamped EBB, Problem 2 (P2): undamped simply supported EBB, and Problem 3 (P3): damped vibration of a variable-coefficient EBB. These problems demonstrate the robustness, accuracy, and generalization capability of the proposed SpectONet framework under different vibration scenarios.

5.1. Error Metrics

To quantitatively assess the predictive accuracy of the proposed framework, the relative $\mathcal{L}_2$ error, root mean square error (RMSE), mean absolute error (MAE), and maximum absolute error (Max Error) are employed. Let $u^{(i)}(x, t)$ and $\hat{\mathcal{G}}_\theta(u_0^{(i)})(x, t)$ denote the exact and predicted solutions, respectively, corresponding to the i -th sampled initial displacement function. Suppose the solutions are evaluated at N points $\{(x_j, t_j)\}_{j=1}^N$. The error metrics are defined as follows:

$$\text{Relative } \mathcal{L}_2 \text{ Error} = \sqrt{\frac{\sum_{j=1}^N \left(u^{(i)}(x_j, t_j) - \hat{\mathcal{G}}_\theta(u_0^{(i)})(x_j, t_j) \right)^2}{\sum_{j=1}^N \left(u^{(i)}(x_j, t_j) \right)^2}}. \quad (49)$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{j=1}^N \left(u^{(i)}(x_j, t_j) - \hat{\mathcal{G}}_\theta(u_0^{(i)})(x_j, t_j) \right)^2}. \quad (50)$$

$$\text{MAE} = \frac{1}{N} \sum_{j=1}^N \left| u^{(i)}(x_j, t_j) - \hat{\mathcal{G}}_\theta(u_0^{(i)})(x_j, t_j) \right|. \quad (51)$$

$$\text{Max Error} = \max_{1 \leq j \leq N} \left| u^{(i)}(x_j, t_j) - \hat{\mathcal{G}}_\theta(u_0^{(i)})(x_j, t_j) \right|. \quad (52)$$

$$\text{Improvement}(\%) = \frac{\text{RMSE}_{\text{baseline}} - \text{RMSE}_{\text{SpectONet}}}{\text{RMSE}_{\text{baseline}}} \times 100\%. \quad (53)$$

5.2. Experimental Setup

All numerical experiments were performed under the same computational environment to ensure a fair comparison among the considered models. The hardware and software configuration used throughout this study is summarized in Table 2.

Table 2. Hardware and software configuration used for the numerical experiments.

Component	Specification
Processor	AMD Ryzen 7 5700U with Radeon Graphics (1.80 GHz)
CPU Configuration	8 Cores / 16 Threads
System Memory	8 GB RAM (7.35 GB usable)
Computing Platform	CPU-based computation
Operating System	Windows 11 (64-bit)
Programming Language	Python 3.11.1
Deep Learning Framework	PyTorch 2.7.1

The network architectures and optimization hyperparameters employed for the proposed SpectONet are summarized in Table 3. Unless otherwise specified, the same optimization strategy and training settings are adopted across all benchmark problems, while only the branch and trunk network architectures are adjusted according to the complexity of each problem.

Table 3. Network architecture and hyperparameters employed for the proposed model.

Parameter	P1	P2	P3
Activation function	Tanh	Tanh	Tanh
Learning rate	$1e - 03$	$2e - 03$	$2e - 03$
Adam iterations	500	500	500
L-BFGS iterations	5000	5000	5000
Evaluation grid	101×101	101×101	101×101
Branch network architecture	[12, 10, 10]	[12, 24, 24]	[12, 32, 32, 32, 32, 32]
Trunk network architecture	[2, 10, 10]	[2, 24, 24]	[2, 32, 32, 32, 32, 32]
Loss weights	[1, 10, 10, 10, 10, 10, 10]	[1, 10, 10, 10, 10, 10, 10]	[1, 10, 10, 10, 10, 10, 10]

5.3. Forced Undamped Euler-Bernoulli Beam

In the first benchmark problem, we have considered the forced undamped EBB vibration equation as depicted in Figure 1a. The proposed SpectONet is trained to learn the solution operator that maps a prescribed external forcing function to the corresponding transverse displacement field. Let $u(x, t)$ denote the transverse displacement of the beam at spatial position x and time t .

For this problem, the parameters in Eq. (5) are chosen as [50]

$$\rho(x) = 1, \quad EI(x) = 1, \quad (54)$$

corresponding to an undamped beam with constant material properties. The external forcing function is assumed to have the separable form

$$f(x, t) = g(x) \cos t, \quad (55)$$

where $g(x)$ denotes the spatial component of the distributed load. Substituting these quantities into Eq. (5) yields

$$\frac{\partial^2 u}{\partial t^2} + \frac{\partial^4 u}{\partial x^4} = f(x, t), \quad x \in [0, 1], \quad t \in [0, 1]. \quad (56)$$

This problem represents a forced undamped EBB, subjected to a harmonic excitation and serves as a benchmark for evaluating the capability of neural operators to learn mappings from input forcing functions to displacement responses. The ICs and BCs are [50]

$$u(x, 0) = x^2(1 - x)^2, \quad \frac{\partial u}{\partial t}(x, 0) = 0.$$

$$u(0, t) = 0, \quad u(1, t) = 0,$$

and

$$\frac{\partial^2 u}{\partial x^2}(0, t) = M_0(t) = 2 \cos t, \quad \frac{\partial^2 u}{\partial x^2}(1, t) = M_1(t) = 2 \cos t.$$

For the forcing function $g(x) = 24 - x^2(1 - x)^2$, the corresponding analytical solution is given by [50]

$$u(x, t) = x^2(1 - x)^2 \cos t. \quad (57)$$

Thus, the operator learning problem is expressed as

$$\widehat{\mathcal{G}}_\theta : g(x) \mapsto u(x, t), \quad (58)$$

where $\widehat{\mathcal{G}}_\theta$ denotes the neural operator approximated by the SpectONet. The branch network receives the values of the spatial forcing function $g(x)$ at selected sensor locations, while the trunk network takes the spatio-temporal coordinates (x, t) as input. The output of the network approximates the displacement response $u(x, t)$ over the computational domain.

In this study, the branch input is constructed using 12 CGL sensor locations over the beam domain $[0, 1]$. Taking $N = 11$ in Eq. (33) yields a total of $N + 1 = 12$ sensor points. Consequently, the branch network receives the discrete forcing representation

$$\mathbf{g} = [g(x_0), g(x_1), \dots, g(x_{11})]^T \in \mathbb{R}^{12}. \quad (59)$$

These CGL sensor points are clustered near the two beam boundaries, thereby improving the representation of boundary-dominated vibration features without increasing the number of sensors.

Let $\{g^{(i)}(x)\}_{i=1}^{N_f}$ denote the sampled forcing functions used for training, where N_f is the number of training input functions. To enforce the governing physics, the residual associated with P1 is defined as

$$R_\theta(g^{(i)})(x, t) = \frac{\partial^2 \widehat{\mathcal{G}}_\theta(g^{(i)})(x, t)}{\partial t^2} + \frac{\partial^4 \widehat{\mathcal{G}}_\theta(g^{(i)})(x, t)}{\partial x^4} - g^{(i)}(x) \cos t. \quad (60)$$

The residual loss is given by

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_f N_r} \sum_{i=1}^{N_f} \sum_{j=1}^{N_r} \left| \frac{\partial^2 \widehat{\mathcal{G}}_\theta(g^{(i)})(x_j^r, t_j^r)}{\partial t^2} + \frac{\partial^4 \widehat{\mathcal{G}}_\theta(g^{(i)})(x_j^r, t_j^r)}{\partial x^4} - g^{(i)}(x_j^r) \cos(t_j^r) \right|^2. \quad (61)$$

where N_r denotes the number of collocation points sampled inside the computational domain. The initial displacement loss is defined as

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_f N_0} \sum_{i=1}^{N_f} \sum_{j=1}^{N_0} \left(\left| \widehat{\mathcal{G}}_\theta(g^{(i)})(x_j^0, 0) - (x_j^0)^2(1 - x_j^0)^2 \right|^2 + \left| \frac{\partial \widehat{\mathcal{G}}_\theta(g^{(i)})(x_j^0, 0)}{\partial t} \right|^2 \right). \quad (62)$$

The BC loss is introduced to enforce the displacement and bending moment conditions and is defined as

$$\begin{aligned} \mathcal{L}_{\text{BC}} = \frac{1}{N_f N_b} \sum_{i=1}^{N_f} \sum_{j=1}^{N_b} & \left(\left| \widehat{\mathcal{G}}_\theta(g^{(i)})(0, t_j^b) - 0 \right|^2 + \left| \widehat{\mathcal{G}}_\theta(g^{(i)})(1, t_j^b) - 0 \right|^2 \right. \\ & \left. + \left| \frac{\partial^2 \widehat{\mathcal{G}}_\theta(g^{(i)})(0, t_j^b)}{\partial x^2} - 2 \cos t_j^b \right|^2 + \left| \frac{\partial^2 \widehat{\mathcal{G}}_\theta(g^{(i)})(1, t_j^b)}{\partial x^2} - 2 \cos t_j^b \right|^2 \right). \end{aligned} \quad (63)$$

The overall training objective is obtained by combining the residual, IC, and BC losses as defined in Eq. (21).

The network architecture and training parameters adopted for this experiment are summarized in Table 3. To enforce the governing equation together with the associated IC and BC, 128 initial points, 128 boundary points, and 1024 interior collocation points randomly sampled from the spatio-temporal domain $(x, t) \in [0, 1] \times [0, 1]$. The proposed SpectONet is trained using a two-stage optimization strategy, where the Adam optimizer is first applied for 500 iterations to obtain a good initial approximation, followed by the L-BFGS optimizer for a maximum of

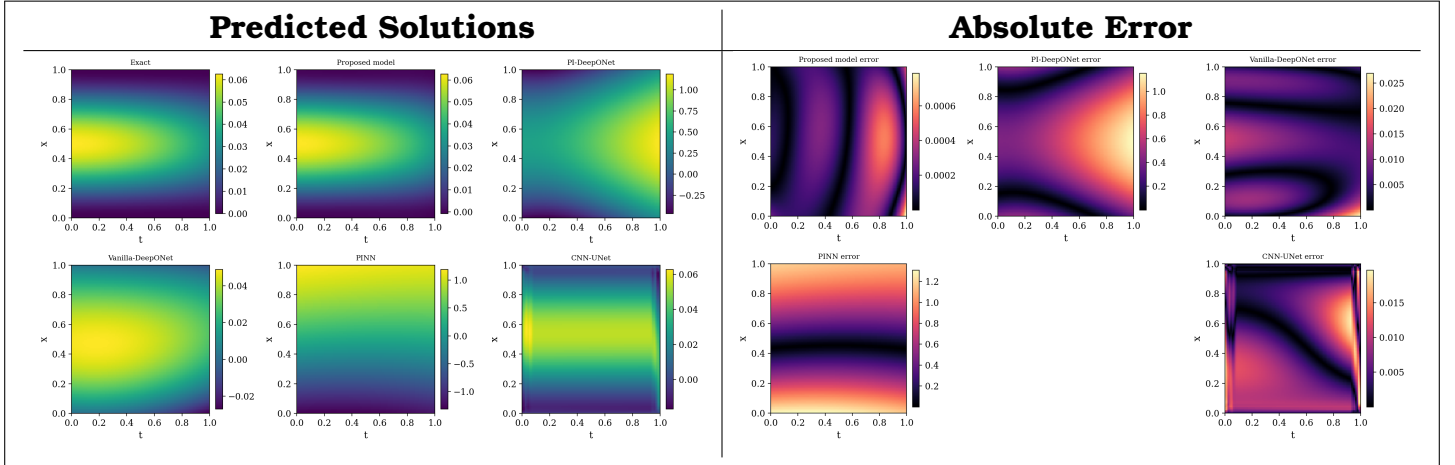


Figure 4. Qualitative comparison of the proposed SpectONet with baselines for P1. The left column presents the 2D predicted solution fields, whereas the right column shows the corresponding point-wise absolute error distributions over the spatio-temporal domain $(x, t) \in [0, 1] \times [0, 1]$.

