C.A. Yan, R. Vescovini, L. Dozio
A Framework Based on Physics-Informed Neural Networks and Extreme Learning for the Analysis of Composite Structures
Computers & Structures, Vol. 265, 2022, 106761 (20 pages)
doi:10.1016/j.compstruc.2022.106761

The final publication is available at https://doi.org/10.1016/j.compstruc.2022.106761

A Framework based on Physics-Informed Neural Networks and Extreme Learning for the Analysis of Composite Structures

C.A. Yan* R. Vescovini and L. Dozio

Dipartimento di Scienze e Tecnologie Aerospaziali, Politecnico di Milano

Via La Masa 34, 20156 Milano, Italy

Abstract

This paper presents a novel approach for solving direct problems in linear elasticity involving plate and shell structures. The method relies upon a combination of Physics-Informed Neural Networks and Extreme Learning Machine. A subdomain decomposition method is proposed as a viable mean for studying structures composed by multiple plate/shell elements, as well as improving the solution in domains composed by one single element. Sensitivity studies are presented to gather insight into the effects of different network configurations and sets of hyperparameters. Within the framework presented here, direct problems can be solved with or without available sampled data. In addition, the approach can be extended to the solution of inverse problems. The results are compared with exact elasticity solutions and finite element calculations, illustrating the potential of the approach as an effective mean for addressing a wide class of problems in structural mechanics.

Keywords: Physics-Informed Neural Networks; Extreme Learning Machine; Structural analysis; Shell structures.

1 Introduction

The standard modeling approach in the field of solid mechanics is based on the definition of governing equations starting from balance principles. The solution is generally sought using analytical or computational techniques, such as the Finite Element Method (FEM)

*Corresponding author. *Email address:* chengangelo.yan@polimi.it (Cheng Angelo Yan),

[1] and meshfree methods [2]. The resulting computational models can be referred to as physics-based, as they are completely defined by the set of equations arising from the relevant principles. However, physics-based modeling is not applicable whenever the lack of an exhaustive understanding of the problem does not allow the problem to be formulated in terms of governing equations. An alternative paradigm is represented by data-driven techniques, which are typically based on Machine Learning (ML) methods. In this case, the model is constructed by fitting large volumes of data representing the behavior of the system under investigation. This approach allows the main features of the process to be modeled even if the governing equations are not available. In this context, Artificial Neural Networks (ANNs) represent one of the most popular ML methods for building data-driven models. This is even more true due to recent progresses in this field (see, e.g., Deep Learning [3]) and availability of new advanced algorithms for ML, such as Automatic Differentiation [4]. In the field of solid mechanics, several applications have been proposed. Petrolo and Carrera [5] suggested the use of ANNs as an effective mean for selecting the element kinematics in finite element meshes. A finite element zooming technique where boundary conditions are estimated using neural networks is presented in Ref. [6]. Another example is the work of Liu et al. [7], where neural networks are used for modeling complex nonlinear constitutive laws and for predicting damage accumulation in composite materials. Data-driven models based on ANNs were developed in Ref. [8] to be adopted as surrogates for accelerating the stress analysis of isotropic elastic structures with different geometries. Bisagni and Lanzi [9] trained ANNs with data coming from nonlinear FE analysis for a post-buckling optimization procedure of composite stiffened panels loaded in compression. In the above-mentioned works, ANNs are used as a "black-box", whose effectiveness is highly dependent on the available data, both qualitatively and quantitatively. In many real-world applications, however, data can be scarce and/or very expensive to be generated. Hybrid approaches, based on the idea of combining the mathematical model with available data, are a promising way for developing reliable and accurate models. Several strategies have been proposed for building such models in the framework of ML methods [10]. One possibility is represented by Physics-Informed Neural Networks (PINNs) [11], a novel ML paradigm consisting in incorporating the available governing equations in a Deep Learning framework. This approach allows the "black-box" neural network to be enriched with information available on the underlying

physical laws. The result is a "gray-box" approach: the learning process is physics-oriented and so the requirements on training data can be relaxed. The idea of PINNs was first proposed by Raissi et al. [11], who demonstrated the effectiveness of PINNs for solving and discovering partial differential equations. Other recent studies in the field are due to Zhang et al. [12], addressing the solution of stochastic differential problems, and Haghighat et al. [13], solving the PDE corresponding to the linear equilibrium of elastic bodies. A neural network-based plasticity model embedding thermodinamical consistency as a physical constraint is illustrated in Ref. [14]. Recent contributions focused on the improvement of PINNs learning performance. A variational form of PINNs is proposed in Refs. [15] and [16], the main advantage consisting in faster training due to the reduced order of the derivatives entering the loss function. Adaptive activation functions were employed by Jagtap et al. [17]. The authors concluded that a progressive scaling of the activation function during the training process can improve the convergence rate and the accuracy of PINNs. Similarly, an adaptive method was proposed in Ref. [18] for selecting training points for PINNs with a criterion based on the loss function value. Better robustness of the training process was achieved, especially for problems characterized by non-smooth solutions. Domain decomposition approaches were integrated in the PINN framework in Refs. [19] and [20]. A division of the computational domain into subregions and the combined use of multiple subnetworks were found to lead to improved representation performance, more efficient hyperparameter tuning and the possibility of representing steep gradients or discontinuities. A crucial aspect in the development of PINNs regards the time for training. One attempt to obtain more efficient strategies is found in Refs. [21] and [22]. They exploited the concept of Transfer Learning, obtaining improved computational efficiencies starting the training process from a pretrained state holding an initial knowledge of the problem. Extreme Learning Machine (ELM) [23] is another learning algorithm that has been successfully applied aiming at reducing the computational cost for training, without affecting the performance of PINNs. In ELM, the weights of the hidden layer are generated randomly and do not need to be learnt, with a clear advantage in terms of learning speed. Successful applications of ELM within PINN frameworks are found in Refs. [20, 24].

Starting from the idea pursued in these two references, this paper aims at presenting, for the first time, a PINN/ELM-based approach for the direct and inverse solution of linear

elasticity problems. A formulation is presented, which is capable of handling the analysis of composite thin-walled structures. Past efforts have focused on applications with relatively simple domains. The proposed approach brings the application of PINN/ELM one step further by developing a domain decomposition strategy as a viable mean for studying assemblies of plate and shell-like structural elements. Exemplary test cases are presented to illustrate the potentials of this strategy and its use with or without labeled data available. The paper is organized as follows: the governing PDEs expressing the shell mathematical model to be used during the training process of PINNs are derived in Section 2; Section 3 is devoted to the description of the PINN framework and the learning procedures adopted in this work; the results are presented in Section 4, where a comparison against reference solutions is shown for validation purposes, along with a set of parametric studies on the networks hyperparameters. Two practical applications are then illustrated regarding the static analysis of an isotropic panel with a cutout and the free vibration analysis of a stiffened composite panel. A final example is devoted to the application of the method to solve an inverse problem, where the stacking sequence of a variable-stiffness plate is identified for a target static response.

2 Formulation

The starting point of PINNs is the definition of the relevant physical laws governing the problem at hand. While the overall framework of PINNs offers a wide range of applications – meaningful examples are found in the field of fluid dynamics, quantum mechanics, solid mechanics [11, 13] –, the present work aims at presenting their use in the context of linear elasticity. Specifically, the class of problems considered here encompasses static, free vibration and buckling analysis of thin plates and shells, as well as assemblies of them. Donnell theory [25] is used as underlying theory along with the assumptions of linear elastic material response. Composite structures are considered and the elastic properties are allowed to vary along the in-plane directions, so variable-stiffness (VS) configurations, see e.g. [26, 27, 28, 29, 30], can be studied.

This preliminary section aims at presenting the governing equations to be used later to *inform* the neural network in the training process. Despite the possibility of adopting energy

formulations relying upon a variational principle [16], the mathematical model is expressed here in terms of Partial Differential Equations (PDEs). So, a strong-form approach is presented.

It is worth highlighting that the neural networks presented in this work are informed with mathematical physics models and do not necessarily need to be supplemented with data. In spite of this, the acronym PINN is retained for consistency with the earliest works in the literature [11], although the latest definitions of physics-informed machine learning mention the seamless integration of data and mathematical physics models, even in partially understood, uncertain and high-dimensional contexts.

A sketch of a cylindrical shell is reported in Figure 1, where R denotes the radius of curvature, a and b are the dimensions along the axial and the circumferential directions, respectively, and t is the thickness. An orthogonal curvilinear coordinate system is taken in correspondence of the midsurface, with x, y and z axis running as illustrated in the sketch. The vectors e_n, e_t and e_r are directed along the shell edges' normal, tangential and radial directions, respectively.

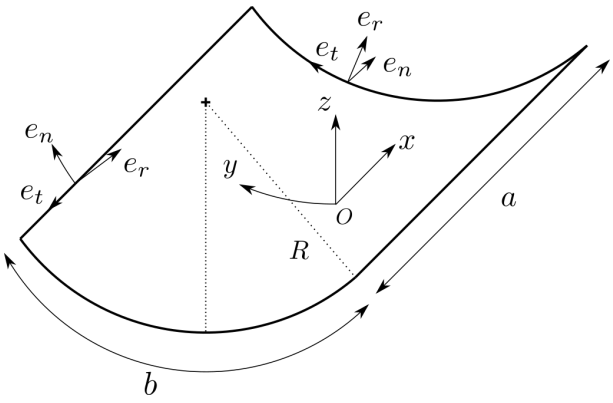


Figure 1: Shell geometry and reference system.

The equation expressing the shell static/dynamic equilibrium are [31]:

$$\begin{cases} N_{xx,x} + N_{xy,y} + \beta_1 q_x + \beta_2 \left(-I_0 \ddot{u} + I_1 \ddot{w}_{,x}\right) = 0 \\ N_{xy,x} + N_{yy,y} + \beta_1 q_y + \beta_2 \left(-I_0 \ddot{v} + I_1 \ddot{w}_{,y}\right) = 0 \\ M_{xx,xx} + 2M_{xy,xy} + M_{yy,yy} + N_{yy}/R + \beta_1 q_z + \beta_2 \left[-I_0 \ddot{w} + I_2 \left(\ddot{w}_{,xx} + \ddot{w}_{,yy}\right) - I_1 \left(\ddot{u}_{,x} + \ddot{v}_{,y}\right)\right] + \\ + \beta_3 \left(\overline{N}_{xx} w_{,xx} + 2\overline{N}_{xy} w_{,xy} + \overline{N}_{yy} w_{,yy}\right) = 0 \end{cases} \tag{1}$$

where a comma followed by an index denotes partial differentiation with respect to that index, while dot defines the time derivative. Use is made of the Boolean flags β_i [32, 33], whose values lead to different interpretations of Eq. (1). Denoting with δ_{ik} the Kronecker's delta, static equilibrium equations are obtained by taking $\beta_i = \delta_{i1}$; the dynamic equilibrium with no external loads ($q_i = 0$) is expressed by the equations obtained with $\beta_i = \delta_{i2}$; buckling equations are available by setting $\beta_i = \delta_{i3}$, where $\overline{N}_{ij}$ are the pre-buckling resultants and all the other terms have to be understood as variations with respect to the pre-buckling equilibrium configuration.

The terms N_{ij} and M_{ij} are the force and moment resultants, defined as:

$$N_{ij} = \int_{-t/2}^{t/2} \sigma_{ij} \, \mathrm{d}z \quad \text{and} \quad M_{ij} = \int_{-t/2}^{t/2} \sigma_{ij} z \, \mathrm{d}z \quad i,j = x,y \tag{2}$$

where σ_{ij} are the components of the stress tensor.

The mass properties are specified by integrating the density ρ along the thickness as:

$$I_i = \int_{-t/2}^{t/2} z^i \rho \, \mathrm{d}z \quad i = 0,1,2 \tag{3}$$

The force and moment resultants appearing in Eq. (1) can be related to the middle surface displacement components by means of the strain-displacement relation and the material constitutive law.

In the context of Donnell thin shell theory, the relation between strains and displacements is expressed as:

$$\begin{Bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{Bmatrix} = \begin{Bmatrix} u_{,x} \\ v_{,y} - w/R \\ u_{,y} + v_{,x} \end{Bmatrix} + z \begin{Bmatrix} -w_{,xx} \\ -w_{,yy} \\ -2w_{,xy} \end{Bmatrix} = \boldsymbol{\epsilon}_0 + z\mathbf{k} \tag{4}$$

where ϵ_{xx}, ϵ_{yy} and γ_{xy} are the in-plane components of the strain tensor, while u, v and w are the three midplane displacement components along the directions x, y and z, respectively. The constitutive law reads:

$$\begin{Bmatrix} \mathbf{N} \\ \mathbf{M} \end{Bmatrix} = \begin{bmatrix} \mathbf{A}(x,y) & \mathbf{B}(x,y) \\ \mathbf{B}(x,y) & \mathbf{D}(x,y) \end{bmatrix} \begin{Bmatrix} \boldsymbol{\epsilon}_0 \\ \mathbf{k} \end{Bmatrix} \tag{5}$$

where $\mathbf{N} = \{N_{xx} \ N_{yy} \ N_{xy}\}^{\mathrm{T}}$ and $\mathbf{M} = \{M_{xx} \ M_{yy} \ M_{xy}\}^{\mathrm{T}}$ are the vectors collecting the force and moment resultants, whereas $\mathbf{A}$, $\mathbf{D}$ and $\mathbf{B}$ are the membrane, bending and membrane-bending coupling stiffness matrices, respectively, available from Classical Lamination Theory [34]. The constitutive law does not depend on the planar coordinates (x, y) in the case of isotropic or composite materials with straight fibers. On the contrary, this dependence arises in the case of variable-stiffness configurations, where fiber orientation θ is represented via Lagrange polynomials as [28]:

$$\theta\left(x,y\right) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} T_{mn} \prod_{n \neq i} \frac{\left(x - x_i\right)}{\left(x_n - x_i\right)} \cdot \prod_{m \neq j} \frac{\left(y - y_j\right)}{\left(y_m - y_j\right)} \tag{6}$$

where θ is interpolated starting from assigned values in a $M \times N$ grid of points whose coordinates are (x_r, y_s), with $r = \{i, n\}$ and $s = \{j, m\}$. The coefficients of the Lagrangian polynomials T_{mn} are uniquely determined by imposing $\theta\left(x_n, y_m\right) = T_{mn}$ at each reference point. Hence, the lamination sequence is defined through a matrix $\mathbf{T} \in \mathbb{R}^{M \times N}$, as shown in Figure 2.