5000 iterations to further refine the network parameters. This hybrid optimization strategy improves convergence and enhances the overall prediction accuracy.

Figures 4 and 5 present the qualitative comparisons between the exact solution and the corresponding predictions obtained by the proposed SpectONet, and baselines for P1 over the spatio-temporal domain $(x, t) \in [0, 1] \times [0, 1]$. Figure 4 shows the two dimensional (2D) predicted solution fields together with the corresponding point-wise absolute error distributions, whereas Figure 5 presents the corresponding three-dimensional (3D) solution surfaces.

The 2D and 3D comparisons show that the proposed model reproduces the spatial and temporal variations of the exact solution with high fidelity, and its predicted response is visually almost indistinguishable from the exact solution throughout the computational domain. The 3D surface comparison further confirms that SpectONet captures the oscillatory behavior of the beam very well. In contrast, PI-DeepONet and PINN exhibit noticeable discrepancies in the predicted amplitude and spatial distribution. Although Vanilla-DeepONet and CNN-UNet provide comparatively better approximations, they still show visible differences in certain regions.

The point-wise absolute error distributions shown in Figure 4 further demonstrate that SpectONet produces the smallest and most uniformly distributed errors across the spatio-temporal domain. By comparison, PI-DeepONet and PINN yield considerably larger errors, while Vanilla-DeepONet and CNN-UNet exhibit localized error concentrations despite their relatively improved predictions. Overall, the 2D solution maps, 3D surface comparisons, and absolute error distributions confirm that the proposed SpectONet achieves the closest agreement with the exact solution and provides superior predictive accuracy and robustness for P1.

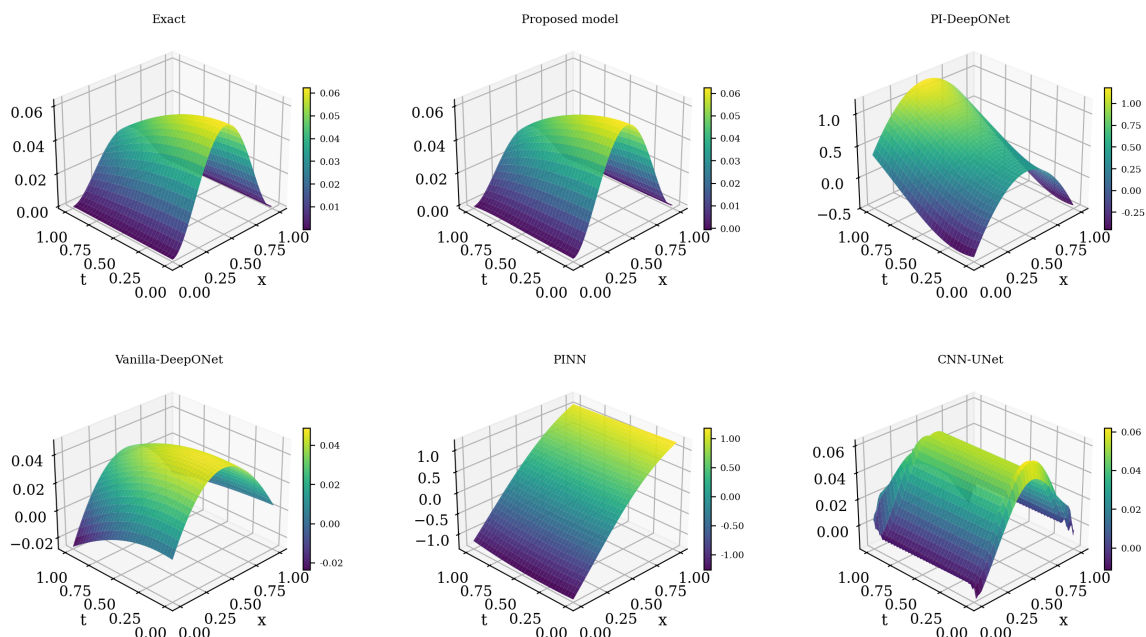


Figure 5. 3D comparison of the exact solution and the predicted solutions obtained by the proposed SpectONet for P1 over the spatio-temporal domain $(x, t) \in [0, 1] \times [0, 1]$.

Table 4 presents the performance comparison of the SpectONet with the competing ap-

proaches for P1. The proposed SpectONet consistently achieves the best performance across all evaluation metrics, demonstrating its superior ability to learn the solution operator for P1. In terms of the relative $\mathcal{L}_2$ error, the SpectONet reduces the prediction error by approximately 28 and 32 times compared with Vanilla-DeepONet and CNN-UNet, respectively, while achieving more than three orders of magnitude lower error than PI-DeepONet and PINN. Similar improvements are observed for the RMSE, MAE, and maximum error, confirming the robustness and predictive accuracy of the SpectONet framework. This improvement can be attributed to the integration of optimized CGL sensor locations with physics-informed constraints, which enables more effective feature representation and physically consistent operator learning.

Table 4. Performance comparison of the proposed model with the baseline models for P1 in terms of the relative $\mathcal{L}_2$ error, RMSE, MAE, maximum error, and parameters.

Model	Relative $\mathcal{L}_2$ Error	RMSE	MAE	Max Error	Parameters
Proposed model	7.349e-03	2.483e-04	2.039e-04	7.891e-04	381
PI-DeepONet	1.657e+01	5.600e-01	4.779e-01	1.154e+00	381
Vanilla-DeepONet	2.055e-01	6.945e-03	5.935e-03	2.696e-02	381
PINN	2.170e+01	7.333e-01	6.424e-01	1.309e+00	381
CNN-UNet	2.320e-01	7.840e-03	6.569e-03	1.969e-02	381

5.4. Undamped Simply Supported Euler-Bernoulli Beam

In this problem, we consider the transverse vibration of an undamped simply supported EBB, subjected to an external forcing function as shown in Figure 1b. The objective is to learn the solution operator that maps the forcing profile to the corresponding displacement field of the beam. Let $u(x, t)$ denote the transverse displacement at spatial position x and time t . For this problem, the parameters in Eq. (9) are chosen as [50]

$$\rho(x) = 1, \quad EI(x) = 1, \quad (64)$$

corresponding to an undamped beam with constant material properties. The external excitation is assumed to possess the separable form

$$f(x, t) = g(x) \cos t, \quad (65)$$

where $g(x)$ denotes the spatial component of the distributed load. Substituting these quantities into Eq. (9) becomes

$$\frac{\partial^2 u}{\partial t^2} + \frac{\partial^4 u}{\partial x^4} = f(x, t), \quad x \in [0, 1], \quad t \in [0, 1]. \quad (66)$$

This problem represents a harmonically forced EBB and serves as a benchmark for assessing the capability of neural operators to learn mappings from input forcing functions to displacement responses. The corresponding ICs and simply supported BCs are given by [50]

$$\begin{aligned} u(x, 0) &= \sin(\pi x), \quad \frac{\partial u}{\partial t}(x, 0) = 0, \\ u(0, t) &= u(1, t) = 0, \quad \frac{\partial^2 u}{\partial x^2}(0, t) = \frac{\partial^2 u}{\partial x^2}(1, t) = 0. \end{aligned}$$

The analytical solution corresponding to the forcing function $g(x) = (\pi^4 - 1) \sin(\pi x)$ is given by [50]

$$u(x, t) = \sin(\pi x) \cos t. \quad (67)$$

Thus, the operator learning problem is expressed as

$$\widehat{\mathcal{G}}_\theta : g(x) \mapsto u(x, t), \quad (68)$$

where $\widehat{\mathcal{G}}_\theta$ is approximated by the SpectONet. The network architecture and training procedure are the same as those used for P1, with the branch network taking the sampled forcing function $g(x)$ as input. The physics-informed residual for P2 is defined as

$$R_\theta(g^{(i)})(x, t) = \frac{\partial^2 \widehat{\mathcal{G}}_\theta(g^{(i)})(x, t)}{\partial t^2} + \frac{\partial^4 \widehat{\mathcal{G}}_\theta(g^{(i)})(x, t)}{\partial x^4} - g^{(i)}(x) \cos t. \quad (69)$$

The residual loss is given by

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_f N_r} \sum_{i=1}^{N_f} \sum_{j=1}^{N_r} \left| \frac{\partial^2 \hat{\mathcal{G}}_\theta(g^{(i)})(x_j^r, t_j^r)}{\partial t^2} + \frac{\partial^4 \hat{\mathcal{G}}_\theta(g^{(i)})(x_j^r, t_j^r)}{\partial x^4} - g^{(i)}(x_j^r) \cos(t_j^r) \right|^2. \quad (70)$$

The ICs loss is defined as

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_f N_0} \sum_{i=1}^{N_f} \sum_{j=1}^{N_0} \left(\left| \hat{\mathcal{G}}_\theta(g^{(i)})(x_j^0, 0) - \phi_d(x_j^0) \right|^2 + \left| \frac{\partial \hat{\mathcal{G}}_\theta(g^{(i)})(x_j^0, 0)}{\partial t} - \phi_v(x_j^0) \right|^2 \right). \quad (71)$$

where

$$\phi_d(x) = \sin(\pi x), \quad \phi_v(x) = 0. \quad (72)$$

The boundary loss is introduced to enforce the simply supported conditions and is given by

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_f N_b} \sum_{i=1}^{N_f} \sum_{j=1}^{N_b} \left(\left| \hat{\mathcal{G}}_\theta(g^{(i)})(0, t_j^b) \right|^2 + \left| \hat{\mathcal{G}}_\theta(g^{(i)})(1, t_j^b) \right|^2 + \left| \frac{\partial^2 \hat{\mathcal{G}}_\theta(g^{(i)})(0, t_j^b)}{\partial x^2} \right|^2 + \left| \frac{\partial^2 \hat{\mathcal{G}}_\theta(g^{(i)})(1, t_j^b)}{\partial x^2} \right|^2 \right). \quad (73)$$

The total loss function is constructed by combining the residual, IC, and BC losses according to Eq. (21). The network architecture and implementation details are summarized in Table 3. To enforce the IC and BC, 128 initial points and 128 boundary points are employed. Furthermore, 1024 interior collocation points are randomly sampled from the spatio-temporal domain $(x, t) \in [0, 1] \times [0, 1]$ to compute the residual loss. The network is trained using the same optimization strategy and hyperparameter settings as those employed for P1.

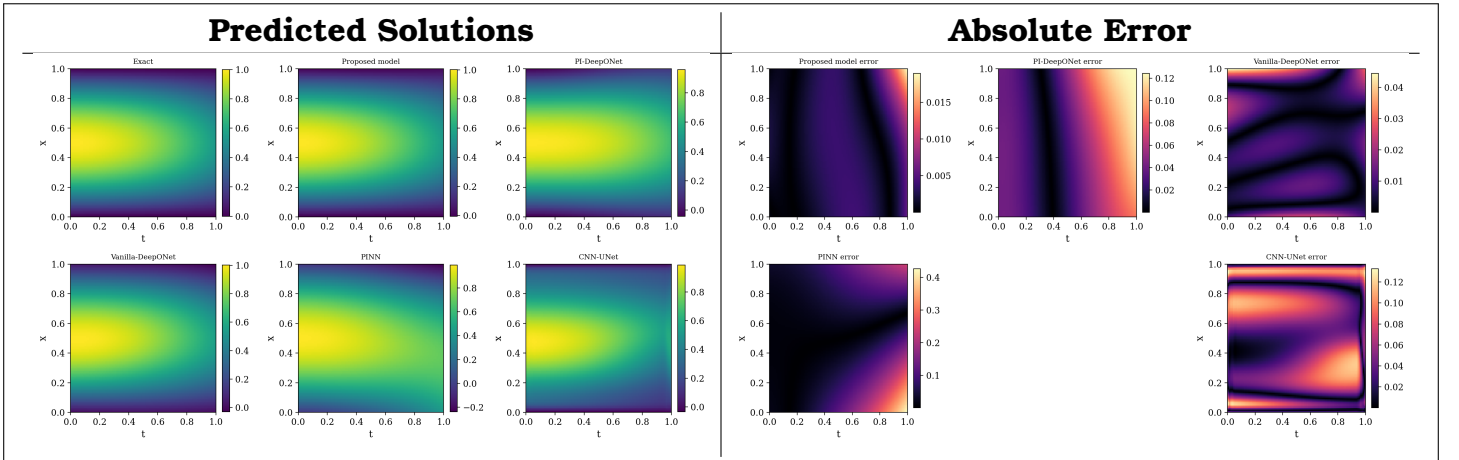


Figure 6. Qualitative comparison of the proposed SpectONet with baselines for P2. The left column presents the 2D predicted solution fields, whereas the right column shows the corresponding point-wise absolute error distributions over the spatio-temporal domain $(x, t) \in [0, 1] \times [0, 1]$.

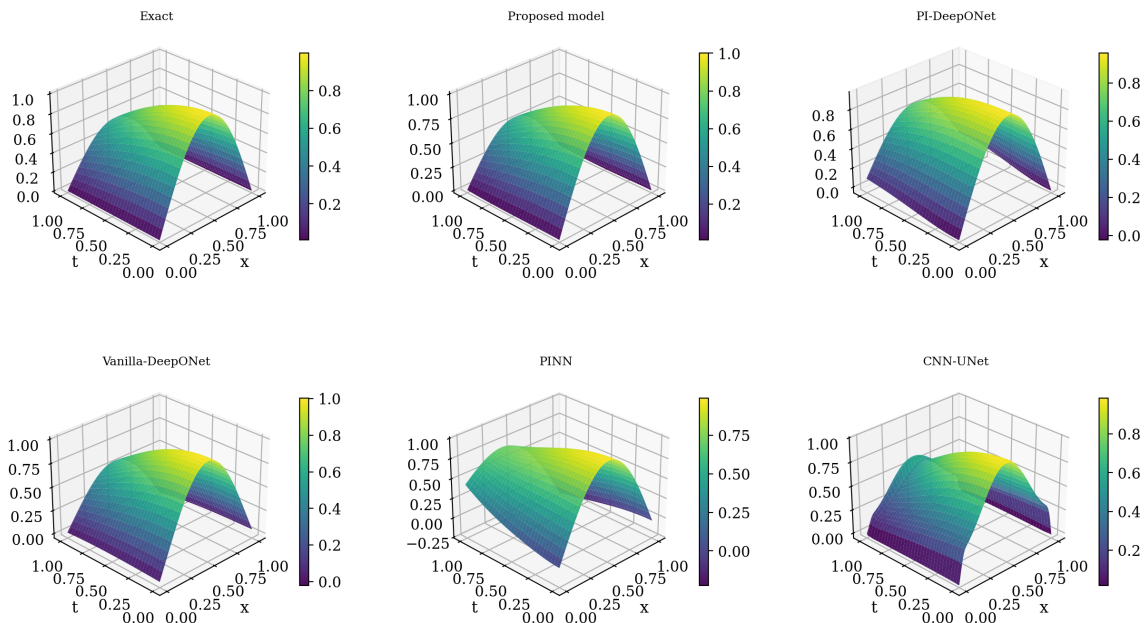


Figure 7. 3D comparison of the exact solution and the predicted solutions obtained by the proposed spectONet with baseline for P2 over the spatio-temporal domain $(x, t) \in [0, 1] \times [0, 1]$.

Figures 6 and 7 present the qualitative comparisons between the exact solution and the corresponding predictions obtained by the proposed SpectONet and the baseline models for P2 over the spatio-temporal domain $(x, t) \in [0, 1] \times [0, 1]$. Figure 6 shows the 2D predicted solution fields together with the corresponding point-wise absolute error distributions, whereas Figure 7 presents the corresponding 3D solution surfaces.

The 2D and 3D comparisons show that the SpectONet reproduces the exact solution, with the predicted response being visually almost indistinguishable from the exact solution throughout the computational domain. In contrast, PI-DeepONet and PINN show noticeable discrepancies from the exact solution, whereas Vanilla-DeepONet and CNN-UNet provide comparatively better approximations but still exhibit visible differences in certain regions. These qualitative comparisons demonstrate the superior capability of SpectONet to learn the solution operator associated with P2.

The corresponding absolute error distributions shown in Figure 6 further indicate that SpectONet produces the smallest and most uniformly distributed errors over the spatio-temporal domain. By comparison, PI-DeepONet and PINN yield considerably larger errors, while Vanilla-DeepONet and CNN-UNet exhibit localized error concentrations despite their relatively improved predictions. Overall, the 2D solution maps, 3D surface comparisons, and absolute error distributions confirm the superior predictive accuracy, robustness, and generalization capability of the SpectONet for P2.

Table 5. Performance comparison of the proposed model with the baseline models for P2 in terms of the relative $\mathcal{L}_2$ error, RMSE, MAE, maximum error, and parameters.

Model	Rel. $\mathcal{L}_2$ Error	RMSE	MAE	Max Error	Parameters
Proposed model	5.829e-03	3.496e-03	2.651e-03	1.894e-02	1585
PI-DeepONet	1.007e-01	6.042e-02	5.001e-02	1.243e-01	1585
Vanilla-DeepONet	1.647e-02	9.880e-03	7.685e-03	4.455e-02	1585
PINN	1.699e-01	1.019e-01	6.906e-02	4.268e-01	1585
CNN-UNet	1.033e-01	6.198e-02	5.350e-02	1.327e-01	1585

Table 5 summarizes the quantitative performance of the SpectONet and the competing approaches for P2. The SpectONet consistently achieves the best performance across all evaluation metrics, demonstrating its superior predictive accuracy. In terms of the relative $\mathcal{L}_2$ error, SpectONet achieves approximately 2.8 times, 17 times, 29 times, and 17 times lower errors than Vanilla-DeepONet, PI-DeepONet, PINN, and CNN-UNet, respectively. These results highlight the superior parameter efficiency of the SpectONet. This performance gain is attributed to the integration of optimized CGL sensor locations and physics-informed constraints.

5.5. Damped Vibration of a Variable-Coefficient Euler-Bernoulli Beam

In this problem, we consider the damped vibration of a variable-coefficient EBB as shown in Figure 1c. The objective is to learn the solution operator associated with the transient response of the beam. Let $u(x, t)$ denote the transverse displacement at spatial position x and time t .

For this problem, the material and damping parameters in Eq. (13) are chosen as [7]

$$\rho(x) = 1, \quad c(x) = 2, \quad EI(x) = 1 + x, \quad (74)$$

with zero external excitation, i.e., $F(x, t) = 0$. Substituting these quantities into Eq. (13) yields the governing equation

$$\frac{\partial^2 u}{\partial t^2} + 2 \frac{\partial u}{\partial t} + (1 + x) \frac{\partial^4 u}{\partial x^4} + 2 \frac{\partial^3 u}{\partial x^3} = 0, \quad x \in [0, 1], \quad t \in [0, 1.5]. \quad (75)$$

This problem represents a damped EBB with spatially varying flexural rigidity, making it a challenging benchmark for evaluating the capability of neural operators to learn transient vibration dynamics. The corresponding ICs and BCs are prescribed according to the literature [7] as

$$u(x, 0) = u_0(x), \quad \frac{\partial u}{\partial t}(x, 0) = -2u_0(x),$$

$$u(0, t) = u(1, t) = 0,$$

and

$$\frac{\partial^2 u}{\partial x^2}(0, t) = B_0(t) = 2e^{-2t}, \quad \frac{\partial^2 u}{\partial x^2}(1, t) = B_1(t) = 2e^{-2t}.$$

The analytical solution corresponding to this problem with $u_0(x) = x^2$ is given by [7]

$$u(x, t) = x^2 e^{-2t}. \quad (76)$$

Accordingly, the operator learning task is formulated as

$$\widehat{\mathcal{G}}_\theta : u_0(x) \mapsto u(x, t), \quad (77)$$

where $\widehat{\mathcal{G}}_\theta$ denotes the neural operator approximated by the proposed SpectONet framework. The branch network receives the values of the initial displacement profile sampled at selected sensor locations, whereas the trunk network takes the spatio-temporal coordinates (x, t) as input. Their combination provides an approximation to the displacement field over the computational domain.

In this study, the branch input is constructed using 12 CGL sensor locations over the beam domain $[0, 1]$. Taking $N = 11$ in Eq. (33) yields a total of $N + 1 = 12$ sensor points. Consequently, the branch network receives the discrete input vector

$$\mathbf{u}_0 = [u_0(x_0), u_0(x_1), \dots, u_0(x_{11})]^T \in \mathbb{R}^{12}. \quad (78)$$

These CGL sensor points are clustered near the two beam boundaries, thereby improving the representation of boundary-dominated vibration features without increasing the number of sensors.

$$R_\theta(u_0^{(i)})(x, t) = \frac{\partial^2 \widehat{\mathcal{G}}_\theta(u_0^{(i)})(x, t)}{\partial t^2} + 2 \frac{\partial \widehat{\mathcal{G}}_\theta(u_0^{(i)})(x, t)}{\partial t} + (1+x) \frac{\partial^4 \widehat{\mathcal{G}}_\theta(u_0^{(i)})(x, t)}{\partial x^4} + 2 \frac{\partial^3 \widehat{\mathcal{G}}_\theta(u_0^{(i)})(x, t)}{\partial x^3}. \quad (79)$$

The residual loss is given by

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_f N_r} \sum_{i=1}^{N_f} \sum_{j=1}^{N_r} \left| R_\theta(u_0^{(i)})(x_j^r, t_j^r) \right|^2. \quad (80)$$

The IC loss is defined as

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_f N_0} \sum_{i=1}^{N_f} \sum_{j=1}^{N_0} \left(\left| \widehat{\mathcal{G}}_\theta(u_0^{(i)})(x_j^0, 0) - u_0^{(i)}(x_j^0) \right|^2 + \left| \frac{\partial \widehat{\mathcal{G}}_\theta(u_0^{(i)})(x_j^0, 0)}{\partial t} + 2u_0^{(i)}(x_j^0) \right|^2 \right). \quad (81)$$

The BC loss defined as

$$\begin{aligned} \mathcal{L}_{\text{BC}} = \frac{1}{N_f N_b} \sum_{i=1}^{N_f} \sum_{j=1}^{N_b} & \left(\left| \widehat{\mathcal{G}}_\theta(u_0^{(i)})(0, t_j^b) \right|^2 + \left| \widehat{\mathcal{G}}_\theta(u_0^{(i)})(1, t_j^b) \right|^2 \right. \\ & \left. + \left| \frac{\partial^2 \widehat{\mathcal{G}}_\theta(u_0^{(i)})(0, t_j^b)}{\partial x^2} - 2e^{-2t_j^b} \right|^2 + \left| \frac{\partial^2 \widehat{\mathcal{G}}_\theta(u_0^{(i)})(1, t_j^b)}{\partial x^2} - 2e^{-2t_j^b} \right|^2 \right). \end{aligned} \quad (82)$$

The total loss function is constructed by combining the residual, IC, and BC losses according to Eq. (21). The network architecture and training hyperparameters are summarized in Table 3. To enforce the IC and BC, 48 initial points and 48 boundary points are employed. Furthermore, 256 interior collocation points are randomly sampled from the spatio-temporal domain $(x, t) \in [0, 1] \times [0, 1]$ to evaluate the residual loss. Unless otherwise stated, the remaining training settings are identical to those adopted for P1.