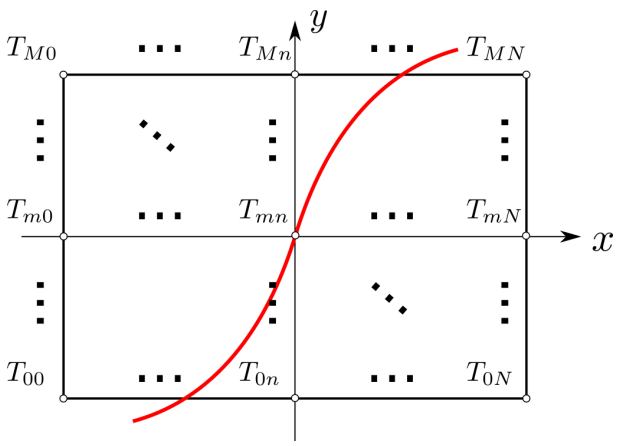


Figure 2: Fiber path definition via Lagrange polynomials.

The equilibrium conditions reported in Eq. (1) can be expressed in terms of middle surface displacement components upon substitution of Eqs. (4) and (5) in Eq. (1), resulting in:

$$\boldsymbol{\mathcal{R}} := \left(\boldsymbol{\mathcal{K}} - \beta_2\omega^2\boldsymbol{\mathcal{M}} + \beta_3\lambda\boldsymbol{\mathcal{G}}\right)\mathbf{u} + \beta_1\mathbf{q} = \mathbf{0} \quad \text{in } \Omega \tag{7}$$

where $\boldsymbol{\mathcal{R}}$ is the vector of the residual functions, $\boldsymbol{\mathcal{K}}$, $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{G}}$ are matrices of differential operators as defined in the Appendix; the scalar ω defines the vibration frequency, while λ is the buckling multiplier; the vectors $\mathbf{u} = \{u\ v\ w\}^{\mathrm{T}}$ and $\mathbf{q} = \{q_x\ q_y\ q_z\}^{\mathrm{T}}$ collect the displacements and surface loads, respectively.

The combination of Eq. (7) along with the approximation of the displacement functions via neural networks is the first step to build the PINN. In other words, Eq. (7) provides the information regarding the mathematical model to be accounted for by the neural network. The whole definition of the problem is completed by specifying the boundary conditions. Referring to Figure 1 and denoting the boundary with $\partial\Omega$, the conditions are:

$$\boldsymbol{\mathcal{B}} = \begin{cases} u_n - \hat{u}_n = 0 & \text{or} \quad N_{nn} - \hat{N}_{nn} = 0 \\ u_t - \hat{u}_t = 0 & \text{or} \quad N_{nt} - \hat{N}_{nt} = 0 \\ w_n - \hat{w}_n = 0 & \text{or} \quad V_n - \hat{V}_n = 0 \\ w_{n,n} - \hat{w}_{n,n} = 0 & \text{or} \quad M_{nn} - \hat{M}_{nn} = 0 \end{cases} \quad \text{in } \partial\Omega \tag{8}$$

where the caret defines any prescribed quantity, either in terms of forces or displacements, and:

$$\begin{cases} u_n = n_x u + n_y v \\ u_t = n_x v - n_y u \\ w_n = w \\ w_{n,n} = n_x w_{,x} + n_y w_{,y} \end{cases} \quad \text{and} \quad \begin{cases} N_{nn} = n_x^2 N_{xx} + n_y^2 N_{yy} + 2n_x n_y N_{xy} \\ N_{nt} = n_x n_y \left(N_{yy} - N_{xx}\right) + \left(n_x^2 - n_y^2\right) N_{xy} \\ V_n = n_x V_x + n_y V_y \\ M_{nn} = n_x^2 M_{xx} + n_y^2 M_{yy} + 2n_x n_y M_{xy} \end{cases} \tag{9}$$

where $V_x = M_{xx,x} + 2M_{xy,y}$ and $V_y = M_{yy} + 2M_{xy,x}$ are the Kirchhoff shear forces, n_x and n_y the components of the unitary vector e_n normal to the boundary $\partial\Omega$.

The set of equations (7) and (8) can be used for analyzing single-domain structures. A further extension is needed when assemblies of plate and shell elements are of concern. Specifically, the mathematical model is rephrased to account for the equilibrium and boundary

conditions of each single element and to consider the natural and essential conditions at the interfaces between elements. The whole set of conditions expressing the differential problem is then:

$$\begin{cases} \boldsymbol{\mathcal{R}}^{(p)} = 0 \quad \text{in } \Omega^{(p)} \\ \boldsymbol{\mathcal{B}}^{(p)} = 0 \quad \text{in } \partial\Omega^{(p)} \end{cases} \quad \text{for } p = 1...P$$

$$\begin{cases} \boldsymbol{\mathcal{I}}^{(q)}_{\text{con}} = 0 \\ \boldsymbol{\mathcal{I}}^{(q)}_{\text{equ}} = 0 \end{cases} \quad \text{in } \partial\Omega^{(q)}_{\text{int}} \quad \text{for } q = 1...Q \tag{10}$$

where P is the total number of elements composing the structure, while Q is the number of interfaces between elements. The last two sets of equations above specify the compatibility of displacements and the equilibrium conditions at the interfaces. The operators $\boldsymbol{\mathcal{I}}^{(q)}_{\text{con}}$ and $\boldsymbol{\mathcal{I}}^{(q)}_{\text{equ}}$ are defined as follows:

$$\boldsymbol{\mathcal{I}}^{(q)}_{\text{con}} = \begin{cases} u^{(i)}_n + u^{(j)}_n \cos\alpha^{(i)(j)} - w^{(j)}_n \sin\alpha^{(i)(j)} = 0 \\ u^{(i)}_t + u^{(j)}_t = 0 \\ w^{(i)}_n - w^{(j)}_n \cos\alpha^{(i)(j)} - u^{(j)}_n \sin\alpha^{(i)(j)} = 0 \\ w^{(i)}_{n,n} + w^{(j)}_{n,n} = 0 \end{cases} \quad \text{for } j = 1...J,\ j \neq i$$

$$\boldsymbol{\mathcal{I}}^{(q)}_{\text{equ}} = \begin{cases} N^{(i)}_{nn} - \sum_{j\neq i}^{J} \left(N^{(j)}_{nn} \cos\alpha^{(i)(j)} - V^{(j)}_n \sin\alpha^{(i)(j)} \right) = 0 \\ N^{(i)}_{nt} - \sum_{j\neq i}^{J} N^{(j)}_{nt} = 0 \\ V^{(i)}_n + \sum_{j\neq i}^{J} \left(V^{(j)}_n \cos\alpha^{(i)(j)} + N^{(j)}_{nn} \sin\alpha^{(i)(j)} \right) = 0 \\ M^{(i)}_{nn} - \sum_{j\neq i}^{J} M^{(j)}_{nn} = 0 \end{cases} \tag{11}$$

where J is the number of elements joining each other at $\partial\Omega^{(q)}_{\text{int}}$, and $\alpha^{(i)(j)}$ denotes the relative orientation between the interface elements i and j. The convention for defining the positive sense of the rotations is illustrated in Figure 3.

3 Solution via Physics-Informed Neural Networks

The system of PDEs defined by Eq. (10) represents the mathematical model of the structure. Unfortunately, analytical solutions can be hardly found unless specific and simplifying

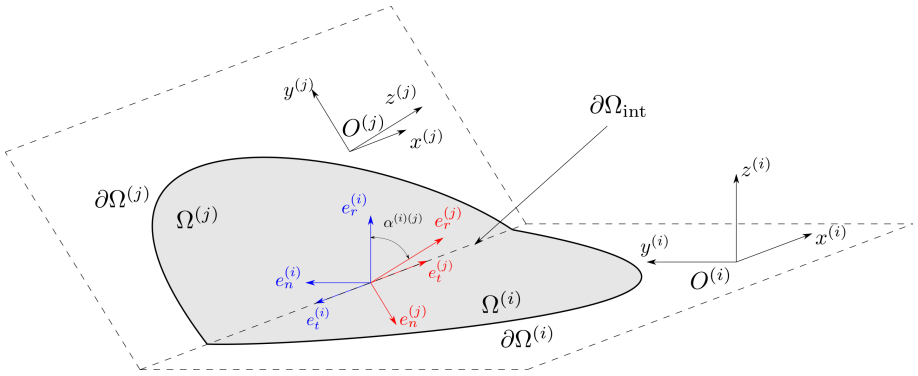


Figure 3: Adjacent plate/shell elements: convention for the relative rotation.

assumptions are introduced. Semi-analytical and numerical solution strategies are, in most cases, the only viable approach. Galerkin [31] and its modified version [35] are well-known examples of solution strategies belonging to the first class, while numerical strong-form solutions have been proposed in the literature using methods such as the Differential Quadrature Method (DQM) [36] and its generalized version (GQM) [37].

An alternative and still relatively unexplored strategy relies on methods based on Artificial Neural Networks (ANNs), see e.g [38, 39, 40, 41, 11, 42]. The advantages of adopting ANNs as ansatz of the solution are manifold. Firstly, the solution approximated by ANNs will inevitably inherit their generalization properties, which are known to be universal [43]. Secondly, the absence of a dependency on a computational mesh makes the handling of complex geometries straightforward. Indeed, only a set of training points, provided they are appropriately sampled, is required for the definition of the computational domain.

In the following section, Physics-Informed Neural Networks (PINNs) [11], which can be viewed as an ANN-based method for solving PDEs, are introduced for the solution of the differential problem presented earlier. Firstly, preliminary information regarding ANNs is presented. The underlying concept of PINNs is then illustrated along with relevant aspects regarding their implementation and training. With this purpose in mind, two different learning strategies are proposed. The first one, Gradient-based Learning (GBL), is classically employed in ML; the second, Extreme Learning Machine (ELM), is a relatively new strategy offering huge potential to guarantee faster training yet accurate solutions.

3.1 Preliminaries

Artificial Neural Networks (ANNs) are mathematical models composed by simple computational units, called neurons, which are interconnected each other in a layer-like structure [44], as depicted in Figure 4. In feedforward ANNs, information flows in one direction, i.e.

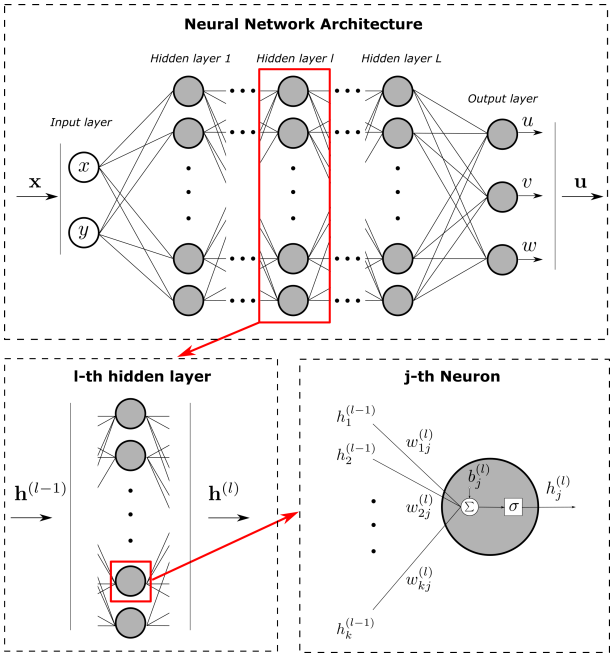


Figure 4: Neural Network: architecture, layers and neurons.

from the input layer to the output one. In this data stream, the input vector $\mathbf{x}$ undergoes a series of transformation, including multiplication by weighting factors, summation to given biases and generic nonlinear operators defined by the activation function of the neurons. Referring to an ANN with an arbitrary number of hidden layers L, the output vector $\mathbf{u}$ due to the input $\mathbf{x}$ can be defined as follows:

$$\begin{aligned} \mathbf{u} &= \mathbf{C}\mathbf{h}^{(L)} \\ \mathbf{h}^{(0)} &= \mathbf{x} \\ \mathbf{h}^{(l)} &= \sigma^{(l)}\left(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right) \quad \text{for } l = 1, 2, ..., L \end{aligned} \tag{12}$$

where $\sigma^{(l)}$, $\mathbf{b}^{(l)}$ and $\mathbf{h}^{(l)}$ are the activation function, the vector of biases and the vector of outputs of the generic hidden layer l, respectively; the matrix $\mathbf{W}^{(l)}$ defines the weights

connecting the l-th hidden layer with the previous one, while $\mathbf{C}$ collects all the output weights, i.e. the ones between hidden layer L and the output one.

The internal parameters of the neural network $\boldsymbol{\Theta}$ are represented by the set of all weights and biases, i.e. $\boldsymbol{\Theta} = \{\mathbf{W}^{(l)},\ \mathbf{b}^{(l)},\ \ \mathbf{C}, \}$ (for $l = 1, ..., L$). Their tuning is conducted through a learning procedure wherein pairs of labeled data $\{\mathbf{x}_i, \mathbf{u}_i^*\}$ are submitted to the network. Aim of the training is allowing the network to emulate these data and generalize to inputs not available during the training phase. In this regard, the training process can be seen as the solution of an optimization problem consisting in the minimization of a loss or cost function. This objective function is typically in the form of the mean squared error:

$$\mathcal{L}_{\mathrm{u}} = \sum_{i=1}^{N_u} \frac{|\mathbf{u}_i - \mathbf{u}_i^*|^2}{2N_{\mathrm{u}}} \tag{13}$$

where $|\cdot|$ is the Euclidean norm, N_{u} is the total number of available labeled data, $\mathbf{u}_i^*$ is the target value for the i-th input data $\mathbf{x}_i$, while $\mathbf{u}_i$ is the corresponding prediction of the ANN.

Labeled samples are, in general, defined through numerical simulations and/or experiments, so data generation is usually a costly operation. Referring to the class of problems presented here, the available data can be in the form of displacements or deformation measurements from a limited number of points, such as in the case of strain gauges providing local deformations is few spots of the structure.

Due to the inherent cost of data acquisition, ANNs are typically trained in a small data regime. This consideration explains why the occurrence of overfitting problems is a tangible risk, which is even more true if the available dataset is also affected by some degree of noise. Under these circumstances, the resulting ANN may have poor generalization performances, causing the corresponding solution to violate the underlying physics of the problem.

In this context, PINNs represent a new class of ANNs that can be trained to inherently satisfy some known physical laws of the problem at hand [11] or a given mathematical model, thus enriching the information available from the training dataset. Collocation points are introduced for this scope.