Figures 8 and 9 present qualitative comparisons between the exact solution and the predictions obtained by the proposed SpectONet and the baseline models for Problem P3 over the spatio-temporal domain $(x, t) \in [0, 1] \times [0, 1.5]$. Figure 8 shows the 2D solution fields together with the corresponding point-wise absolute error distributions, whereas Figure 9 presents the corresponding 3D solution surfaces. The 2D and 3D comparisons show that SpectONet reproduces the exact solution, with its predicted response being visually almost indistinguishable from the reference solution throughout the computational domain. In contrast, PI-DeepONet exhibits more deviations, while Vanilla-DeepONet, PINN, and CNN-UNet provide comparatively better predictions but still display visible discrepancies in several regions.

The absolute error distributions shown in Figure 8 further demonstrate that SpectONet produces the smallest and most uniformly distributed errors across the spatio-temporal domain. PI-DeepONet yields considerably larger errors, whereas Vanilla-DeepONet, PINN, and

CNN-UNet exhibit localized error concentrations despite their comparatively improved predictions. Overall, the 2D solution maps, 3D surface comparisons, and absolute error distributions consistently confirm that the SpectONet effectively captures the damped dynamics and variable-coefficient behavior of P3, demonstrating superior predictive accuracy, spatio-temporal consistency, and robustness compared with the considered baseline models.

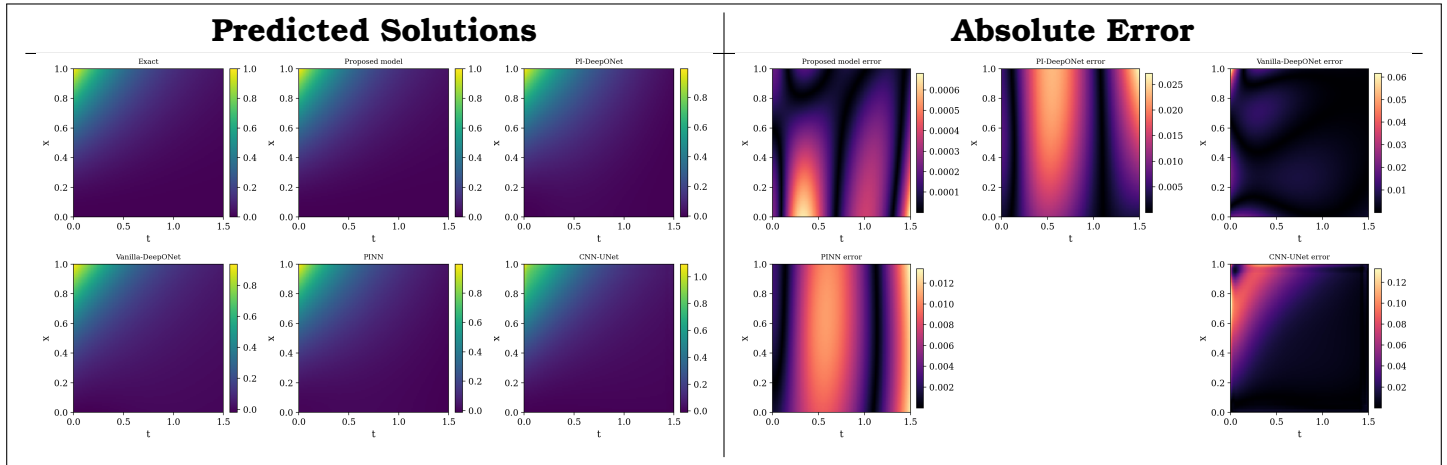


Figure 8. Qualitative comparison of the proposed SpectONet with baselines for P3. The left column presents the 2D predicted solution fields, whereas the right column shows the corresponding point-wise absolute error distributions over the spatio-temporal domain $(x, t) \in [0, 1] \times [0, 1.5]$.

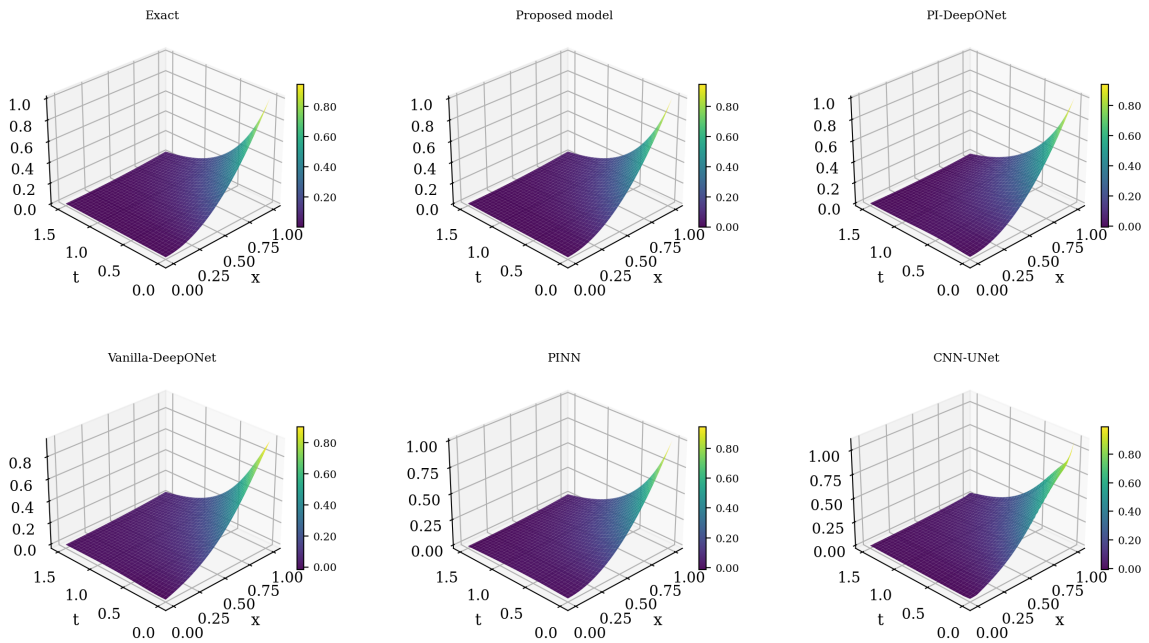


Figure 9. 3D comparison of the exact solution and the predicted solutions obtained by the proposed spectONet with baselines for P3 over the spatio-temporal domain $(x, t) \in [0, 1] \times [0, 1.5]$.

Table 6. Performance comparison of the proposed model with the baseline models for P3 in terms of the relative $\mathcal{L}_2$ error, RMSE, MAE, maximum error, and parameters.

Model	Rel. $\mathcal{L}_2$ Error	RMSE	MAE	Max Error	Parameters
Proposed model	1.215e-03	2.255e-04	1.785e-04	6.813e-04	6849
PI-DeepONet	6.695e-02	1.242e-02	1.047e-02	2.718e-02	6849
Vanilla-DeepONet	3.065e-02	5.688e-03	4.031e-03	6.167e-02	6849
PINN	3.756e-02	6.970e-03	6.065e-03	1.340e-02	6849
CNN-UNet	1.713e-01	3.179e-02	2.078e-02	1.331e-01	6849

Table 6 presents the performance comparison of the proposed SpectONet with the baseline models for P3. The SpectONet consistently delivers the best performance across all evaluation metrics, demonstrating its superior predictive capability. In terms of the relative $\mathcal{L}_2$ error, SpectONet achieves approximately 55, 25, 31, and 141 times lower errors than PI-DeepONet, Vanilla-DeepONet, PINN, and CNN-UNet, respectively. This improvement can be attributed to the combination of optimized CGL sensor locations and physics-informed constraints.

6. Discussion

This section provides a comprehensive discussion of the performance of the SpectONet beyond the quantitative error metrics presented earlier. Specifically, the predicted and exact displacement profiles are compared, the training and testing loss histories are analyzed to evaluate convergence and generalization, and an ablation study is conducted to investigate the influence of key hyperparameters on the robustness and predictive performance of the proposed framework.

6.1. Spatial Displacement Profiles at Different Time Instants

The numerical results presented in the previous section 5 demonstrate the high prediction accuracy of the SpectONet through quantitative error metrics and visual comparison. To further assess its generalization capability, Figure 10 compares the predicted and exact spatial displacement profiles for the three undertaken problems at representative time instants. These comparisons provide a qualitative evaluation of SpectONet to capture different beam dynamics under varying governing equations and physical conditions. Across all cases, the

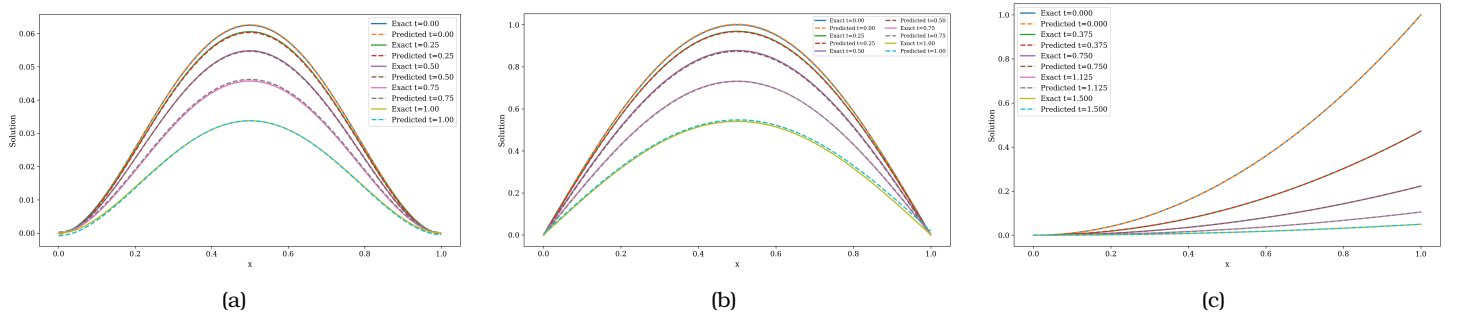


Figure 10. Comparison of the exact and SpectONet-predicted spatial displacement profiles for the three benchmark EBB problems P1, P2, and P3 at representative time instants.

predicted solutions exhibit excellent agreement with the corresponding analytical solutions throughout the spatial domain, with the predicted curves almost completely overlapping the exact curves. These comparisons demonstrate the ability of SpectONet to learn the underlying solution operator under different physical settings.

For P1, Figure 10a compares the predicted and exact displacement profiles at the representative time instants $t = 0.00, 0.25, 0.50, 0.75$, and 1.00 . The close agreement between the two solutions at all representative time instants demonstrates the capability of the SpectONet to capture the dynamic behavior of the beam.

For P2, Figure 10b the predicted displacement profiles remain in close agreement with the exact solutions at $t = 0.00, 0.25, 0.50, 0.75$, and 1.00 . The SpectONet closely reproduces the polynomial displacement profiles induced by the external forcing, exhibiting only negligible deviations from the exact solution. The nearly overlapping curves indicate that SpectONet effectively learns the forced beam dynamics and satisfies the governing equation together with the prescribed IC and BC.

For P3, Figure 10c the displacement profiles at $t = 0.00, 0.375, 0.750, 1.125$, and 1.500 demonstrate that SpectONet effectively predicts the damped vibration response despite the presence of variable coefficients. The predicted solutions closely follow the analytical profiles at every time instant and correctly capture the progressive reduction in vibration amplitude. This excellent agreement highlights the robustness and stability of the SpectONet framework for solving more challenging beam vibration problems.

Overall, the consistently close agreement between the predicted and exact displacement profiles across all three benchmark problems confirms the high predictive accuracy, robustness, and strong generalization capability of the SpectONet. These qualitative observations are consistent with the quantitative error metrics reported in the previous section and further demonstrate the effectiveness of SpectONet as a reliable operator-learning framework for EBB vibration analysis.

6.2. Training and Testing Loss Analysis

The training and testing loss histories of the SpectONet and the baseline models for P1, P2, and P3 are presented in Figure 11. These curves provide valuable insights into the optimization efficiency, learning behavior, and generalization capability of the considered approaches.

For P1, the SpectONet exhibits rapid convergence during the initial training stages and subsequently reduces both the training and testing losses in a smooth and stable manner, as shown in Figure 11a. The close alignment between the training and testing curves indicates

strong generalization capability without evidence of overfitting. In contrast, PI-DeepONet and PINN exhibit slower convergence rates and maintain comparatively higher loss values throughout the training process. Vanilla-DeepONet achieves improved convergence relative to these methods, while CNN-UNet also demonstrates stable optimization behavior. Nevertheless, the SpectONet attains the second-lowest final testing loss among all competing approaches while achieving the lowest RMSE, MAE, relative $\mathcal{L}_2$ error, and maximum error.

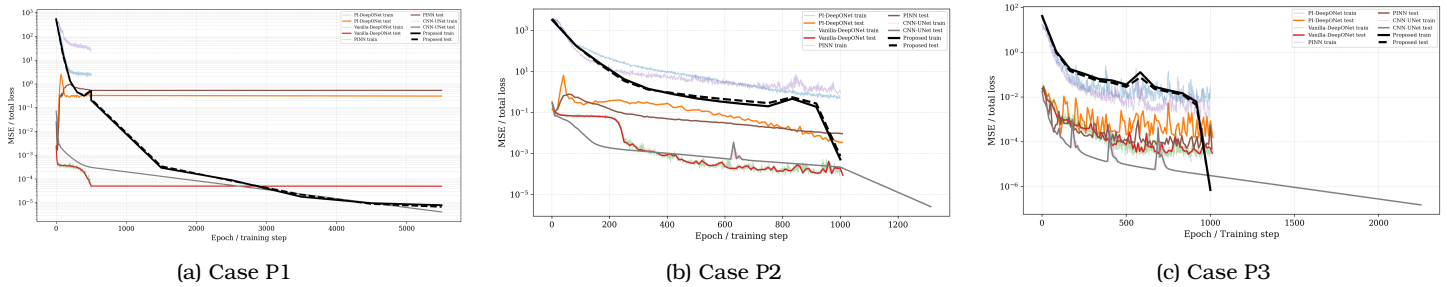


Figure 11. Training and testing loss histories for the proposed SpectONet and the baseline models across the three benchmark problems.

As shown in Figure 11b, a similar trend is observed for P2. The SpectONet framework consistently decreases both training and testing losses and achieves the second smallest final error. PI-DeepONet shows slower convergence along with noticeable fluctuations in the testing loss, indicating limited stability. PINN also converges slowly and remains at relatively higher loss levels. Vanilla-DeepONet improves the convergence rate compared to PI-DeepONet, while CNN-UNet provides a smooth optimization trajectory.

For the more challenging variable-coefficient damped beam problem P3, SpectONet maintains stable optimization and converges to the lowest training and testing losses, as shown in Figure 11c. Despite the increased complexity of the governing vibration equation, the proposed model exhibits smooth and consistent convergence. In comparison, PI-DeepONet and PINN converge more slowly and retain higher loss values, whereas Vanilla-DeepONet provides moderate improvement. Although CNN-UNet also demonstrates stable convergence, its final loss remains higher than that achieved by SpectONet.

Overall, the training and testing loss histories demonstrate that SpectONet consistently achieves faster and more stable optimization, lower loss values, and better generalization than the baseline models across all three benchmark problems. The integration of optimized CGL sensor locations with physics-informed learning improves the optimization process and enables accurate approximation of the underlying solution operator, highlighting the robustness and effectiveness of the proposed SpectONet framework.

6.3. Ablation Study

To further test the robustness and sensitivity of the proposed SpectONet, an ablation study is conducted on P1 only. Four important aspects that directly influence the training process are investigated: (i) the effect of random parameter initialization, (ii) the influence of different physics-informed loss-weight configurations, (iii) the impact of the number of collocation points used for enforcing the governing equation and associated constraints. (iv) the sensitivity of individual loss components. In each experiment, only one factor is varied while all remaining hyperparameters are kept unchanged, thereby enabling an independent assessment of its influence on the predictive performance of the SpectONet framework.

6.3.1. Effect of Random Initialization

To assess the robustness of the SpectONet with respect to random initialization, experiments were conducted using four different random seeds while keeping all other training settings fixed. The corresponding error metrics are reported in Table 7, where the highlighted row denotes the random seed adopted throughout this study.

Table 7. Effect of random initialization on the predictive performance of the proposed model for P1

Seed	N_r	N_b	N_i	Loss Weights	Relative $\mathcal{L}_2$	RMSE	MAE	Max Error
42	1024	128	128	(1, 10, 10, 10, 10, 10, 10)	$6.129e-02$	$2.071e-03$	$1.657e-03$	$6.641e-03$
2024	1024	128	128	(1, 10, 10, 10, 10, 10, 10)	$6.814e-02$	$2.303e-03$	$1.775e-03$	$7.502e-03$
20260628	1024	128	128	(1, 10, 10, 10, 10, 10, 10)	$3.035e-02$	$1.026e-03$	$8.461e-04$	$3.858e-03$
1234	1024	128	128	(1, 10, 10, 10, 10, 10, 10)	$7.348e-03$	$2.483e-04$	$2.039e-04$	$7.891e-04$

The SpectONet consistently achieves accurate predictions across different random initializations. Although slight variations are observed in the error metrics, the overall performance remains stable, indicating that the optimization process is not highly sensitive to the choice of random seed. The configuration with seed **1234**, highlighted in the table, was used to generate all experimental results presented in this paper and achieved the best predictive accuracy among the tested seeds. The remaining initialization settings also produce competitive results with only marginal degradation in performance, confirming the robustness, reproducibility, and stable convergence behavior of the SpectONet.

6.3.2. Effect of Loss-Weight Configuration

The performance of physics-informed neural operator models strongly depends on the relative weighting assigned to the governing PDE residual and the associated BC and IC losses. To investigate the influence of the loss-weight configuration, several weighting strategies were evaluated while keeping the others fixed. The corresponding prediction errors are summarized in Table 8. The highlighted row denotes the loss-weight configuration adopted throughout this study.

Table 8. Influence of different loss-weight configurations on the predictive performance of the proposed model for P1.

Configuration	Seed	N_r	N_b	N_i	Loss Weights	Relative $\mathcal{L}_2$	RMSE	MAE	Max Error
Balanced	1234	1024	128	128	(1, 1, 1, 1, 1, 1, 1)	$4.679e-02$	$1.581e-03$	$1.302e-03$	$5.392e-03$
BC/IC-heavy	1234	1024	128	128	(1, 20, 20, 20, 20, 20, 20)	$2.977e-02$	$1.006e-03$	$7.950e-04$	$2.834e-03$
PDE-heavy	1234	1024	128	128	(10, 10, 10, 10, 10, 10, 10)	$1.873e-02$	$6.328e-04$	$4.681e-04$	$4.428e-03$
Reduced PDE Weight	1234	1024	128	128	(0.1, 10, 10, 10, 10, 10, 10)	$1.489e-02$	$5.032e-04$	$3.601e-04$	$3.420e-03$
Default	1234	1024	128	128	(1, 10, 10, 10, 10, 10, 10)	$7.348e-03$	$2.483e-04$	$2.039e-04$	$7.891e-04$

The proposed SpectONet is strongly influenced by the choice of loss-weight configuration. The adopted configuration, highlighted in the table and used throughout this study, achieves the best performance across all error metrics, indicating an effective balance between enforcing the PDE residual and satisfying the IC and BC constraints. Reducing the contribution of the PDE residual (reduced PDE weight), assigning excessive emphasis to the PDE residual (PDE-heavy), increasing the IC/BC weights (BC/IC-heavy), or assigning equal weights to all loss components (balanced) leads to degraded prediction accuracy. These results demonstrate that an appropriate loss-weighting strategy is essential for accurate, stable, and physically consistent operator learning.

6.3.3. Effect of Collocation-Point Density

The number of collocation points determines how effectively the governing PDE and the associated IC and BC are enforced during training. To investigate the influence of the collocation-point density, several sampling configurations were evaluated while keeping others fixed.

Table 9. Influence of different collocation-point densities on the predictive performance of the proposed model for P1.

Configuration	Seed	N_r	N_b	N_i	Loss Weights	Relative $\mathcal{L}_2$	RMSE	MAE	Max Error
Small	1234	256	64	64	(1, 10, 10, 10, 10, 10, 10)	$8.411e-03$	$2.842e-04$	$2.295e-04$	$1.182e-03$
Medium	1234	512	128	128	(1, 10, 10, 10, 10, 10, 10)	$2.933e-02$	$9.910e-04$	$7.968e-04$	$3.379e-03$
Default	1234	1024	128	128	(1, 10, 10, 10, 10, 10, 10)	$7.348e-03$	$2.483e-04$	$2.039e-04$	$7.891e-04$
Large	1234	2048	256	256	(1, 10, 10, 10, 10, 10, 10)	$7.225e-03$	$2.374e-04$	$1.869e-04$	$6.934e-04$

Table 9 presents the influence of different collocation-point densities on the predictive performance of the SpectONet. The adopted (default) configuration, highlighted in the table and used throughout this study, provides an effective balance between prediction accuracy and computational cost. Reducing the number of collocation points degrades the predictive performance due to insufficient physics information during training. Although the large configuration achieves slightly lower error metrics, the improvement is marginal while requiring nearly twice the training time. Therefore, the default configuration is adopted for all subsequent experiments as it offers the best trade-off between accuracy and computational cost. These results demonstrate the robustness of the SpectONet with respect to the collocation-point density.

6.3.4. Sensitivity of Loss Components

To understand the contribution of each loss component in the proposed SpectONet framework, the individual roles of the physics-residual loss $\mathcal{L}_{\text{phys}}$, IC loss $\mathcal{L}_{\text{IC}}$, and BC loss $\mathcal{L}_{\text{BC}}$ are investigated by selectively removing each term while keeping all other training settings unchanged. Excluding the physics-residual loss converts the model into a predominantly data-driven framework, which may weaken its physical consistency and generalization capability. Similarly, removing the IC loss can lead to an incorrectly initialized solution whose error propagates over time, while excluding the BC loss may produce noticeable violations near the spatial boundaries. The results reported in Table 10 demonstrate that the best predictive performance is achieved when all loss components are incorporated jointly with appropriate weights.

For a fair comparison, the random seed, network architecture, loss weights of the retained components, collocation-point distribution, optimizer settings, and number of training iterations are kept unchanged across all configurations. This systematic comparison makes it possible to identify the individual contribution of each loss component and determine whether the physics residual, IC, and BC constraints act independently or cooperatively in improving the accuracy and physical consistency of the SpectONet predictions.

Table 10. Sensitivity of different loss-component in SpectONet for P1.