Referring to the set of PDEs introduced in the previous section, and considering a single-

domain structure, the equations can be rephrased in more convenient way as:

$$\begin{cases} \boldsymbol{\mathcal{R}}\left(\mathbf{u},\mathbf{x}\right)=0 \quad \mathbf{x}\in\Omega \\ \boldsymbol{\mathcal{B}}\left(\mathbf{u},\mathbf{x}\right)=0 \quad \mathbf{x}\in\partial\Omega \end{cases} \tag{14}$$

It is now useful to provide an interpretation of Eq. (14) under the perspective of the PINN approach; specifically, the components of the vector $\mathbf{u}=\{u\ v\ w\}^{\mathrm{T}}$ are the physical quantities to be learnt by the neural network, while the components of $\mathbf{x}=\{x\ y\}^{\mathrm{T}}$ are the input parameters, as depicted in Figure 4.

The training process of PINNs is performed via definition of a physics-based loss function, where the information content of available data, defined in Eq. (13), is enriched with the underlying mathematical model sampled in correspondence of the collocation points:

$$\mathcal{L}=\mathcal{L}_{\mathrm{u}}+\mathcal{L}_{\mathrm{c}} \tag{15}$$

where the contribution associated with the physics/mathematical model $\mathcal{L}_{\mathrm{c}}$ reads:

$$\mathcal{L}_{\mathrm{c}}=\sum_{m=1}^{N_{\mathrm{f}}}\frac{\left|\boldsymbol{\mathcal{R}}_m-0\right|^2}{2N_{\mathrm{f}}}+\sum_{n=1}^{N_{\mathrm{b}}}\frac{\left|\boldsymbol{\mathcal{B}}_n-0\right|^2}{2N_{\mathrm{b}}} \tag{16}$$

where N_{f} and N_{b} are the number of collocation points inside the domain Ω and at the boundaries $\partial\Omega$, respectively, while $\boldsymbol{\mathcal{R}}_m=\boldsymbol{\mathcal{R}}\left(\mathbf{u}_m,\mathbf{x}_m\right)$ and $\boldsymbol{\mathcal{B}}_n=\boldsymbol{\mathcal{B}}\left(\mathbf{u}_n,\mathbf{x}_n\right)$ are the vector of residuals. The differential nature of $\mathcal{L}_{\mathrm{c}}$ implies the need for evaluating the derivatives of the network's output with respect to its inputs. Algorithmic Differentiation is employed for this scope [4].

The distinctive trait of PINNs is represented by the additional loss contribution $\mathcal{L}_{\mathrm{c}}$. This term provides a beneficial regularization effect on the training process by penalizing solutions not respectful of the specified physical laws or mathematical model. As a consequence, the training process becomes more effective: essentially, the neural network is restricted to seek a solution within the class of the physically admissible ones. This feature also means that the quality of the solution is drastically improved, even in those cases where the labeled data are scarce and noisy.

Depending on the contributions retained in Eq. (15), different types of neural network are obtained according to the nomenclature presented in Table 1.

Table 1: Neural Network nomenclature.

Neural Network	Loss function
Black-box	$\mathcal{L} = \mathcal{L}_{\mathrm{u}}$
White-box	$\mathcal{L} = \mathcal{L}_{\mathrm{c}}$
Gray-box	$\mathcal{L} = \mathcal{L}_{\mathrm{u}} + \mathcal{L}_{\mathrm{c}}$

In particular, *black-box* ANNs are characterized by a training process that relies upon labeled data only. These networks have been traditionally adopted for the construction of data-driven models of complex physical phenomena in absence of any laws or equations describing the process under analysis, see e.g. [9]. *White-box* ANNs are trained exclusively with collocation points, whose scope is enforcing the underlying governing equations at specific locations of the domain. These networks can be seen as numerical solvers for partial differential equations [38], like the FEM or meshfree methods. *Gray-box* ANNs combine a learning process based on labeled data and collocation points, and represent a hybrid configuration of the two types of networks presented above. This architecture allows to fully exploit the available information for the problem at hand, i.e. raw data coming from local measurements – used only by *black-box* ANNs – and physical laws – upon which *white-box* ANNs fully rely.

3.2 Training process

The internal parameters of the neural network $\boldsymbol{\Theta}$ are learned by minimizing the cost function defined in Eq. (15). The approach developed in this paper relies upon the use of Extreme Learning Machine. For completeness, a gradient-based approach, as commonly done for PINN-based approaches, is developed as well and used for comparison purposes. The two strategies are presented here below.

3.2.1 Gradient-Based Learning

Gradient-Based Learning (GBL) algorithms are optimization techniques commonly adopted for training ANNs [45]. An iterative process is performed, where all internal parameters are

recursively updated after evaluating the loss function $\mathcal{L}$ and its gradient $\nabla\mathcal{L}$. A sketch of this process is presented in Figure 5, where all the steps of the procedure are presented: (I) training data acquisition, (II) evaluation of the loss function through forward pass of the network and forward propagation of derivatives, (III) check of the tolerance as a stopping criterion, (IV) evaluation of the gradient of the loss function $\nabla\mathcal{L}$ through back propagation of the derivatives, and (V) updating of the internal parameters.

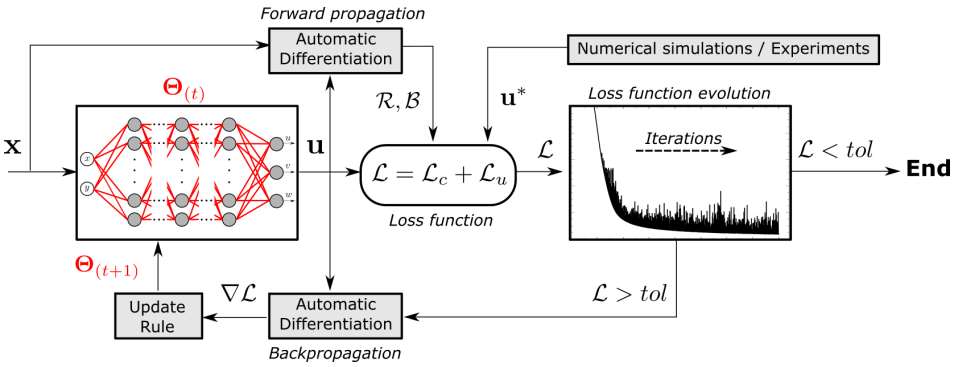


Figure 5: Gradient-Based Learning: workflow.

In the present work, the Adaptive Moment Estimation (Adam) [46] is employed as the updating rule for the training process of the networks. After a preliminary comparison between the different GBL algorithms, Adam proved to guarantee an excellent tradeoff between convergence performance and robustness. According to this training algorithm the internal parameters are iteratively updated as follows:

$$
\begin{aligned}
m_{(t)} &= \beta_1 m_{(t-1)} + (1-\beta_1)\nabla\mathcal{L}(\boldsymbol{\Theta}_{(t)}) \\
\nu_{(t)} &= \beta_2 \nu_{(t-1)} + (1-\beta_2)\nabla^2\mathcal{L}(\boldsymbol{\Theta}_{(t)}) \\
\hat{m}_{(t)} &= \frac{m_{(t)}}{1-\beta_1^t} \quad , \quad \hat{\nu}_{(t)} = \frac{\nu_{(t)}}{1-\beta_2^t} \\
\boldsymbol{\Theta}_{(t+1)} &= \theta_{(t)} - \frac{\eta}{\sqrt{\hat{\nu}_{(t)}} + \epsilon}\hat{m}_{(t)}
\end{aligned}
\tag{17}
$$

where (t) represents the current iterative step. Based on preliminary studies, in this work the adjustable hyperparameters β_1, β_2 and ϵ are taken as 0.9, 0.999 and 10^{-8}, respectively, with a learning rate of $\eta = 0.001$.

GBL algorithms are considered the state-of-the-art techniques for training ANNs. However, the learning process can be lengthy and time-consuming for many reasons. Firstly, the high

non-convexity of the minimization problem may cause the training process to stall in local minima and/or saddle points. To overcome this issue, repeated training procedures are generally needed, with optimization runs to be performed by considering different initial points $\boldsymbol{\Theta}_{(0)}$. In addition, the hyperparameters – parameters to be set before the training process, such as the network architecture, tolerance values and learning rate – generally require a preliminary tuning via trial and error processes. It follows that several runs are needed to find the optimum set up for the network and its learning algorithm in order to maximize the learning performance. A final aspect regards the number of internal parameters to be learnt, which can be very large in the case of ANNs with deep architectures.

For the reasons above, a learning algorithm called Extreme Learning Machine (ELM) is proposed as an alternative to GBL training.

3.2.2 Extreme Learning Machine

Extreme Learning Machine is a fast learning algorithm for training single hidden layer feedforward neural networks [23]. The main differences with GBL relies upon the limited set of internal parameters adjustable by the learning algorithm. The remaining ones are chosen randomly. Hence, the training process is carried out in a single step through the solution of a least-square problem. This approach allows the iterative process described by Figure 5 to be avoided, resulting in a drastic decrease of the time for training. In many cases, the time is several orders of magnitude smaller than for GBL-based algorithms.

On the contrary, the main limitation of ELM is the constraint on the network architecture, which is restricted to be in the form of a single hidden layer configuration. It follows that deep ANNs, which are inherently associated with a higher level of abstraction, cannot be used in this framework. It is worth noting that the limitation on the network depth does not determine a reduction on the representation capability of the network. Indeed, the universal approximation theorem ensures that *"multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available"* [43].

By considering a single hidden layer architecture with a generic number of hidden neurons

N_n, the output of the network is defined as:

$$\mathbf{u} = \mathbf{C}\sigma\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) \tag{18}$$

where σ is the activation function adopted in the hidden layer, $\mathbf{W} \in \mathbb{R}^{N_\mathrm{n}\times 2}$ and $\mathbf{b} \in \mathbb{R}^{N_\mathrm{n}\times 1}$ collect the input weights and biases, while $\mathbf{C} = [\mathbf{c}^{(u)}\ \mathbf{c}^{(v)}\ \mathbf{c}^{(w)}]^\mathrm{T} \in \mathbb{R}^{3\times N_\mathrm{n}}$ is the matrix of output weights. In ELM only these parameters are trainable, while the other ones are chosen randomly and are kept fixed throughout the learning process.

In the presence of labeled points $\{\mathbf{x}_i, \mathbf{u}_i^*\}$, the tuning process of the output weights is carried out so that the network predicts as accurately as possible the N_u available data. Using the network approximation of Eq. (18), this condition is expressed as:

$$\begin{cases} u_i = C_{1k}\sigma\left(W_{k1}x_i + W_{k2}y_i + b_k\right) = u_i^* \\ v_i = C_{2k}\sigma\left(W_{k1}x_i + W_{k2}y_i + b_k\right) = v_i^* \quad \text{for} \quad i = 1, ..., N_\mathrm{u} \\ w_i = C_{3k}\sigma\left(W_{k1}x_i + W_{k2}y_i + b_k\right) = w_i^* \end{cases} \tag{19}$$

where summation is implied over repeated indices.

The conditions of Eq. (19) correspond to solving three independent linear algebraic problems, one for each displacement component, in the form of:

$$\mathbf{H}\mathbf{c}^{(u)} = \mathbf{t}^{(u)} \qquad \mathbf{H}\mathbf{c}^{(v)} = \mathbf{t}^{(v)} \qquad \mathbf{H}\mathbf{c}^{(w)} = \mathbf{t}^{(w)} \tag{20}$$

where $\mathbf{H} \in \mathbb{R}^{N_\mathrm{u}\times N_\mathrm{n}}$ is the hidden layer matrix, whose generic element $h_{ik} = \sigma\left(W_{k1}x_i + W_{k2}y_i + b_k\right)$ represents the output of the k-th hidden neuron due to the i-th input data, $\mathbf{c}^{(u)}$, $\mathbf{c}^{(v)}$ and $\mathbf{c}^{(w)} \in \mathbb{R}^{N_\mathrm{n}\times 1}$ are the row vectors of the output weight matrix $\mathbf{C}$, while $\mathbf{t}^{(u)}$, $\mathbf{t}^{(v)}$ and $\mathbf{t}^{(w)} \in \mathbb{R}^{N_\mathrm{u}\times 1}$ are the vectors collecting the target values u_i^*, v_i^* and w_i^*, respectively.

In typical Machine Learning applications, the number of hidden neurons is taken smaller or equal to the number of training data, i.e. $N_\mathrm{n} \leq N_\mathrm{u}$. Therefore, the solutions of Eq. (20) can be found in a least-square sense through pseudoinversion of the coefficient matrix $\mathbf{H}$, i.e.:

$$\mathbf{c}^{(s)} = \mathbf{H}^\dagger\mathbf{t}^{(s)} \quad \text{for} \quad s = \{u, v, w\} \tag{21}$$

where $\mathbf{H}^\dagger \in \mathbb{R}^{N_\mathrm{n}\times N_\mathrm{u}}$ is the Moore-Penrose generalized inverse of $\mathbf{H}$.

Considering a PINN where the training dataset is integrated with a number of collocation

points, the output weights are trained to satisfy also Eq. (14) along with Eq. (19). Observing that the system of PDEs decribed by Eq. (14) is fully coupled in the three displacement components, the set of algebraic equations is then obtained in the form of:

$$\mathbf{L}\mathbf{c} = \mathbf{t} \tag{22}$$

where $\mathbf{c} = \{\mathbf{c}^{(u)}\ \mathbf{c}^{(v)}\ \mathbf{c}^{(w)}\}^{\mathrm{T}} \in \mathbb{R}^{3N_{\mathrm{n}}\times 1}$ is a global vector of unknowns collecting all the output weights of the network, while $\mathbf{L} = [\mathbf{L}_{\mathrm{u}}\ \mathbf{L}_{\mathrm{c}}]^{\mathrm{T}} \in \mathbb{R}^{3(N_{\mathrm{u}}+N_{\mathrm{f}}+N_{\mathrm{b}})\times 3N_{\mathrm{n}}}$ and $\mathbf{t} = \{\mathbf{t}_{\mathrm{u}}\ \mathbf{t}_{\mathrm{c}}\}^{\mathrm{T}} \in \mathbb{R}^{3(N_{\mathrm{u}}+N_{\mathrm{f}}+N_{\mathrm{b}})\times 1}$ are the coefficient matrix and vector of targets, respectively, and are assembled by substituting the network approximation of Eq. (18) in the conditions given by Eqs. (14) and (19).