Configuration	Physics Loss	IC Loss	BC Loss	Relative $\mathcal{L}_2$ Error
Default	✓	✓	✓	7.348e − 03
Physics only	✓	×	×	1.243e-01
Protocol 1	×	✓	×	7.800e+01
Protocol 2	×	×	✓	8.980e+02
Protocol 3	✓	✓	×	3.634e-02
Protocol 4	✓	×	✓	5.359e-02
Protocol 5	×	✓	✓	2.634e+01
Protocol 6	×	×	×	6.634e+03

6.3.5. Summary

Overall, the ablation study demonstrates that the proposed SpectONet exhibits stable and robust performance under different training hyperparameter settings. The framework consistently converges under different random initializations, achieves the highest prediction accuracy using the selected default loss-weight configuration, and exhibits only marginal accuracy improvements when the collocation-point density is increased beyond the default setting, at the expense of significantly higher computational cost and training time. Furthermore, the loss-component sensitivity analysis confirms that the combined enforcement of the physics, IC, and BC losses is essential for obtaining accurate and physically consistent predictions. Removing one or more of these loss components leads to a noticeable deterioration in the predictive performance, indicating that each component contributes meaningfully to the overall training process. These observations validate the robustness, stability, and strong generalization capability of the SpectONet while highlighting its ability to achieve an effective balance between predictive accuracy and computational efficiency for solving forced EBB vibration problems.

7. Real-World Structural Vibration Validation

Although the proposed SpectONet is developed using synthetic EBB simulations, it is important to investigate whether the learned operator can generalize to real world measurements. To this end, an additional validation study has been conducted using the Switzerland Z24 bridge vibration dataset [55, 56], a widely used benchmark in SHM. The proposed SpectONet reconstructs the complete bridge vibration field from only a sparse subset of sensors selected using the proposed CGL inspired sensor placement strategy.

7.1. Z24 Bridge Dataset

The Z24 bridge was an overpass of the national highway A1 between Bern and Zurich, Switzerland. The dataset contains vibration responses collected from 27 uniaxial accelerometer channels under 17 different structural conditions, including the healthy state and multiple

progressive damage scenarios. Each vibration recording is associated with an integer label indicating its corresponding structural condition.

The processed dataset is represented as $\mathbf{X} \in \mathbb{R}^{1530 \times 27 \times 6000}$, where 1530 is the number of vibration recordings, 27 is the number of accelerometer sensors, and 6000 represents the temporal acceleration samples recorded by each sensor. Examining the dataset, we assume the sampling frequency adopted during preprocessing is 100 Hz, and each vibration recording spans approximately 60 s.

7.2. Experimental Setup

From the vibration recordings from 27 accelerometers, only the measurements from the selected 7 sensors are used as the training dataset. Therefore, SpectONet learns a nonlinear mapping from sparse sensor measurements to the complete structural response.

The vibration signals are normalized using robust statistics computed from the training set,

$$\tilde{u} = \frac{u - \text{median}}{\text{IQR}/1.349 + \epsilon}, \quad (83)$$

where u represents the raw acceleration measurement recorded by a sensor, IQR denote the inter quartile range computed from the training data, and $\epsilon = 10^{-8}$ is a small constant introduced for numerical stability. The normalized values are clipped to the interval $[-30, 30]$ to reduce the influence of extreme outliers. During evaluation, the predicted responses are transformed back to the original acceleration scale before computing the reported error metrics.

7.3. Proposed Model

The branch network of SpectONet encodes the vibration histories measured at a sparse set of input sensors, whereas the trunk network encodes the spatial coordinate of the query sensor together with the corresponding time coordinate. The outputs of the two networks are combined to predict the vibration response at each queried sensor-time location.

The model is trained by minimizing an objective function that combines a weighted reconstruction loss with spectral, temporal, and spatial consistency terms. The total loss is defined as

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda_f \mathcal{L}_{\text{fft}} + \lambda_t \mathcal{L}_{\text{temp}} + \lambda_s \mathcal{L}_{\text{spat}}, \quad (84)$$

where $\mathcal{L}_{\text{rec}}$ denotes the reconstruction loss, $\mathcal{L}_{\text{fft}}$ enforces consistency in the frequency domain, $\mathcal{L}_{\text{temp}}$ promotes temporal consistency, and $\mathcal{L}_{\text{spat}}$ preserves spatial consistency across the sensor locations.

To assign greater importance to high-amplitude vibration responses, a peak-aware weighted reconstruction loss is employed:

$$\mathcal{L}_{\text{rec}} = \frac{1}{N} \sum_{i=1}^N [1 + \alpha \mathbb{I}(|u_i| > \tau)] (\hat{u}_i - u_i)^2, \quad (85)$$

where u_i and $\hat{u}_i$ denote the measured and predicted vibration responses, respectively, N is the total number of sensor-time samples, and $\mathbb{I}(\cdot)$ denotes the indicator function. The peak-aware weighting parameters are set as $\alpha = 0.25$ and $\tau = 3.00$. The weights of the auxiliary loss components are selected as

$$\lambda_f = 10^{-4}, \quad \lambda_t = 10^{-2}, \quad \lambda_s = 2 \times 10^{-3}. \quad (86)$$

7.4. Quantitative Results and Discussions

Table 11 compares the reconstruction performance of the proposed SpectONet with three baseline methods: nearest-input-sensor copying, linear spatial interpolation, and a coordinate-based MLP. The RMSE is reported for both the complete dataset and the unseen test dataset.

The nearest-input-sensor and linear-interpolation baselines exhibit similar reconstruction errors, indicating that simple local copying or spatial interpolation is insufficient to capture the complete dynamic response of the bridge. In contrast, SpectONet learns a nonlinear mapping between the sparse measurements and the complete sensor field, thereby enabling it to recover spatially and temporally varying vibration patterns.

Although the coordinate-MLP baseline achieves the lowest RMSE when the complete dataset is considered, the proposed SpectONet obtains the lowest RMSE on the unseen test dataset. In

Table 11. Comparison of the proposed model with baseline vibration-field reconstruction methods.

Method	RMSE (all 27 sensors)	RMSE (test sensors)	Improvement (%) test sensors only
Proposed Model	$8.2647e - 04$	$6.0758e - 04$	NA
Nearest input sensor	$1.1742e - 03$	$9.7591e - 04$	38
Linear interpolation	$1.1735e - 03$	$9.7288e - 04$	38
MLP baseline	$7.4306e - 04$	$6.9898e - 04$	13

particular, SpectONet reduces the test RMSE by approximately 13.08% relative to the coordinate-MLP baseline, 37.74% relative to the nearest-input-sensor method, and 37.55% relative to linear interpolation. Since the primary objective of sparse-sensor reconstruction is to predict vibration responses at unmeasured locations, performance on the unseen test recordings and non-input sensor locations provides the most relevant measure of generalization. The results indicate that the proposed SpectONet architecture is more effective than the considered baseline methods in reconstructing previously unseen structural responses.

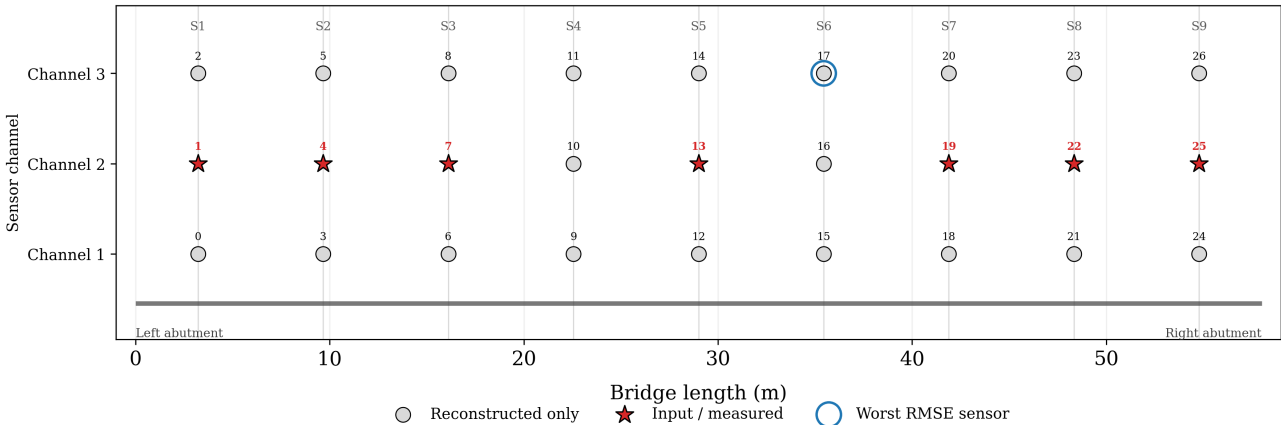


Figure 12. Sparse input sensor placement on the Z24 bridge layout. Red markers denote the measured input sensors provided to the model, while gray markers denote reconstructed sensor locations.

Figure 12 shows the locations of the selected input sensors and the sensors reconstructed by the proposed model on the Z24 bridge layout. Figure 13 illustrates the sensor-time distribution of the absolute reconstruction error for a representative test recording. Overall, the results demonstrate that SpectONet improves vibration-field reconstruction at unmeasured sensor locations compared with baselines. These findings provide preliminary evidence that the proposed framework can be extended beyond synthetic EBB simulations to real-world sparse-sensor structural vibration measurements.

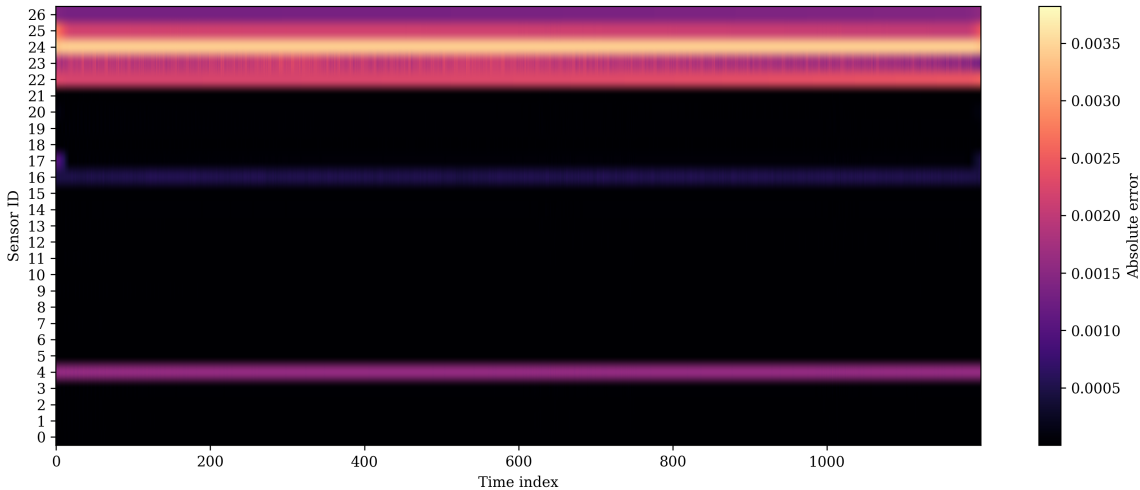


Figure 13. Absolute reconstruction error over 27 sensor locations.

8. Conclusion

Accurate prediction of structural vibration responses from sparse measurements remains a challenging task. Conventional operator-learning methods often employ uniformly distributed

sensors and may not adequately enforce the underlying physical laws. To address these limitations, the present study proposed the physics-guided Spectral Deep Operator Network (SpectONet), which integrates deterministic CGL sensor locations with PI-DeepONet.

The proposed model represents input functions using fewer sensors and incorporates the governing PDE, IC, and BC into the loss function. SpectONet was evaluated on three EBB vibration problems having different dynamics. Its real-world applicability was also examined using the Z24 bridge dataset. The main conclusions are summarized as follows:

- (1) SpectONet uses deterministic CGL sensor locations to represent input functions effectively with fewer sampling points.
- (2) The governing PDE, IC, and BC are incorporated into the loss function, thereby improving the physical consistency of the learned operator.
- (3) SpectONet solves diverse EBB vibration problems involving external forcing, damping effects, and spatially varying coefficients.
- (4) The Z24 bridge results demonstrate that SpectONet can reconstruct vibration responses at unmeasured locations from sparse experimental measurements.
- (5) SpectONet achieves an improvement of at least 64% over the baseline models across the three synthetic problems and at least 37% for the real-world problem.