The data-driven part of the system of Eq. (22) is defined as:

$$\mathbf{L}_{\mathrm{u}} = \begin{bmatrix} h_{ik} & 0_{ik} & 0_{ik} \\ 0_{ik} & h_{ik} & 0_{ik} \\ 0_{ik} & 0_{ik} & h_{ik} \end{bmatrix} \quad \text{and} \quad \mathbf{t}_{\mathrm{u}} = \begin{Bmatrix} u_i^* \\ v_i^* \\ w_i^* \end{Bmatrix} \quad \text{for} \quad \begin{matrix} i = 1, ..., N_{\mathrm{u}} \\ k = 1, ..., N_{\mathrm{n}} \end{matrix} \tag{23}$$

where $\mathbf{L}_{\mathrm{u}} \in \mathbb{R}^{3N_{\mathrm{u}}\times 3N_{\mathrm{n}}}$ is a block diagonal matrix obtained from the hidden layer matrix evaluated at the points where labeled data are available, while $\mathbf{t}_u \in \mathbb{R}^{3N_{\mathrm{u}}\times 1}$ collects all the corresponding target values.

The physics-driven part of Eq. (22) is given by:

$$\mathbf{L}_{\mathrm{c}} = \left(\mathbf{K} - \beta_2\omega^2\mathbf{M} + \beta_3\lambda\mathbf{G}\right) \quad \text{and} \quad \mathbf{t}_{\mathrm{c}} = -\beta_1\mathbf{f} \tag{24}$$

where $\mathbf{K},\ \mathbf{M},\ \mathbf{G} \in \mathbb{R}^{3(N_{\mathrm{f}}+N_{\mathrm{b}})\times 3N_{\mathrm{n}}}$ and $\mathbf{f} \in \mathbb{R}^{3(N_{\mathrm{f}}+N_{\mathrm{b}})\times 1}$, numerically interpreted as the stiffness, mass and geometric stiffness matrices and vector of external loads, respectively, are assembled as:

$$\begin{aligned} \mathbf{K} &= \begin{bmatrix} \boldsymbol{\mathcal{K}}(h_{mk}) \\ \tilde{\boldsymbol{\mathcal{B}}}(h_{nk}) \end{bmatrix} \qquad \mathbf{M} = \begin{bmatrix} \boldsymbol{\mathcal{M}}(h_{mk}) \\ 0_{nk} \end{bmatrix} \\ \mathbf{G} &= \begin{bmatrix} \boldsymbol{\mathcal{G}}(h_{mk}) \\ 0_{nk} \end{bmatrix} \qquad \mathbf{f} = \begin{Bmatrix} \mathbf{q}(\mathbf{x}_m) \\ \mathbf{g}(\mathbf{x}_n) \end{Bmatrix} \quad \text{with } \{m, n, k\} = 1, ..., \{N_{\mathrm{f}}, N_{\mathrm{b}}, N_{\mathrm{n}}\} \end{aligned} \tag{25}$$

with $\mathbf{g}(\mathbf{x})$ specifying nonhomogenous boundary conditions such that $\mathcal{B}(\mathbf{u}, \mathbf{x}) = \tilde{\mathcal{B}}(\mathbf{u}, \mathbf{x}) - \mathbf{g}(\mathbf{x})$.

The training process of PINNs with ELM is carried out as shown in Figure 6, where the

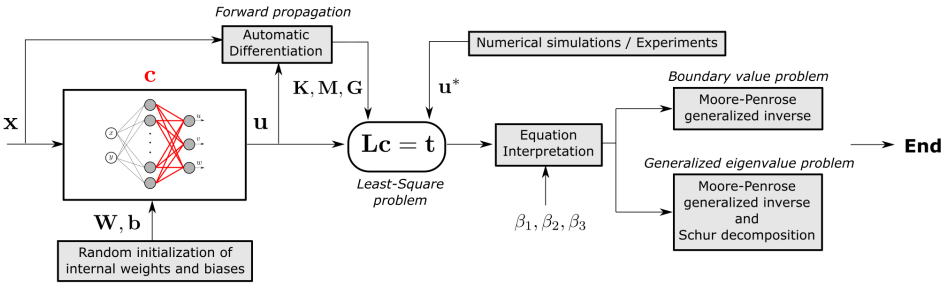


Figure 6: Extreme Learning Machine: workflow.

dependency over the parameter β_i is illustrated.

Specifically, the output weights are found by computing the pseudoinverse of the coefficient matrix $\mathbf{L}$ in the case of a static problem ($\beta_i = \delta_{i1}$), i.e.:

$$\text{For} \quad \beta_i = \delta_{i1}: \quad \mathbf{Lc} = \mathbf{t} \qquad \Rightarrow \qquad \mathbf{c} = \mathbf{L}^{\dagger}\mathbf{t} \tag{26}$$

where Moore-Penrose generalized inverse $\mathbf{L}^{\dagger}$ is computed via Singular Value Decomposition (SVD).

In the other cases ($\beta_i = \delta_{i2}$ and $\beta_i = \delta_{i3}$), a generalized eigenvalue problem is obtained, where only collocation points are considered. It follows that Eq. (22) reduces to:

$$\mathbf{L}_{\mathrm{c}}\mathbf{c} = \mathbf{t}_{\mathrm{c}} \tag{27}$$

As the linear algebraic system defined for training is, in general, rectangular, the use of a preconditioner $\mathbf{P}$ is required to transform the problem into a form that is more suitable for the numerical solution. The preconditioning matrix is based on the pseudoinverse of the stiffness matrix, i.e. $\mathbf{P} = \mathbf{K}^{\dagger}$, which premultiplies Eq. (27) leading to:

$$\begin{aligned} &\text{For} \quad \beta_i = \delta_{i2}: \quad \mathbf{Kc} = \omega^2\mathbf{Mc} \qquad \Rightarrow \qquad (\mathbf{K}^{\dagger}\mathbf{K})\mathbf{c} = \omega^2(\mathbf{K}^{\dagger}\mathbf{M})\mathbf{c} \\ &\text{For} \quad \beta_i = \delta_{i3}: \quad \mathbf{Kc} = -\lambda\mathbf{Gc} \qquad \Rightarrow \qquad (\mathbf{K}^{\dagger}\mathbf{K})\mathbf{c} = -\lambda(\mathbf{K}^{\dagger}\mathbf{G})\mathbf{c} \end{aligned} \tag{28}$$

where $\mathbf{K}^{\dagger}$ can be interpreted as a transformation matrix projecting the rectangular problem defined by Eq. (27) from the space $\mathbb{R}^{3(N_{\mathrm{f}}+N_{\mathrm{b}})\times 3N_{\mathrm{n}}}$ to the space $\mathbb{R}^{3N_{\mathrm{n}}\times 3N_{\mathrm{n}}}$.

The rightmost equations of Eq. (28) are solved as a standard generalized eigenvalue problem via Schur decomposition. The eigenvalues correspond to the natural frequencies and buckling multipliers, and eigenvectors are the mode shapes defined by the trained output weights.

4 Parameter identification via Physics-Informed Neural Networks and Extreme Learning Machine

The application of PINNs for parameter discovery of PDEs has been discussed in previous efforts in the literature. For instance, PINNs have been applied for learning unknown model parameters of the Navier–Stokes and Korteweg–de Vries equations in Ref. [11]. An application of a similar framework to the parameter identification in solid mechanic problems is found in Ref. [13]. In the above mentioned works, the inverse problem with PINNs is discussed in the context of GBL approaches; on the contrary, this work addresses the same problem by referring to ELM.

When dealing with the inverse problem, one is interested in identifying a set of unknown parameters $\mathbf{\Lambda}$ of a mathematical model starting from a set of observed data $\{\mathbf{x}_i, \mathbf{u}_i^*\}$. The model is then expressed as:

$$\begin{cases} \boldsymbol{\mathcal{R}}\left(\mathbf{\Lambda}, \mathbf{u}, \mathbf{x}\right) = 0 \quad \mathbf{x} \in \Omega \\ \boldsymbol{\mathcal{B}}\left(\mathbf{\Lambda}, \mathbf{u}, \mathbf{x}\right) = 0 \quad \mathbf{x} \in \partial\Omega \end{cases} \tag{29}$$

The inverse problem is solved by training a PINN, where the set of parameters to be learnt includes now the output weights of the network $\mathbf{C}$ as well as the model parameters to be identified $\mathbf{\Lambda}$. It follows that the global vector of unknowns is defined as:

$$\mathbf{c} = \{\mathbf{c}^{(u)}\ \mathbf{c}^{(v)}\ \mathbf{c}^{(w)}\ \mathbf{\Lambda}\}^{\mathrm{T}} \in \mathbb{R}^{(3N_{\mathrm{n}}+N_{\Lambda})\times 1} \tag{30}$$

where N_Λ is the number of unknown model parameters. The resulting least-square problem $\mathbf{Lc} = \mathbf{t}$ is now nonlinear inasmuch $\mathbf{L} = \mathbf{L}(\mathbf{\Lambda})$, as seen from Eq. (29).

The solution is sought using an iterative least-square approach [24], where the vector of unknowns is updated according to:

$$\mathbf{c}_{(t+1)} = \mathbf{c}_{(t)} + \Delta\mathbf{c} \tag{31}$$

with $\Delta\mathbf{c}$ defined from the solution of the linear least-square problem:

$$\mathbf{J}(\mathbf{c}_{(t)})\Delta\mathbf{c} = \mathbf{r}(\mathbf{c}_{(t)}) \tag{32}$$

where $\mathbf{r} = \mathbf{Lc} - \mathbf{t}$ and $\mathbf{J} = \frac{\partial \mathbf{r}}{\partial \mathbf{c}}$ are the vector of residuals and the Jacobian matrix, respectively. Starting from an initial guess $\mathbf{c}_{(0)}$, the residual and the Jacobian are evaluated

at each iteration, the latter by making use of automatic differentiation. The incremental vector $\Delta\mathbf{c}$ is then obtained by solution of Eq. (32) as $\Delta\mathbf{c} = \mathbf{J}^{\dagger}\mathbf{r}$.

The convergence of the procedure is checked by referring to two criteria: the first one refers to the current loss function, i.e. $\mathcal{L}_{(t)} < \text{tol}$; the second one relies upon the difference between the loss function at two consecutive iterations, i.e. $|\mathcal{L}_{(t)} - \mathcal{L}_{(t-1)}| < \text{tol}$. The iterative process is terminated when one of the two criteria is met.

5 Results

In this section, the proposed PINNs-based framework is applied for the solution and identification of different problems in linear elasticity involving plate- and shell-like structures. The section is organized as follows: in the first part, a validation is presented against reference solutions to demonstrate the effectiveness of PINNs to solve PDEs and to check the correct implementation of the method; in the second part, a series of parametric studies is presented to gather insight into the hyperparameters regulating the learning process of PINNs; the last part is devoted to the application of the method to relatively complex problems, such as those arising from the analysis of real-life engineering structures. Examples are presented involving general geometries, arbitrary boundary conditions and interactions between plate and shell subdomains.

5.1 Validation

A preliminary validation is conducted by considering the analysis of symmetrically layered specially orthotropic plates, subjected to simply-supported boundary conditions. Thus, the structure under investigation has a simple geometry and is believed of interest inasmuch exact solutions are available for this case. The governing equation for this problem reads [31]:

$$\begin{aligned}\boldsymbol{\mathcal{R}} =& D_{11}w_{,xxxx} + 2\left(D_{12} + 2D_{66}\right)w_{,xxyy} + D_{22}w_{,yyyy} + \beta_1 n_z + \beta_2\left[I_0\ddot{w} - I_2\left(\ddot{w}_{,xx} + \ddot{w}_{,yy}\right)\right] + \\ &+ \beta_3\left(\overline{N}_{xx}w_{,xx} + 2\overline{N}_{xy}w_{,xy} + \overline{N}_{yy}w_{,yy}\right) = 0 \quad \text{in} \quad \Omega \end{aligned} \tag{33}$$

which can be understood as a special case of Eq. (1). The simply-supported boundary conditions are:

$$\boldsymbol{\mathcal{B}} = \begin{cases} w_n = 0 \\ M_{nn} = 0 \end{cases} \qquad \text{in} \quad \partial\Omega \tag{34}$$

The exact solutions for bending, free vibration and buckling are summarized in the Appendix.

For validation purposes, a square plate is considered with nondimensional thickness $h/a = 1/500$. The material elastic coefficients are $E_{11}/E_{22} = 16.67$, $G_{12}/E_{22} = 0.56$, $\nu_{12} = 0.32$, while the layup is $[90/0]_s$.

A single hidden layer architecture with 100 hidden neurons is considered. The learning algorithm is the ELM and the hyperbolic tangent is adopted as activation function in all the hidden units. The input weights and biases are chosen randomly from a uniform Gaussian distribution in the range $(\mathbf{W}, \mathbf{b}) \in [-1, 1]$. The set of training data is constituted by a uniform grid of $N_c = 15 \times 15$ collocation points, expressing the requirements of Eqs. (33) and (34). Therefore, the PINN is used here as a *white-box*. A total of 400 testing points, distributed randomly in the domain, are used for assessing the accuracy of the predicted solution. All data points are normalized according to the transformation $\xi = 2x/a$ and $\eta = 2y/b$, with $x \in \left[-a/2, a/2\right]$ and $y \in \left[-b/2, b/2\right]$. Two performance parameters are used for verifying the quality of the results, i.e. the $\mathbb{L}_2$-norm of the error distribution for field quantities (displacements and stress distributions), and the relative error percentage $\mathbb{E}_{\%}$ for scalar ones (e.g. natural frequencies, buckling multipliers, local displacements). These two metrics are defined as:

$$\mathbb{L}_2\left[\Phi\right] = \sqrt{\sum_{k=1}^{400} \left(\frac{\Phi_k - \Phi_k^{\text{ref}}}{\Phi_{\text{max}}^{\text{ref}}} \right)^2} \quad \text{and} \quad \mathbb{E}_{\%}\left[\phi\right] = \left| \frac{\phi - \phi^{\text{ref}}}{\phi^{\text{ref}}} \right| \times 100 \tag{35}$$

where Φ and ϕ are two generic field and scalar quantities predicted by the network, respectively, while Φ^{ref} and ϕ^{ref} are the corresponding reference solutions.

The solution for the bending problem is illustrated in Figure 7, where the normalized deflection shape $\overline{w} = w/w_{\text{max}}^{\text{ref}}$ is reported along with the error distribution $|w - w^{\text{ref}}|/w_{\text{max}}^{\text{ref}}$. Similar plots are reported for the slope $w_{,x}$ and bending moment resultant M_{xx}. The comparison against the exact solution (see Appendix, Eq. (58)) reveals the excellent agreement

between the present solution and the analytical one.