Despite its promising performance, the present study has some limitations. The physics-guided numerical experiments are restricted to one-dimensional EBB vibration problems with deterministic CGL sensor locations. The effects of measurement noise, irregular sensor availability, uncertain material properties, strongly nonlinear structural behaviour, and complex multidimensional geometries have not been investigated here. Moreover, the CGL-inspired sensor selection used for the Z24 bridge depends on the available discrete sensor locations and does not necessarily represent a globally optimal sensor configuration. The real-world experiment should therefore be interpreted as an initial validation of the sparse-sensor reconstruction capability of the SpectONet framework.

Future work will extend SpectONet to Timoshenko beams, plates, shells, composite structures, nonlinear vibration problems, inverse analysis, damage detection, uncertainty quantification, and real-time SHM. Adaptive sensor placement, noise-robust training, transfer learning, and advanced neural operator architectures will also be explored in future.

Overall, SpectONet provides an accurate, efficient, and physically consistent operator-learning-based framework for research in structural vibration responses.

Appendix A.

This appendix presents the training procedures of the baseline models. For completeness and reproducibility, the implementation details of Vanilla DeepONet, PI-DeepONet, PINN, and CNN-UNet are summarized in Algorithms 3–5. These algorithms correspond to the implementations used for the comparative numerical experiments reported in Section 5. To distinguish the proposed framework from the competing methods, Table A.12 compares the loss components used during training.

Table A.12. Comparison of training objectives used by the proposed method and baseline models.

Model	PDE Residual loss	BC loss	IC loss	Supervised Loss
Vanilla-DeepONet	×	×	×	✓
CNN-UNet	×	×	×	✓
PINN	✓	✓	✓	×
PI-DeepONet	✓	✓	✓	✓
SpectONet (Ours)	✓	✓	✓	×

Appendix A.1. Vanilla Deep Operator Network

Algorithm 2 summarizes the training procedure of the Vanilla-DeepONet [19] baseline. Unlike PI-DeepONet, this model employs the same branch and trunk network architecture but is trained solely using supervised data without explicitly incorporating the governing PDE, ICs, or BCs into the loss function.

Algorithm 2: Training Procedure of Vanilla-DeepONet

```
1 Initialize branch-network parameters  $\theta_b$ , trunk-network parameters  $\theta_t$ , and set  $\theta = \{\theta_b, \theta_t\}$ ;
2 Construct uniformly distributed sensor locations  $\{x_j\}_{j=1}^{N_s}$  and form the input sensor vector
    $\mathbf{u} = [u(x_1), u(x_2), \dots, u(x_{N_s})]^T$ ;
3 Generate the training samples  $\{(\mathbf{u}, x_i, t_i, u_i)\}_{i=1}^N$ ;
4 Set convergence tolerance  $\varepsilon$ ;
5 Train the network using the Adam optimizer;
6 for each Adam iteration do
7   Compute branch features  $\mathbf{b} = \mathcal{B}_{\theta_b}(\mathbf{u})$ ;
8   Compute trunk features  $\mathbf{t} = \mathcal{T}_{\theta_t}(x, t)$ ;
9   Predict the solution using Eq (27);
10  Compute the total loss using Eq (28);
11  Update  $\theta$  using Adam;
12 Switch to the L-BFGS optimizer;
13 while  $\mathcal{L}(\theta) > \varepsilon$  do
14   Evaluate  $\mathcal{L}(\theta)$ ;
15   Update  $\theta$  using L-BFGS;
16 return Trained Vanilla-DeepONet model;
```

The trained Vanilla-DeepONet model is subsequently evaluated, and the corresponding prediction errors are reported for comparative analysis. The network architecture and training hyperparameters employed for the Vanilla-DeepONet model are summarized in Table A.13.

Table A.13. Network architecture and hyperparameters employed for the Vanilla-DeepONet model.

Parameter	P1	P2	P3
Activation function	Tanh	Tanh	Tanh
Learning rate	$2e - 03$	$2e - 03$	$2e - 03$
Adam iterations	500	1000	1000
L-BFGS iterations	5000	5000	5000
Evaluation grid	101×101	101×101	101×101
Branch network architecture	[12, 8, 8, 6]	[12, 15, 15, 15, 12]	[12, 43, 43, 27]
Trunk network architecture	[2, 8, 8, 6]	[2, 15, 15, 15, 12]	[2, 43, 43, 27]

Appendix A.2. Physics-informed Deep Neural Operator

Algorithm 3 summarizes the training procedure of the PI-DeepONet [32] baseline. The model employs branch and trunk networks to approximate the solution operator, while the governing PDE, IC, and BC are incorporated through a composite physics-informed loss function.

The trained PI-DeepONet model is subsequently used to predict the beam response over the prescribed space-time domain, and the corresponding error metrics are evaluated for comparison with the SpectONet framework. The network architectures and training hyperparameters adopted for the three benchmark problems are summarized in Table A.14.

Algorithm 3: Training Procedure of PI-DeepONet

```
1 Initialize branch-network parameters  $\theta_b$ , trunk-network parameters  $\theta_t$ , and set  $\theta = \{\theta_b, \theta_t\}$ ;
2 Construct uniformly distributed sensor locations  $\{x_j\}_{j=1}^{N_s}$  and form the input sensor vector
    $\mathbf{u} = [u(x_1), u(x_2), \dots, u(x_{N_s})]^T$ ;
3 Generate interior collocation points  $(x_i^r, t_i^r)$ , boundary points  $(x_i^b, t_i^b)$ , and initial points  $(x_i^0, 0)$ ;
4 Set convergence tolerance  $\varepsilon$ ;
5 Train the network using the Adam optimizer;
6 for each Adam iteration do
7   Compute branch features  $\mathbf{b} = \mathcal{B}_{\theta_b}(\mathbf{u})$ ;
8   Compute trunk features  $\mathbf{t} = \mathcal{T}_{\theta_t}(x, t)$ ;
9   Predict the solution in Eq. (43);
10  Compute the PDE residual  $R_\theta$  from Eq. (44) using AD;
11  Compute the residual loss  $\mathcal{L}_r$  using Eq. (45);
12  Compute the BCs loss  $\mathcal{L}_b$  using Eq. (46);
13  Compute the ICs loss  $\mathcal{L}_i$  using Eq. (47);
14  Compute the total loss according to Eq. (21);
15  Update  $\theta$  using Adam;
16 Switch to the L-BFGS optimizer;
17 while  $\mathcal{L}(\theta) > \varepsilon$  do
18   Evaluate  $\mathcal{L}(\theta)$ ;
19   Update  $\theta$  using L-BFGS;
20 return Trained PI-DeepONet model;
```

Table A.14. Network architecture and hyperparameters employed for the PI-DeepONet model.

Parameter	P1	P2	P3
Activation function	Tanh	Tanh	Tanh
Learning rate	$2e - 03$	$2e - 03$	$2e - 03$
Adam iterations	500	1000	1000
L-BFGS iterations	5000	5000	5000
Evaluation grid	101×101	101×101	101×101
Branch network architecture	[12, 8, 8, 6]	[12, 15, 15, 15, 12]	[12, 43, 43, 27]
Trunk network architecture	[2, 8, 8, 6]	[2, 15, 15, 15, 12]	[2, 43, 43, 27]
Loss weights	[1, 10, 10, 10, 10, 10, 10, 10, 10]	[1, 10, 10, 10, 10, 10, 10, 10, 10]	[1, 10, 10, 10, 10, 10, 10, 10, 10]

Appendix A.3. Physics-informed Neural Network

Algorithm 4 summarizes the training procedure of the PINN [11] baseline. The governing equation together with the ICs and BCs are directly incorporated into the loss function without employing an operator learning architecture.

Algorithm 4: Training Procedure of PINN

```
1 Generate interior collocation points  $(x_i^r, t_i^r)$ , boundary points  $(x_i^b, t_i^b)$ , and initial points  $(x_i^0, 0)$ ;
2 Set convergence tolerance  $\varepsilon$ ;
3 Train the network using the Adam optimizer;
4 for each Adam iteration do
5   Evaluate the governing PDE residual from Eq. (44) using AD;
6   Compute the residual loss using Eq (45);
7   Compute the ICs loss using Eq (47);
8   Compute the BCs loss using Eq (46);
9   Compute the composite physics-informed loss Eq (21);
10  Update the network parameters  $\theta$  using Adam;
11 Switch to the L-BFGS optimizer;
12 while  $\mathcal{L}(\theta) > \varepsilon$  do
13   Evaluate the composite loss  $\mathcal{L}(\theta)$ ;
14   Update  $\theta$  using L-BFGS;
15 return Trained PINN model;
```

The network architectures and training hyperparameters adopted for the three benchmark problems are summarized in Table A.15.

Table A.15. Network architecture and hyperparameters employed for the PINN model.

Parameter	P1	P2	P3
Activation function	Tanh	Tanh	Tanh
Learning rate	$1e - 03$	$2e - 03$	$2e - 03$
Adam iterations	500	500	500
L-BFGS iterations	5000	5000	5000
Evaluation grid	101×101	101×101	101×101
Network architecture	[2, 10, 10, 1]	[2, 24, 24, 1]	[2, 32, 32, 32, 32, 32, 1]
Loss weights	[1, 10, 10, 10, 10, 10, 10]	[1, 10, 10, 10, 10, 10, 10]	[1, 10, 10, 10, 10, 10, 10]

Appendix A.4. U-Net Convolutional Neural Network

Algorithm 5 summarizes the training procedure of the CNN-UNet [54] baseline. The model receives the space-time grid as input and learns the corresponding displacement field through supervised optimization using the reference solution data.

Algorithm 5: Training Procedure of CNN-UNet

```

1 Initialize the encoder and decoder parameters  $\theta$ ;
2 Construct the space-time input grid  $\mathbf{X} = (x, t)$ ;
3 Generate the corresponding reference solution field  $\mathbf{u}$ ;
4 Set the convergence tolerance  $\varepsilon$ ;
5 Train the network using the Adam optimizer;
6 for each Adam iteration do
7   Feed the input grid  $\mathbf{X}$  into the encoder;
8   Extract hierarchical latent feature maps at different resolution levels;
9   Pass the latent feature maps and skip connections through the decoder;
10  Reconstruct the displacement field  $\hat{u}_\theta(x, t)$ ;
11  Compute the total loss using Eq. (28);
12  Update the network parameters  $\theta$  using Adam;
13 Switch to the L-BFGS optimizer;
14 while  $\mathcal{L}(\theta) > \varepsilon$  do
15   Evaluate the supervised training loss;
16   Update the network parameters  $\theta$  using L-BFGS;
17 return Trained CNN-UNet model;
```

Table A.16 summarizes the network architecture and training hyperparameters employed for the CNN-UNet baseline in the three benchmark problems.

Table A.16. Network architecture and hyperparameters employed for the CNN-UNet model.

Parameter	P1	P2	P3
Activation function	Tanh	Tanh	Tanh
Learning rate	$1e - 03$	$2e - 03$	$2e - 03$
Adam iterations	500	500	500
L-BFGS iterations	5000	5000	5000
Evaluation grid	101×101	101×101	101×101
Input channels	2	2	2
Output channels	1	1	1
Base channels	2	4	8

Conflict of Interest Statement

The authors confirm that they have no conflicts of interest related to this paper.

Funding Declaration

The authors confirm that they did not receive any funding to carry out this work.

Authors' Contributions

Shivani Saini: Conceptualization, Methodology, Visualization, Software, Experiment, Writing-original draft & editing. **Ramesh Kumar Vats:** Supervision, Investigation, Formal analysis, Validation, Writing-review & editing. **Arup Kumar Sahoo:** Software, Methodology, Validation, Writing-original draft

Data Availability

The processed Z24 Bridge dataset used in this study is publicly available from the Hugging Face repository: [thanglexuan/Z24-dataset-processed](https://huggingface.co/thanglexuan/Z24-dataset-processed).

Code Availability

The implementation of the proposed framework is publicly available at: <https://github.com/SM-a-RT-Lab/SpectONet>.

References

- [1] S. Chakraverty, A. K. Sahoo, D. Mohapatra, Artificial Neural Networks and Type-2 Fuzzy Set: Elements of Soft Computing and Its Applications, Elsevier, 2025.
- [2] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep learning, Vol. 1, MIT press Cambridge, 2016.
- [3] U. Oparaji, R. J. Sheu, M. Bankhead, J. Austin, E. Patelli, Robust artificial neural network for reliability and sensitivity analyses of complex non-linear systems, Neural Networks 96 (2017) 80–90.
- [4] S. Saini, R. K. Vats, A. K. Sahoo, A-PINN: Auxiliary physics-informed neural networks for structural vibration analysis in continuous Euler-Bernoulli beam, Applied Soft Computing (2026) 115852.
- [5] Y. Kumar, R. K. Vats, A deep Kolmogorov-Arnold Network framework for solving time-fractional partial differential equations, Applied Soft Computing (2025) 114379.
- [6] A. Gartia, S. Chakraverty, Advanced computational modeling and mechanical behavior analysis of multi-directional functionally graded nanostructures: a comprehensive review, Computer Modeling in Engineering & Sciences 142 (3) (2025) 2405.
- [7] O. Baysal, A. Hasanov, A. Kawano, Exponential stability of damped Euler-Bernoulli beam controlled by boundary springs and dampers, Journal of Mathematical Analysis and Applications 533 (2) (2024) 128031.
- [8] S. Chakraverty, Vibration of plates, CRC press, 2008.
- [9] I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE transactions on neural networks 9 (5) (1998) 987–1000.
- [10] Y. Kumar, R. K. Vats, Deep wavelet neural network with lateral connections for solving fractional differential equations arising in oil pollution dynamics, International Journal of Computer Mathematics (2026) 1–43.
- [11] M. Raissi, P. Perdikaris, 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 (2019) 686–707.
- [12] S. Kumar, A. K. Sahoo, S. Chakraverty, Physics-informed machine learning framework for approximating the modified Degasperis-Procesi equation, in: 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIIE), IEEE, 2023, pp. 1–6.
- [13] Y. Kumar, R. Kumar Vats, A. Kumar Nain, S. Yadav, Robust deep learning framework using Hermite interpolation for time-dependent caputo fractional differential equations, Mathematical Methods in the Applied Sciences 49 (7) (2026) 6398–6427.

- [14] S. Yadav, R. K. Vats, A. Rao, Application of extended residual power series method for time-fractional Zakharov–Kuznetsov equations in ocean-based coastal wave, *Pramana* 99 (3) (2025) 97.
- [15] T. Kapoor, H. Wang, A. Núñez, R. Dollevoet, Transfer learning for improved generalizability in causal physics-informed neural networks for beam simulations, *Engineering Applications of Artificial Intelligence* 133 (2024) 108085.
- [16] A. K. Sahoo, S. Kumar, S. Chakraverty, Application of physics-informed neural networks for simulating mass-spring-damper systems (2024).
- [17] J. Song, K. Kawaguchi, Z. Yan, PMNO: A novel physics guided multi-step neural operator predictor for partial differential equations, *Neural Networks* (2026) 108593.
- [18] V. Kumar, S. Goswami, K. Kontolati, M. D. Shields, G. E. Karniadakis, Synergistic learning with multi-task deeponet for efficient PDE problem solving, *Neural Networks* 184 (2025) 107113.
- [19] L. Lu, P. Jin, G. Pang, Z. Zhang, G. E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, *Nature machine intelligence* 3 (3) (2021) 218–229.
- [20] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, *arXiv preprint arXiv:2010.08895* (2020).
- [21] T. Tripura, S. Chakraborty, Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems, *Computer Methods in Applied Mechanics and Engineering* 404 (2023) 115783.
- [22] Q. Cao, S. Goswami, G. E. Karniadakis, Laplace neural operator for solving differential equations, *Nature Machine Intelligence* 6 (6) (2024) 631–640.
- [23] Y. Liu, Y. Chen, R. Liu, R. Li, X. Wang, Z. Peng, K. Tan, Q. Chen, Physics-encoded convolutional neural operators for parametric PDEs: A convergence-guaranteed framework via pre-computed kernel fields, *Neural Networks* (2026) 109309.
- [24] A. Anandkumar, K. Azizzadenesheli, K. Bhattacharya, N. Kovachki, Z. Li, B. Liu, A. Stuart, Neural operator: Graph kernel network for partial differential equations, in: *ICLR 2020 workshop on integration of deep neural models and differential equations*, 2020.
- [25] Y. Guo, J. Song, X. Cao, C. Zhao, H. Leng, F. Yin, PGMNO: A physics-guided mamba neural operator framework for partial differential equations, *Neural Networks* (2026) 108845.
- [26] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed deeponets, *Science advances* 7 (40) (2021) eabi8605.
- [27] L. N. Trefethen, *Spectral methods in MATLAB*, SIAM, 2000.
- [28] M. H. Heydari, Z. Avazzadeh, Chebyshev-Gauss-Lobatto collocation method for variable-order time fractional generalized Hirota–Satsuma coupled KdV system, *Engineering with Computers* 38 (2) (2022) 1835–1844.
- [29] D. A. Kopriva, *Implementing spectral methods for partial differential equations: Algorithms for scientists and engineers*, Springer Science & Business Media, 2009.
- [30] S. Goswami, A. Bora, Y. Yu, G. E. Karniadakis, Physics-informed deep neural operator networks, in: *Machine learning in modeling and simulation: methods and applications*, Springer, 2023, pp. 219–254.
- [31] L. Mandl, S. Goswami, L. Lambers, T. Ricken, Separable physics-informed DeepONet: Breaking the curse of dimensionality in physics-informed machine learning, *Computer Methods in Applied Mechanics and Engineering* 434 (2025) 117586.
- [32] A. Cong, S. Wang, Y. Jin, H. Yi, Z. Chang, Y. Jiang, L. Hou, N. A. Saeed, Respecting causality segmental time-domain training accelerates physics-informed DeepONet on time-dependent PDEs, *Neurocomputing* (2026) 133915.

- [33] K. Michałowska, S. Goswami, G. E. Karniadakis, S. Riemer-Sørensen, Neural operator learning for long-time integration in dynamical systems with recurrent neural networks, in: 2024 International joint conference on neural networks (IJCNN), IEEE, 2024, pp. 1–8.
- [34] B. Yang, X. Li, J. Zhao, Y. Jiang, DD-DeepONet: A domain decomposition DeepONet framework for PDEs on structured domains with geometry and physics variations, *Engineering Analysis with Boundary Elements* 187 (2026) 106727.
- [35] S. Karumuri, L. Graham-Brady, S. Goswami, Physics-informed latent neural operator for real-time predictions of time-dependent parametric PDEs, *Computer Methods in Applied Mechanics and Engineering* 450 (2026) 118599.
- [36] S. Sarthak, D. Srinivasacharya, Physics-informed deep operator learning for long-time simulation of nonlinear partial differential equations via temporal domain decomposition, *Engineering Applications of Artificial Intelligence* 171 (2026) 114261.
- [37] S. Garg, S. Chakraborty, Neuroscience inspired neural operator for partial differential equations, *Journal of Computational Physics* 515 (2024) 113266.
- [38] S. Garg, S. Chakraborty, VB-DeepONet: A bayesian operator learning framework for uncertainty quantification, *Engineering Applications of Artificial Intelligence* 118 (2023) 105685.
- [39] Z. Zhao, C. Liu, Y. Li, Z. Chen, X. Liu, Diffeomorphism neural operator for various domains and parameters of partial differential equations, *Communications Physics* 8 (1) (2025) 15.
- [40] J. He, S. Koric, D. Abueidda, A. Najafi, I. Jasiuk, Geom-DeepONet: A point-cloud-based deep operator network for field predictions on 3d parameterized geometries, *Computer Methods in Applied Mechanics and Engineering* 429 (2024) 117130.
- [41] W. Li, M. Z. Bazant, J. Zhu, Phase-Field DeepONet: Physics-informed deep operator neural network for fast simulations of pattern formation governed by gradient flows of free-energy functionals, *Computer Methods in Applied Mechanics and Engineering* 416 (2023) 116299.
- [42] M. Lu, A. Mohammadi, Z. Meng, X. Meng, G. Li, Z. Li, Deep neural operator for learning transient response of interpenetrating phase composites subject to dynamic loading, *Computational Mechanics* 72 (3) (2023) 563–576.
- [43] J. van Delden, J. Schultz, C. Blech, S. C. Langer, T. Lüddecke, Learning to predict structural vibrations, *Advances in Neural Information Processing Systems* 37 (2024) 103706–103736.
- [44] B. Ahmed, Y. Qiu, D. W. Abueidda, W. El-Sekelly, B. G. de Soto, T. Abdoun, M. E. Mobasher, Physics-informed deep operator networks with stiffness-based loss functions for structural response prediction, *Engineering Applications of Artificial Intelligence* 144 (2025) 110097.
- [45] J. He, S. Kushwaha, J. Park, S. Koric, D. Abueidda, I. Jasiuk, Predictions of transient vector solution fields with sequential deep operator network, *Acta Mechanica* 235 (8) (2024) 5257–5272.
- [46] E. Liu, S. HEDAYATRASAA, Y. Zhu, L. Cheng, A case study on vibration analysis using DeepONet, in: 11th European Workshop on Structural Health Monitoring: EWSHM 2024, Media and Publishing Partner, 2024.
- [47] R. Wan, E. Kharazmi, M. S. Triantafyllou, G. E. Karniadakis, DeepVIVONet: Using deep neural operators to optimize the location of sensors with application to vortex-induced vibrations, *Journal of Fluids and Structures* 137 (2025) 104330.
- [48] X. Tian, C. Wan, F. Duan, D. Wu, L. Chang, X. Nie, Physics-informed deep operator network for bending analysis of stiffened panels, *Thin-Walled Structures* (2026) 114765.
- [49] S. S. Rao, *Vibration of Continuous Systems*, John Wiley & Sons, Hoboken, New Jersey, 2019.

- [50] M. D. Aouragh, Y. Khali, S. Khallouq, M. Segaoui, Compact finite difference scheme for Euler-Bernoulli beam equation with a simply supported boundary conditions, *International Journal of Applied and Computational Mathematics* 10 (1) (2024) 17.
- [51] N. Ketkar, J. Moolayil, Automatic differentiation in deep learning, in: *Deep Learning with python: learn best practices of deep learning models with PyTorch*, Springer, 2021, pp. 133–145.
- [52] G. I. El-Baghdady, M. El-Azab, Chebyshev-Gauss-Lobatto pseudo-spectral method for one-dimensional advection-diffusion equation with variable coefficients, *Math* 3 (1) (2016) 1–8.
- [53] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, *Journal of machine learning research* 18 (153) (2018) 1–43.
- [54] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.
- [55] J. Maeck, G. De Roeck, Description of Z24 benchmark, *Mechanical Systems and Signal Processing* 17 (1) (2003) 127–131.
- [56] G. Steenackers, P. Guillaume, Structural health monitoring of the Z24 bridge in presence of environmental changes using modal analysis, in: *Proceedings of IMAC*, Vol. 23, 2005.