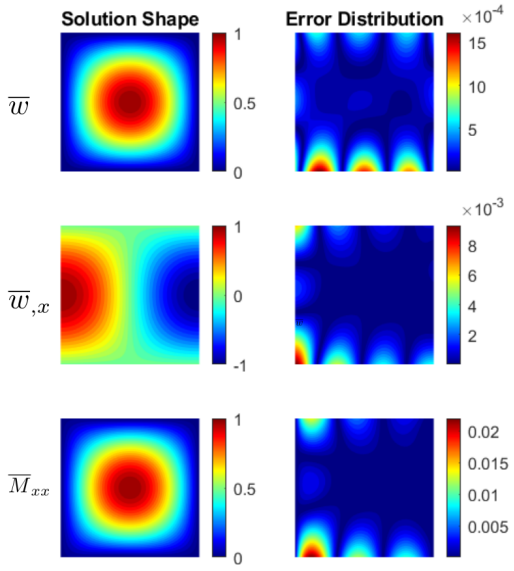


Figure 7: Results and errors against exact solution: bending problem.

Looking at the deflection field w, the maximum error is achieved at the edges, where the displacement gradients are the highest. The same pattern is observed for the rotation and the bending moment. This behaviour is explained by observing that the deflection w is the only quantity directly learned by the network. As a consequence, the errors are amplified when postprocessing by taking the derivative of w.

A summary of the $\mathbb{L}_2$-norm and the percent errors is provided in Table 2, where the excellent agreement between predicted results and reference solutions is further demonstrated.

Table 2: $\mathbb{L}_2$-norms and percent error of the solution predicted by PINN – bending analysis.

	$\mathbb{L}_2$	$\mathbb{E}_{\%}$
Bending		
$\overline{w}$	0.0072	0.0103
$\overline{w}_{,x}$	0.0307	0.0093
$\overline{M}_{xx}$	0.0759	0.0070

Concerning the free vibration and buckling problems, the first three eigenmodes are pre-

sented in Figure 8 along with the corresponding error distributions.

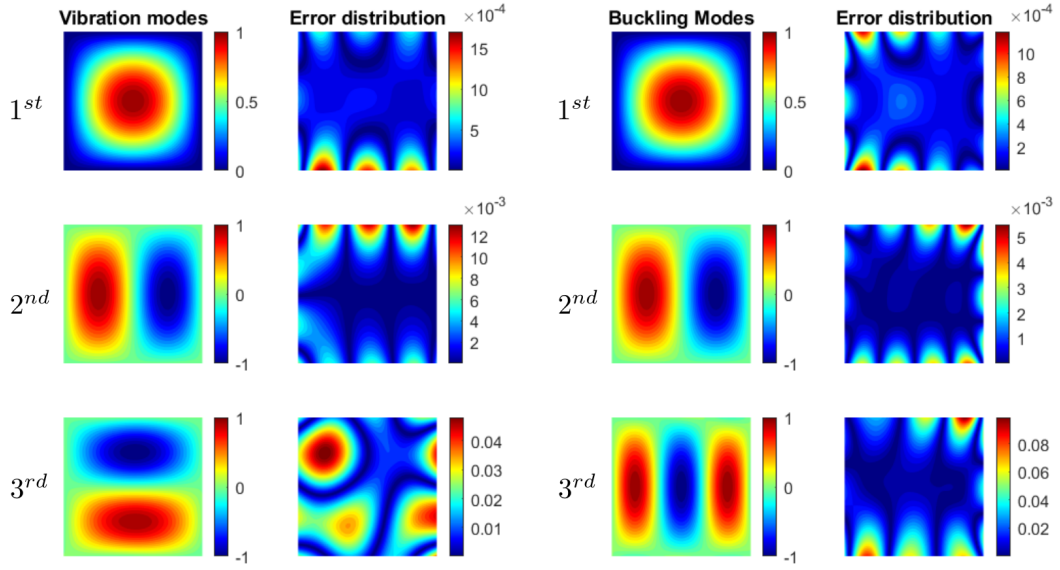


Figure 8: First three modes and error against exact solution for: (a) free vibration problem, (b) buckling problem.

Even in this case, the discrepancies between the predicted solutions and the exact ones are very small, both in terms of mode patterns, i.e. $\mathbb{L}_2\left[w_{mn}\right]$, as well as in terms of frequencies and critical loads, i.e. $\mathbb{E}_{\%}\left[\omega_{mn}\right]$ and $\mathbb{E}_{\%}\left[\lambda_{mn}\right]$, see Table 3.

Regarding the training time, few fractions of seconds were required for completing the training process for the three problems above. In this regard, standard GBL approaches would require much larger times – of the order of minutes, see [16] – for solving analogous problems. Hence, the effectiveness of the ELM-based approach can be exploited to perform parametric studies, which are useful for understanding the main features of PINNs, as well as finding the network architecture for optimizing the training process. These aspects are presented in the following section, where parametric studies are presented on the network hyperparameters.

Table 3: $\mathbb{L}_2$-norms and percent error of the solution predicted by PINN – free vibration and buckling analysis.

	$\mathbb{L}_2$	$\mathbb{E}_{\%}$
Free vibration		
1^{st} mode	0.0081	0.0063
2^{nd} mode	0.0664	0.0191
3^{rd} mode	0.4069	0.0215
Buckling		
1^{st} mode	0.0055	0.0039
2^{nd} mode	0.0222	0.0015
3^{rd} mode	0.4755	0.3865

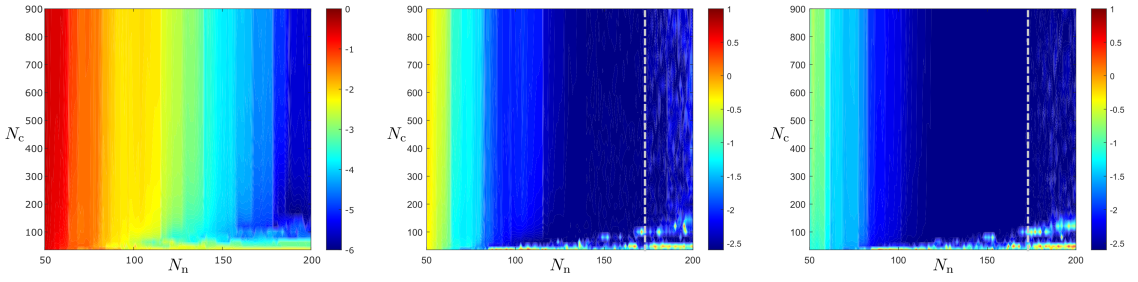
5.2 Parametric study on hyperparameters

The choice of the hyperparameters is a crucial aspect in the network set up, although it can be a non-trivial task and trial and error procedures are often necessary. The study conducted next refers to the same test case presented in the previous section. Starting from the same baseline network architecture presented earlier, the hyperparameters are modified, and their influence on the network predictions and learning performance is illustrated.

Number of neurons and collocation points

In ELM, the number of neurons N_{n} and collocation points N_{c} define the size of the least-square problem – number of columns and rows of the matrix of coefficients, respectively – to be solved for finding the output weights of the network, see Eq. (22). The influence of these two hyperparameters is investigated here for the case of bending, free vibration and buckling problems. With this purpose, the contour plots of the $\mathbb{L}_2$-norm of the error distributions in logarithmic scale are reported for the deflected shape (static analysis) and the eigenmodes (free vibrations and buckling analysis) in Figure 9. The number of neurons and collocation points are taken in the range $N_{\text{n}} \in [50, 200]$ and $N_{\text{c}} \in [36, 900]$, respectively.

For the bending problem, a progressive reduction of the error can be noted for an increasing



(a) (b) (c)

Figure 9: Influence of number of neurons N_n and collocation points N_c on $\log[\mathbb{L}_2]$ for: (a) static deflection, (b) first vibration mode, (c) first buckling mode.

number of neurons in Figure 9(a); on the contrary, the number of collocation points has a slight influence on the solution once a certain threshold, approximately $N_\mathrm{c} = 100$, is reached. Even for free vibration and buckling problems, the solution is not particularly sensitive to the number of collocations points, as revealed by Figures 9(b) and 9(c). As opposed to the static case, the solution improves if the number of neurons is increased up to the dashed lines of Figures 9(b) and 9(c); then, the solution is seen to worsen if this number is further increased. This behavior stems from the poor conditioning of the matrices appearing in Eq. (28), which are typically not full-rank due to the random selection of the input weights and biases, as well as for the presence of rows of zeros, see Eq. (25). The ill-conditioning becomes more severe when the number of neurons is increased, as seen in Figure 10, where the condition number $k(\cdot) = |(\cdot)||(\cdot)^{-1}|$ of the stiffness matrix $\mathbf{K}$ is reported in logarithmic scale for different combinations of N_n and N_c.

Therefore, it is concluded that the number of neurons has to be bounded when dealing with eigenvalue problems to prevent numerical issues. On the contrary, the linear static solution displays much more robustness, which is ascribed to the pseudoinversion algorithm based on a SVD approach.

Convergence of the solution

The number of computational units N_n can be interpreted as the number of shape functions used by the *white-box* neural network to approximate the solution of the PDEs in Eqs. (33)

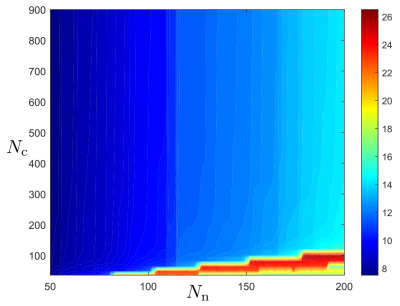


Figure 10: Condition number of the stiffness matrix for different number of neurons N_n and collocation points N_c.

and (34), i.e.:

$$w\left(\xi,\eta\right)=\sum_{k=1}^{N_\mathrm{n}}c_k h_k\left(\xi,\eta\right)\quad\text{with}\quad h_k\left(\xi,\eta\right)=\sigma\left(W_{k1}\xi+W_{k2}\eta+b_k\right)\tag{36}$$

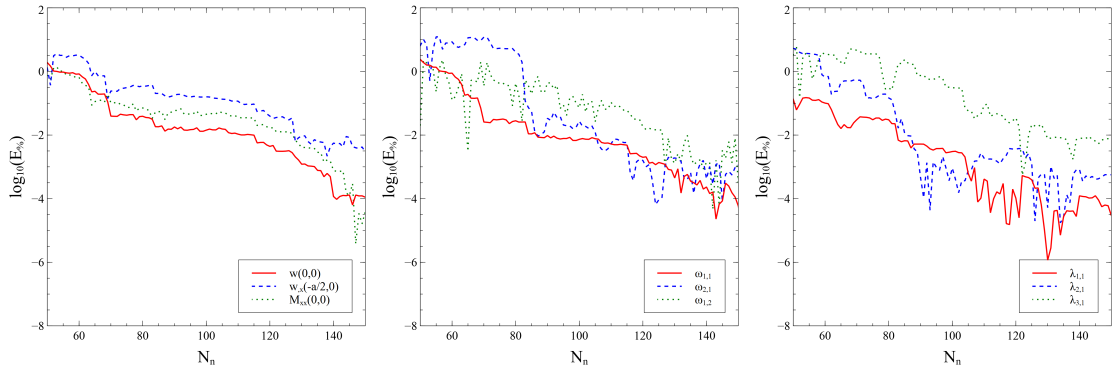
where $h_k\left(\xi,\eta\right)$ is the shape function associated with the k-th neuron in the hidden layer, σ is the activation function, b_k is the internal bias, W_{k1} and W_{k2} are the input weights connecting the neuron with the inputs (ξ,η), while c_k is the output weight acting as the unknown amplitude of the shape function. Therefore, it is possible to study the convergence of the PINN solution by quantifying the error obtained using different numbers of neurons. In particular, the errors with respect to the exact solutions are evaluated for the bending deflection and its derivatives, vibration frequencies and critical loads, as presented in Figure 11.

As seen, the convergence is not uniform, but is characterized by a certain degree of oscillation, both for static and eigenvalue analyses. It is interesting to observe that the derivatives may sometimes be locally better approximated than the unknown function itself, see Figure 11(a). In addition, lower frequencies and buckling multipliers can in some cases be affected by a larger error with respect to higher order ones, see Figures 11(b) to 11(c).

Distribution of collocation points

Another important aspect in the application of PINNs regards how training points are distributed within the computational domain.

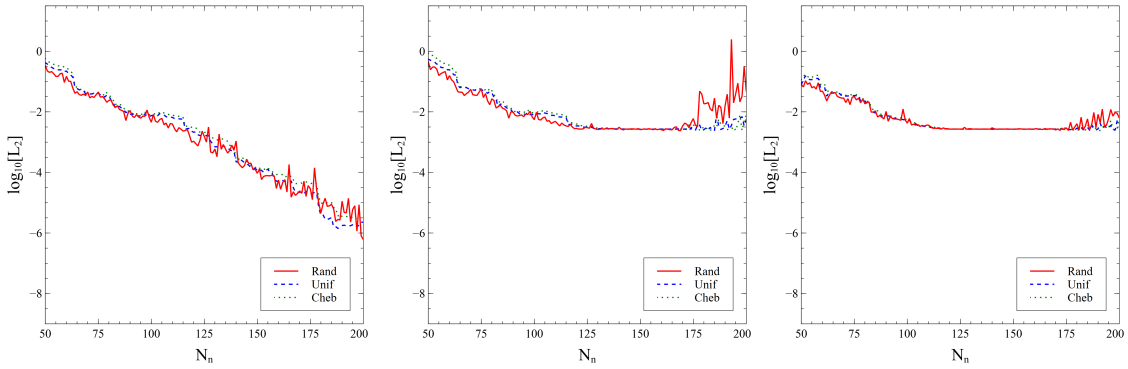
The results of Figure 12 illustrate a convergence study for increasing number of neurons



(a) (b) (c)

Figure 11: Convergence study for: (a) bending, (b) free vibration, (c) buckling problems.

and considering three different distributions of collocation points, i.e. random, uniform and Chebyshev distributions. The study is presented for static, free vibration and buckling problems. The $\mathbb{L}_2$-norm of the errors is referred to the static deflection in the first case, and the first eigenmodes in the second and third cases.



(a) (b) (c)

Figure 12: Effect of collocation point distributions on the $\mathbb{L}_2$-norm of the error for (a) static deflection, (b) first vibration mode, (c) first buckling mode.

For the bending problem, see Figure 12(a), one can see that random distributions lead to overall better results with respect to the ones obtained with organized grids. At the same time, random distributions are associated with largest sensitivity to any change in the

network architecture, as revealed by increasingly pronounced oscillations. These same considerations hold true for free vibration and buckling problems, as depicted in Figures 12(b) and 12(c). However, a detrimental effect is observed on the solution, irrespective of the distribution considered, when the neurons are increased beyond a certain value. Poor conditioning of the matrices occurs especially for random distributions, while numerical issues are milder in the case of organized grids. This behaviour is further clarified by the plots of Figure 13, where the condition number and the rank of the stiffness matrix $\mathbf{K}$ are reported for increasing number of neurons. As seen, the adoption of Chebyshev or uniform grids tends to mitigate the conditioning issues, leading to a solution that is stable even for large numbers of neurons.

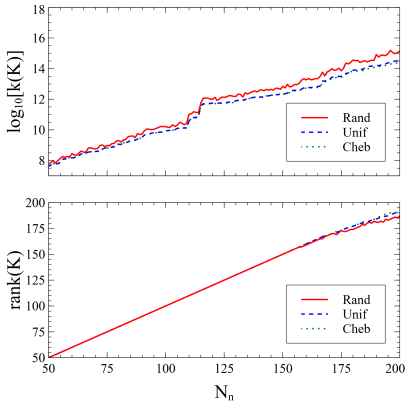


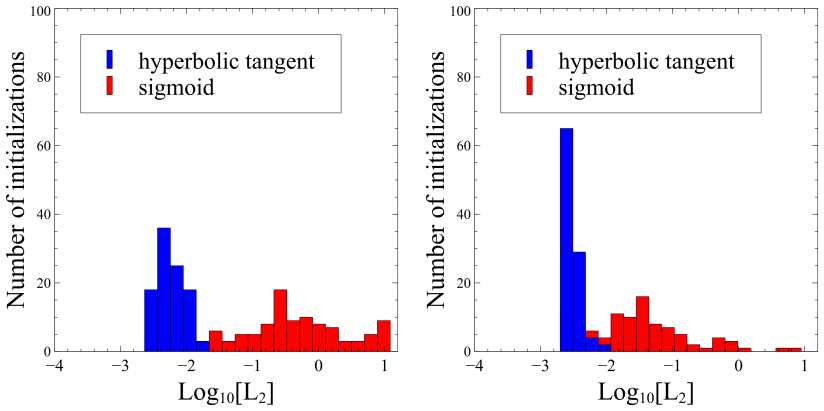
Figure 13: Effect of collocation point distributions on the condition number and rank of the stiffness matrix.

Activation function and initialization of input weights and biases

The activation function and the initialization of the input weights and biases affect the expression of the shape functions, as revealed by Eq. (36). An investigation over their role is then conducted by considering two activation functions typically used in neural networks: hyperbolic tangent, $\sigma = tanh(z)$, and sigmoid logistic function, $\sigma = \frac{e^z}{1+e^z}$; a total of 100 different random initializations of internal weights and biases in the range $(W_{k1}, W_{k2}, b_k) \in [-1, 1]$ are considered.

The distribution of the errors is presented in Figure 14 for the first vibration and buckling

eigenmodes, as the effect of the choice of activation functions and random initializations is more pronounced for these types of analysis.



(a) (b)

Figure 14: Probability distribution of the $\mathbb{L}_2$-norm of the errors for different activation functions: (a) first vibration mode, (b) first buckling mode.

From Figure 14 it can be seen that the the hyperbolic tangent guarantees the smallest average values of the $\mathbb{L}_2$-norm of the errors. With this activation function, the solution can be represented very accurately with error norms $\mathbb{L}_2$ of the order of $10^{-3} - 10^{-2}$, irrespective of the initialization adopted. On the other hand, a larger variability in results is observed for the sigmoid, which is not capable of capturing the exact solution for some initializations. The motivations of this behavior are ascribable to numerical issues affecting the eigenvalue solver. The distribution of condition numbers are reported in Figure 15 for the different activation functions. It is clear that worst conditioning is observed for the sigmoid function compared to the hyperbolic tangent one.

From the parametric studies above, it is concluded that the hyperbolic tangent guarantees, for the problems at hand, smaller errors and less sensitivity to the parameter's initialization. For this reason, this activation function is retained in the following studies.

5.3 Static analysis of a cylindrical panel with cutout

Goal of this section is presenting the potential of PINNs as a mean for solving elasticity problems characterized by more complex configurations. In addition, insights are provided

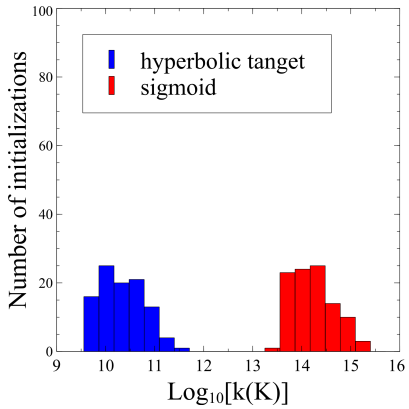


Figure 15: Probability distribution of the condition number of the stiffness matrix considering different activation functions.

regarding the features of the method for different network configurations. Referring to the nomenclature of Table 1, *black-*, *white-* and *gray-box* neural networks are adopted and compared each other.

The structure under investigation is an isotropic cylindrical shell with a circular cutout at its center, a sketch of which is presented in Figure 16.

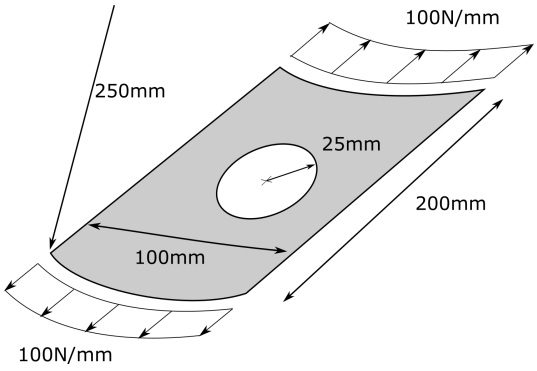


Figure 16: Cylindrical panel with cutout: geometry and loading conditions.

The planar dimensions are 200×100 mm^2, the long side being aligned with the axial direction, and the thickness is $t = 1$ mm. The radius of curvature is $R = 250$ mm, while the circular cutout has radius 25 mm. The elastic properties of the material are $E = 70$ GPa and $\nu = 0.3$. The shell is simply-supported and is loaded with two in-plane uniform tensile loads $\hat{N}_{xx} = 100$ N/mm acting along the short edges. Due to the double symmetry of the

problem, only one quarter of the structure is analyzed. The FE model of the structure is realized using Abaqus S4R shell elements; the mesh is presented in Figure 17.

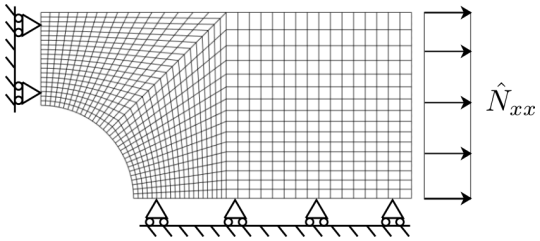


Figure 17: Cylindrical panel with cutout: Finite Element model.

FE simulations are conducted with the twofold aim of obtaining reference solutions for validation purposes, as well as for generating the training dataset to be used in the context of *black-* and *gray-box* approaches.

Four different neural networks, hereinafter referred to as NET1, NET2, NET3 and NET4, are considered for solving the problem. Based on the parametric studies presented earlier, the following setup of hyperparameters will be adopted for the remaining part of this work, unless otherwise specified: uniform distributions are used for collocation points and the hyperbolic tangent is adopted as the activation function for the hidden units. The four networks are trained with different strategies, while sharing the same shallow architecture with two inputs, x and y, one hidden layer and three outputs, u, v and w. An overview of the distribution and type of training data used for the different networks is provided in Figure 18 and discussed here below.

NET1

The first neural network, NET1, is trained in a completely data-driven manner, i.e. no information is provided on the physics of the problem. For the example at hand, data are generated via FE analysis. Displacements are available from the FE model in correspondence of the points reported in Figure 18(a), and represent the available dataset.

The network has $N_{\mathrm{n}} = 400$ hidden neurons and the training process is carried out using a GBL approach. A total of $N_{\mathrm{u}} = 947$ labeled samples $\{\mathbf{x}_i, \mathbf{u}_i^*\}$ are considered, where

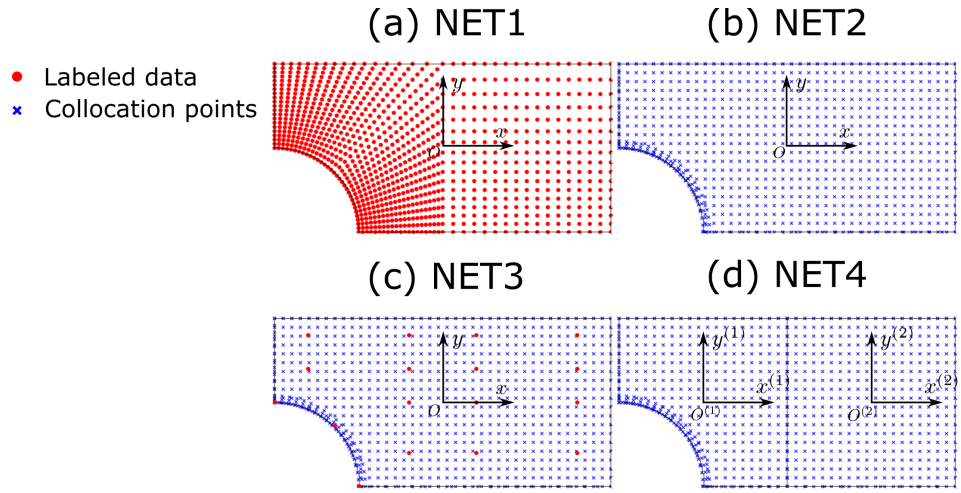


Figure 18: Cylindrical panel with cutout: training data distribution for (a) NET1, (b) NET2, (c) NET3, (d) NET4.

$\mathbf{x}_i = \{x_i, y_i\}$ and $\mathbf{u}_i^* = \{u_i^*, v_i^*, w_i^*\}$. The resulting loss function is defined as:

$$\mathcal{L}_{\text{NET1}} = \sum_{i=1}^{N_\text{u}} \frac{|\mathbf{u}_i - \mathbf{u}_i^*|^2}{2N_\text{u}} \tag{37}$$

where $\mathbf{u}_i = \{u_i, v_i, w_i\}$ represents the output prediction of the network for the input data $\mathbf{x}_i$.

NET2

The second neural network NET2 has the same architecture of NET1, but is trained using a full physics-informed approach. So, it is denoted as a *white-box* network. The training process is carried out using ELM with a total of $N_\text{c} = 900$ collocation data distributed as per Figure 18(b).

The loss function is defined as:

$$\mathcal{L}_{\text{NET2}} = \sum_{m=1}^{N_\text{f}} \frac{|\boldsymbol{\mathcal{R}}_m|^2}{2N_\text{f}} + \sum_{n=1}^{N_\text{b}} \frac{|\boldsymbol{\mathcal{B}}_n|^2}{2N_\text{b}} \tag{38}$$

where $\boldsymbol{\mathcal{R}}$ is the residual function expressing the equilibrium unbalance in the domain, while $\boldsymbol{\mathcal{B}}$ refers to the boundaries; the summatories in Eq. (38) are taken over the collocation points inside the domain and along its border, denoted as N_f and N_b, respectively.

NET3

The third neural relies upon a *gray-box* approach, where labeled samples are integrated with the mathematical model of the structure. In this regards, NET3 displays the same architecture of NET2, but the set of training data is enriched by additional $N_{\mathrm{u}} = 17$ labeled points distributed as shown in Figure 18(c). Hence, the resulting loss function is composed of two parts, expressing the physics-informed and data-driven parts:

$$\mathcal{L}_{\mathrm{NET3}} = \sum_{m=1}^{N_{\mathrm{f}}} \frac{|\boldsymbol{\mathcal{R}}_m|^2}{2N_{\mathrm{f}}} + \sum_{n=1}^{N_{\mathrm{b}}} \frac{|\boldsymbol{\mathcal{B}}_n|^2}{2N_{\mathrm{b}}} + \sum_{i=1}^{N_{\mathrm{u}}} \frac{|\mathbf{u}_i - \mathbf{u}_i^*|^2}{2N_{\mathrm{u}}} \tag{39}$$

The hybrid approach of NET3 provides an interesting example of the potential offered by PINNs, where the network combines both available information of the solution and physics knowledge coming from mathematical models.

NET4

Similarly to NET2, the fourth network configuration falls in the class of *white-box* ANNs. NET4 is proposed as a viable alternative for improving the representation capabilities of NET2 without increasing the number of neurons and, in turn, the training time for weight tuning.

The approach implemented in NET4 consists in distributing the available neurons in multiple subnetworks, each one responsible for approximating the solution in different subportions of the domain.

The two subnetworks composing NET4 are characterized by the same architecture of NET2, but with reduced number of neurons, i.e. $N_{\mathrm{n}}^{(1)} = 300$ and $N_{\mathrm{n}}^{(2)} = 100$. Additional collocations point are introduced at the interface between the two subdomains, as shown in Figure 18(d).

The loss function is defined as:

$$\mathcal{L}_{\mathrm{NET4}} = \sum_{m=1}^{N_{\mathrm{f}}^{(1)}} \frac{|\boldsymbol{\mathcal{R}}_m^{(1)}|^2}{2N_{\mathrm{f}}^{(1)}} + \sum_{n=1}^{N_{\mathrm{b}}^{(1)}} \frac{|\boldsymbol{\mathcal{B}}_n^{(1)}|^2}{2N_{\mathrm{b}}^{(1)}} + \sum_{m=1}^{N_{\mathrm{f}}^{(2)}} \frac{|\boldsymbol{\mathcal{R}}_m^{(2)}|^2}{2N_{\mathrm{f}}^{(2)}} + \sum_{n=1}^{N_{\mathrm{b}}^{(2)}} \frac{|\boldsymbol{\mathcal{B}}_n^{(2)}|^2}{2N_{\mathrm{b}}^{(2)}} + \sum_{j=1}^{N_{\mathrm{int}}} \frac{|\boldsymbol{\mathcal{I}}_j|^2}{2N_{\mathrm{int}}} \tag{40}$$

where N_{int} is the number of the interface collocation points and $\boldsymbol{\mathcal{I}}$ is the residual referred to the interface conditions between the subdomains, which is defined as $\boldsymbol{\mathcal{I}} = [\boldsymbol{\mathcal{I}}_{\mathrm{con}}\ \boldsymbol{\mathcal{I}}_{\mathrm{equ}}]^{\mathrm{T}}$,

where:

$$\boldsymbol{\mathcal{I}}_{\text{con}} = \begin{cases} u^{(1)} - u^{(2)} = 0 \\ v^{(1)} - v^{(2)} = 0 \\ w^{(1)} - w^{(2)} = 0 \\ w^{(1)}_{,x} - w^{(2)}_{,x} = 0 \end{cases} \quad \text{and} \quad \boldsymbol{\mathcal{I}}_{\text{equ}} = \begin{cases} N^{(1)}_{xx} - N^{(2)}_{xx} = 0 \\ N^{(1)}_{xy} - N^{(2)}_{xy} = 0 \\ V^{(1)}_{x} - V^{(2)}_{x} = 0 \\ M^{(1)}_{xx} - M^{(2)}_{xx} = 0 \end{cases} \quad \text{in} \quad \partial\Omega_{\text{int}} \tag{41}$$

Comparison between different architectures

The results are summarized in Figures 19 to 21, where the contours are presented for the static deflection, the membrane resultant N_{xx} and the bending moment per unit length M_{xx}, respectively. In addition, an overview of the errors is provided in Table 4 in terms of $\mathbb{L}_2$-norms of the errors.

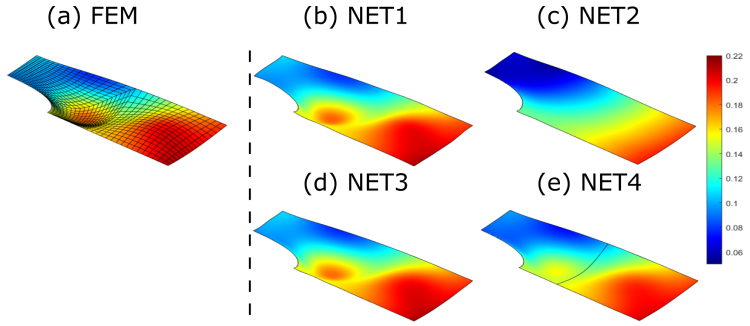


Figure 19: Cylindrical panel with cutout: static deflection: (a) FE, (b) NET1, (c) NET2, (d) NET3, (e) NET4.

The solutions obtained with NET1 are shown in Figures 19(b)-21(b). These solutions required approximately 40000 GBL iterations leading to a final loss function of the order of $\mathcal{L}_{\text{NET1}} = 10^{-4}$.

The comparison against FE solutions reveals excellent agreement in terms of static deflection, while noticeable discrepancies can be noted for the membrane and bending resultants. The $\mathbb{L}_2$ errors, available in Table 4, are of the order of 10^{-1} for the displacements, but the magnitude increases for the stress-related quantities N_{xx} and M_{xx}. This response is motivated by the full data-driven strategy used for training NET1. Indeed, NET1, due to its *black-box* nature, learns blindly the labeled data. The displacement field is predicted

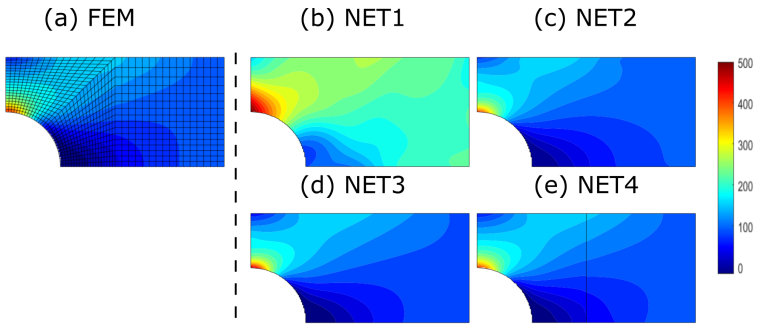


Figure 20: Cylindrical panel with cutout: distribution of the internal force resultant N_{xx}: (a) FE, (b) NET1, (c) NET2, (d) NET3, (e) NET4.

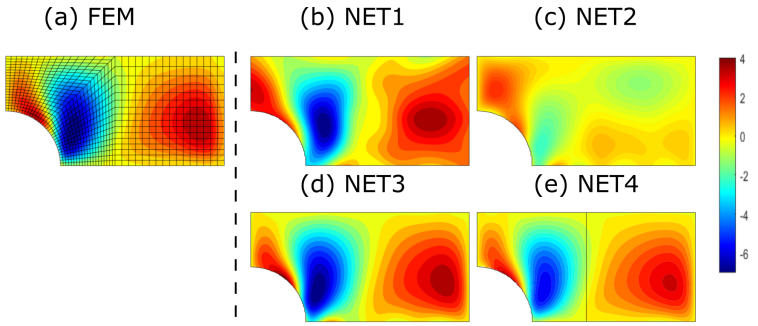


Figure 21: Cylindrical panel with cutout: distribution of the bending moment M_{xx}: (a) FE, (b) NET1, (c) NET2, (d) NET3, (e) NET4.

correctly, but there is clearly no guarantee that derivative-related quantities, such as deformations and stresses, are also captured appropriately. While improvements could be achieved by increasing the amount of labeled data, this approach would be impractical, as data availability is often scarce.

The solution predicted by NET2 is depicted in Figures 19(c)-21(c). As seen from the contours, NET2 provides an accurate estimation of the membrane response, but a poor approximation for the bending part of the solution. The magnitude of the errors is available in Table 4, where bending-related quantities, w and M_{xx}, are seen to be the ones exhibiting the largest values.

These results are explained by the limited representation capabilities of NET2. Indeed, this network architecture has a single hidden layer restricted to $N_{\mathrm{n}} = 400$ neurons. As a

Table 4: Comparison of different network architectures and training algorithms (t_{NET2}=0.9 s).

Architecture	N_{n}	N		$\mathbb{L}_2$					t/t_{NET2}
		N_{u}	N_{c}	u	v	w	N_{xx}	M_{xx}	
NET1 (GBL)	400	965	0	0.1162	0.1691	0.7287	8.596	3.9646	639.5128
NET2 (ELM)	400	0	900	1.9963	6.1111	11.6284	0.7905	10.3400	1.0000
NET3 (ELM)	400	17	900	0.0689	0.6059	0.2153	0.4780	1.5347	1.0714
NET4 (ELM)	300+100	0	937	0.7145	1.3325	2.6796	0.3503	2.3969	0.9721

consequence, the network tends to minimize the error of the residual associated with the dominant part of the response, which is the membrane one, while reducing its effectiveness in minimizing the flexural one.

A significant improvement is noted for NET3, whose outcomes are presented in Figures 19(d)-21(d). In this case, the contours are very close to the ones displayed by the FE analysis both for the membrane and bending responses. The reduced magnitude of the predicted displacement and stress fields are clearly noted by inspection of Table 4.

NET3 is an example of *grey-box* ANN, thus a comparison against the *black-* and *white-box* counterparts, NET1 and NET2, is of particular interest. The first three rows of Table 4 can be analyzed for this scope. Specifically, a drastic reduction of the errors can be seen when passing from NET1 to NET3, despite the same number of neurons is considered. The mathematical model embedded into NET3 allows fewer sampled data N_{u} to obtain accurate results. By reversing the perspective, the comparison against NET2 reveals that the *white-box* approach can be greatly improved providing as few as 17 labeled points, as done in NET3. The synergy between model information and sampled data is clearly established by these results.

The predictions due to NET4 are available in Figures 19(e)-21(e). This network architecture aims at improving the solution's quality of NET2 by dividing the domain into smaller regions. The errors in Table 4 reveal noticeable improvements with respect to the single-domain counterpart, i.e. NET2, demonstrating the improvements due to a split of the

domain into smaller regions. It is worth highlighting that these improvements are obtained using the same number of neurons N_{n}. The approach of NET4 offers a potential as a mean for solving elasticity problems, where the domain can be inherently understood as an assembly of subportions. This aspect is further investigated in the next section.

A final consideration regards the time for training required by the four networks presented earlier. In particular, the overall learning time is reported in the last column of Table 4, where the computational time is normalized with respect to the time t_{NET2}, which is the time to train NET2. The results clearly highlight the speedup due to ELM with respect to the GBL method. The three ELM-based networks, NET2 to NET4, display similar training times, with slight differences ascribable to the dimension and sparsity patterns of the coefficient matrix $\mathbf{L}$, as illustrated in Figure 22. In particular, the subdomain approach of NET4 is responsible for a larger degree of sparsity, which is indeed associated with a faster solution of the least-square problem.

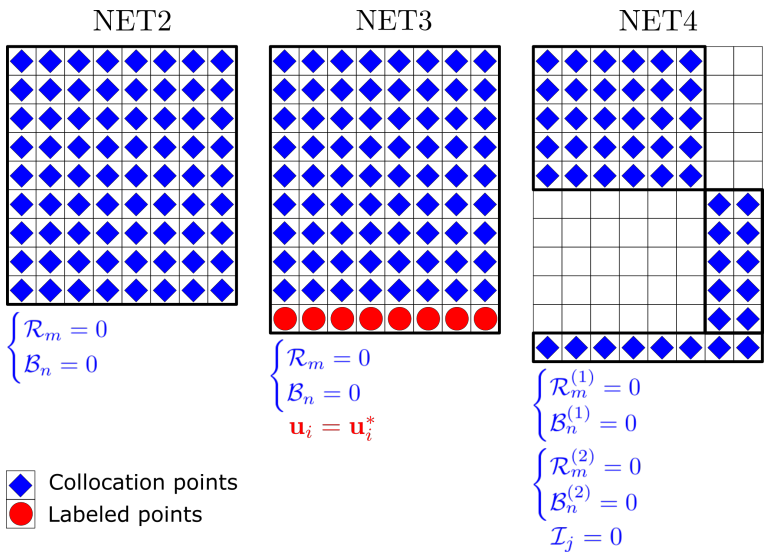


Figure 22: Distribution of the coefficient matrix $\mathbf{L}$ for different networks architectures.

5.4 Free vibrations of a stiffened panel

As shown in the previous section, the adoption of multiple subnetworks provides an effective mean for improving the representation capabilities of the network. To further demonstrate the potential of this strategy, the analysis of a stiffened panel, a configuration commonly

used in aerospace load-bearing components, is presented here.

The structure is characterized by planar dimensions of 100×100 mm^2, radius of curvature $R = 500$ mm and a single blade stiffener with height h=10 mm. Three subdomain are considered for representing the two portions of skin and the stringer. A sketch of the structure is presented in Figure 23, where the local reference systems are reported as well.

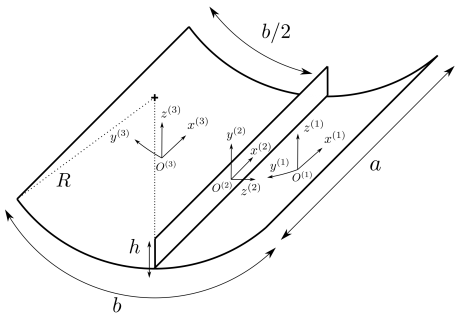


Figure 23: Stiffened panel geometry and local reference systems.

The structure is made of composite material, whose mechanical properties are $E_{11} = 150$ GPa, $E_{22} = 9$ GPa, $G_{12} = 5$ GPa, $\nu_{12} = 0.32$ and $\rho = 1500$ kg/m^3. The skin has variable stiffness layup $\left[90 + \mathbf{T}^{(1)}/\mathbf{T}^{(1)}\right]_s$ in the subdomain $x^{(1)} \in \left[-a/2, a/2\right]$, $y^{(1)} \in \left[-b/4, a/4\right]$, and $\left[90 + \mathbf{T}^{(3)}/\mathbf{T}^{(3)}\right]_s$ in the subdomain $x^{(3)} \in \left[-a/2, a/2\right]$, $y^{(3)} \in \left[-b/4, a/4\right]$, where:

$$\mathbf{T}^{(1)} = \begin{bmatrix} 10 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{T}^{(3)} = \begin{bmatrix} 0 \\ 10 \end{bmatrix} \tag{42}$$

The stringer is layered with a straight-fiber, cross-ply layup $[0/90]_s$. The ply thickness is equal to 1.25 mm for all the laminates composing the structure.

Owing to the three-subdomain representation, three networks are considered for the stringer domain and the two skin portions. The three networks share the same shallow architecture with 2, 120 and 3 neurons in the input, hidden, and output layers, respectively. The learning algorithm adopted is ELM due to its superior performance in terms of time, as demonstrated in the previous section. Training is performed pursuing a *white-box* approach, where a total of $N_c = 1200$ collocation points uniformly distributed in the computational domain are considered. The governing equations for the three subdomains are expressed by

Eq. (7), the boundary conditions are summarized in Figure 24, while the interface conditions become:

$$\boldsymbol{\mathcal{I}}_{\text{con}} = \begin{cases} u^{(1)} - u^{(3)} = 0, \quad u^{(1)} - u^{(2)} = 0 \\ v^{(1)} - v^{(3)} = 0, \quad v^{(1)} + w^{(2)} = 0 \\ w^{(1)} - w^{(3)} = 0, \quad w^{(1)} - v^{(2)} = 0 \\ w^{(1)}_{,y} - w^{(3)}_{,y} = 0, \quad w^{(1)}_{,y} - w^{(2)}_{,y} = 0 \end{cases} , \boldsymbol{\mathcal{I}}_{\text{equ}} = \begin{cases} N^{(1)}_{yy} + V^{(2)}_{y} - N^{(3)}_{yy} = 0 \\ N^{(1)}_{xy} - N^{(2)}_{xy} - N^{(3)}_{xy} = 0 \\ V^{(1)}_{y} - N^{(2)}_{yy} - V^{(3)}_{y} = 0 \\ M^{(1)}_{yy} - M^{(2)}_{yy} - M^{(3)}_{yy} = 0 \end{cases} \quad \text{in} \quad \partial\Omega_{\text{int}} \tag{43}$$

The resulting algebraic problem to be solved for training the PINN is in the form of a rectangular generalized eigenvalue problem, $\mathbf{Kc} = \omega^2\mathbf{Mc}$, where ω represents the natural frequency, and the matrices $\mathbf{K}$ and $\mathbf{M}$ are obtained via substitution of the network approximation into the governing equations, boundary and interface conditions. The workflow for training, consisting of a two-step procedure, is presented in Figure 6. Firstly, the pseudoinverse of the stiffness matrix is computed, then the resulting eigenvalue problem is solved.

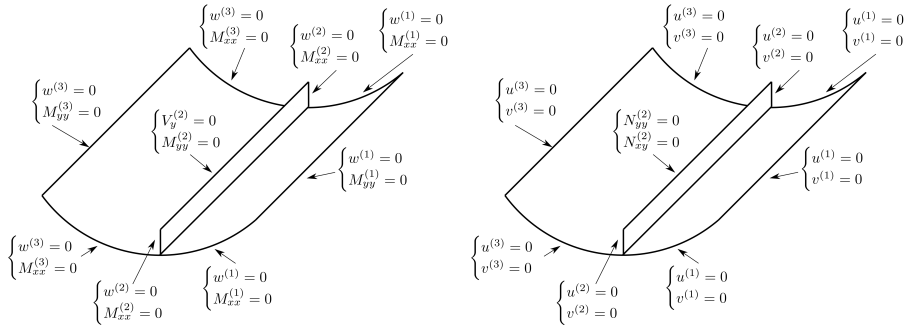


Figure 24: Stiffened panel - Boundary conditions: (a) out-of-plane, (b) in-plane.

The first 10 natural frequencies are reported in Table 5 along with the ones predicted using the FE method. Good agreement is obtained for the frequencies, with maximum percent errors below 1.4%.

By inspection of Table 5, one can note the non–monotone convergence of the PINN solutions, consistently with the findings of the previous sections. The modal shapes are presented in Figures 25 and 26. They include local skin modes – mode 4, 5, 7, 9 –, stringer modes – modes 8 and 10 – and coupled skin/stringer modes – modes 1, 2, 3, 6. All of them are predicted with an excellent degree of accuracy, with PINNs and FE results displaying sim-

Table 5: First ten natural frequencies (Hz) predicted by PINN and FEM.

	Modes									
	1	2	3	4	5	6	7	8	9	10
PINN	1014.8	1387.0	1880.0	2380.4	2447.5	2486.2	2661.6	2865.9	2932.5	3073.8
FEM	1000.6	1371.7	1861.3	2385.0	2450.7	2452.6	2653.8	2875.2	2936.3	3077.4
$\mathbb{E}_{\%}$	-1.4174	-1.1133	-1.0066	0.1930	0.1304	-1.3695	-0.2927	0.3242	0.1300	0.1192

ilar patterns. Further evidence is provided by the contour plot of the logarithmic error of Figure 27 with maximum values of the order of 10^{-1}.

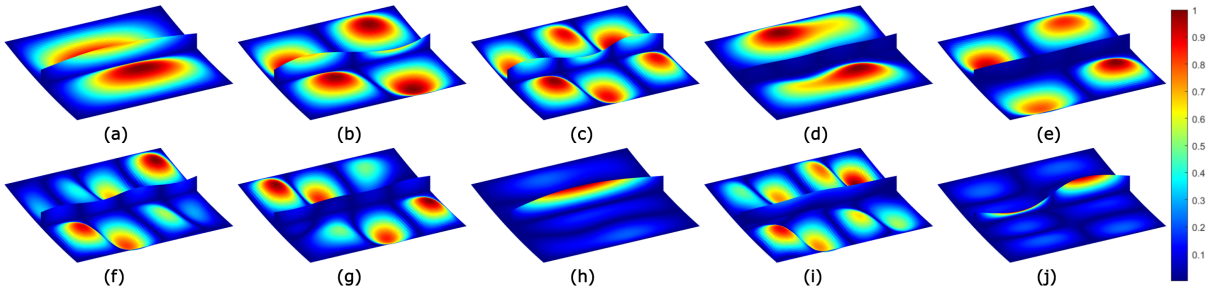


Figure 25: First 10 modal shapes of a stiffened composite VS shell predicted by PINN.

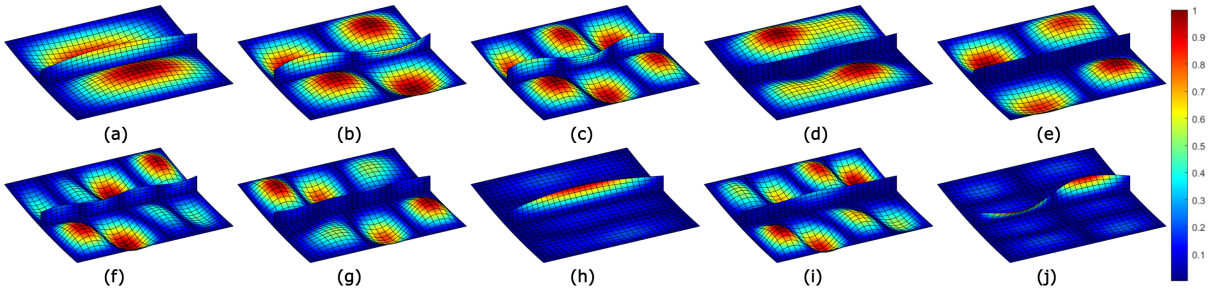


Figure 26: First 10 modal shapes of a stiffened composite VS shell predicted by FEM.

As observed by the contours, larger errors are obtained for smaller halfwave lengths, as in the case of modes 6, 7 and 9. On the contrary, patterns with larger halfwave lengths are better approximated – see the single-halfwave configurations of modes 1 and 8 – and the solution experiences smaller errors, as low as $10^{-3} - 10^{-2}$.

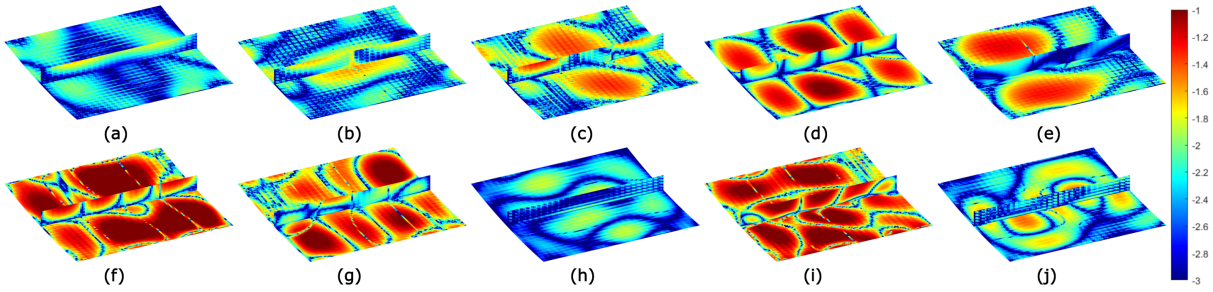


Figure 27: Error distribution for the first 10 modal shapes of a stiffened composite VS shell predicted by PINN.

5.5 Model parameter identification from a static response

In this final section, PINNs and ELM are applied for solving an inverse problem consisting in the identification of a variable stiffness plate layup to meet a known target static response. The plate's geometry is defined by the nondimensional parameters $a/b = 1$ and $a/h = 250$, while its elastic properties are the ones reported in Section 5.1. A sinusoidally distributed pressure is considered, while the edges of the plate are fully clamped.

The stacking sequence to be identified consists of a four-ply, symmetric, variable-stiffness layup $\left[\mathbf{T}^{(1)}/\mathbf{T}^{(2)}\right]_{\mathrm{s}}$ where the matrices defining the layup are:

$$\mathbf{T}^{(1)} = \left[+28.51, +44.17, +33.34\right] \quad \text{and} \quad \mathbf{T}^{(2)} = \left[-32.13, -45.86, -29.82\right] \tag{44}$$

According to Eq. (44), the fibers are allowed to vary along the x direction with a parabolic distribution defined on the basis of Eq. (6).

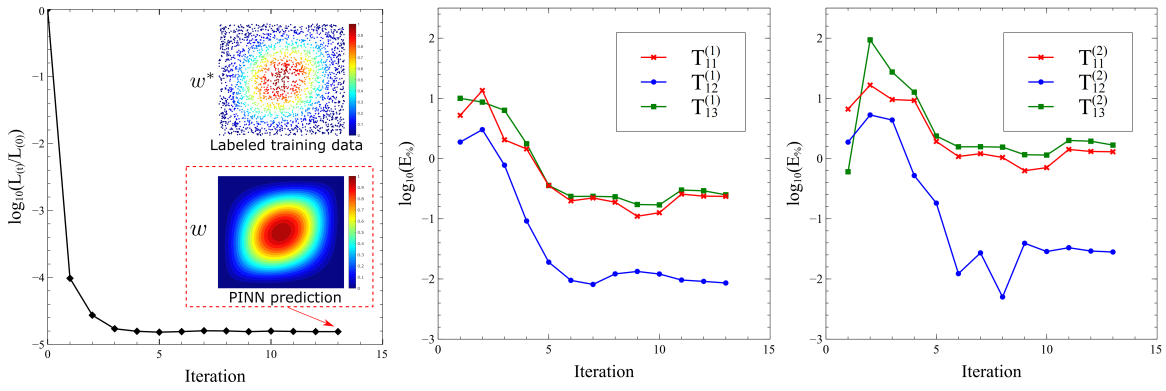
The normalized static response of the plate $\{\mathbf{x}_i, \mathbf{u}_i^*\}$ is available in $N_{\mathrm{u}} = 2500$ points, randomly distributed across the domain. These labeled data were generated via finite element analysis.

The identification process is performed using a single-hidden layer PINN with 1000 hidden neurons. Training is carried out by considering the loss function as per Eq. (15), where a uniform grid of $N_{\mathrm{c}} = 20 \times 20$ collocation points is used along with the N_{u} labeled points. The available physics-based information, which is imposed at collocation points, consists of the governing equations of Eq. (29) and the fiber path given by Eq. (6). Therefore, the unknown model parameters are:

$$\mathbf{\Lambda} = \left[T_{11}^{(1)}, T_{12}^{(1)}, T_{13}^{(1)}, T_{11}^{(2)}, T_{12}^{(2)}, T_{13}^{(2)}\right]^{\mathrm{T}} \tag{45}$$

where $T_{mn}^{(p)}$ are the interpolating angles at the ply p. The training process is initialized by assigning random values to all the network biases and weights in the range $[-1, 1]$. The initial values for the unknown angles are taken as $\mathbf{\Lambda}_{(0)} = [30, 45, 30, -30, -45, -30]^{\mathrm{T}}$. These values can be understood as the angles defining the nominal layup, which are clearly different from the actual ones due to the manufacturing process.

The results of the identification after 13 iterations are displayed in Figure 28 in terms of normalized loss function $\mathcal{L}_{(t)}/\mathcal{L}_{(0)}$ and relative percent error $\mathbb{E}_{\%}\left[T_{mn}^{(p)}\right]$, where $\mathcal{L}_{(0)}$ is the value of the loss function at iteration 0.



(a) (b) (c)

Figure 28: Identification process and evolution of: (a) loss function, (b)-(c) relative error percentage of the identified parameters throughout the learning process.

From Figure 28(a), one can see the uniform decrease of the loss function throughout the iterative process. At convergence, the final value of the loss function is $\mathcal{L}$=4.4 $\times 10^{-3}$. The corresponding static response is also reported in Figure 28(a) along with the labeled data used for the identification.

By inspection of Figures 28(b) and 28(c), one can observe the superior convergence properties of the angles at the plate center, i.e. $T_{12}^{(p)}$, with respect to the ones at the edges, i.e. $T_{11}^{(p)}$ and $T_{13}^{(p)}$. Furthermore, the convergence rate is seen to be dependent on the ply position in the stack: the second ply (Ply 2), which is closer to the midsurface, tends to exhibit slower convergence owing to its smaller contribution to the laminate bending stiffness than Ply 1. These results are further highlighted in Table 6, where the relative percent errors are

reported for the interpolation angles obtained at the end of the process.

Table 6: Identified parameters and corresponding percent errors at the end of the learning process.

	Ply 1			Ply 2		
	$T_{11}^{(1)}$	$T_{12}^{(1)}$	$T_{13}^{(1)}$	$T_{11}^{(2)}$	$T_{12}^{(2)}$	$T_{13}^{(2)}$
Exact	+28.5100	+44.1700	+33.3400	-32.1300	-45.8600	-29.8200
Identified	+28.4554	+44.1662	+33.2669	-32.4715	-45.8543	-30.2576
$\mathbb{E}_{\%}$	0.1916	0.0085	0.2192	1.0628	0.0125	1.4674

As seen, the percent errors obtained for the central angles $T_{12}^{(p)}$ are two orders of magnitude smaller with respect to angles at the edges. In addition, the errors associated with Ply 1 are one order of magnitude smaller than the errors for Ply 2.

6 Conclusions

This work presented a framework based on Physics-Informed Neural Network (PINN) for solving plate and shell problems in linear elasticity, as well as for performing parameter identification of mathematical models. The approach combines the features of PINNs with a procedure relying upon Extreme Learning Machine (ELM) to achieve improved training speed.

Parametric studies are conducted to address the effects of different network configurations and hyperparameters. It is shown that accurate solutions can be achieved for static, free vibration and buckling problems using relatively few collocation points for training. The number of neurons must be large enough to guarantee reduced errors, but has to be bounded to prevent ill-conditioning issues that may affect the solving matrices. The analysis of different grid distributions reveals that random grids may sometimes provide smaller errors, but organized grids tend to be more robust for a wider range of network configurations. Overall, a certain degree of tuning is necessary for defining the network architecture and its parameters. However, the method displays good robustness, and wide class of problems can be analyzed with no need to perform trial-and-errors procedures at any time.

As demonstrated, white-, black- or gray-box approaches can be considered, meaning that the framework can be used as a PDE solver, as a function approximator starting from available data, or a combination of both. The results illustrate the potential of this latter strategy, where labeled data and underlying governing equations are successfully combined to perform data-driven solution and data-driven identification of differential problems. Within the proposed PINN/ELM approach, a domain decomposition is proposed as an effective way to maximize the network performance. This can be done by reducing the number of neurons where the solution is more regular, and by increasing it where more complex responses are expected. In addition, the subdomain approach is naturally extended to consider structures composed by multiple shell and plate elements, such as in the case of stiffened panels. Overall, the ELM training offers drastic reduction of the training time with respect to GBL-based ones. The time for the analysis is reduced and comparable with typical FE solution procedures. In addition, no mesh needs to be generated, so the models are created on the fly.

The extension to static nonlinear analysis and higher-order structural theories is the subject of future investigations.

Acknowledgements

The authors would like to thank Ministero dell'Istruzione, dell'Università della Ricerca for funding this research under PRIN 2017 program.

References

[1] S.S. Rao. *The Finite Element Method in Engineering.* Butterworth-heinemann, Burlington, MA, USA, 2017.

[2] G.R. Liu and Y.T. Gu. *An Introduction to Meshfree Methods and Their Programming.* Springer, Dordrecht, The Netherlands, 2005.

[3] Y. LeCun, Y. Bengio, and G. Hilton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[4] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, and J.M. Siskind. Automatic differenti-

ation in machine learning: a survey. *Journal of Machine Learning Research*, 18:1–43, 2018.

[5] M. Petrolo and E. Carrera. Selection of element-wise shell kinematics using neural networks. *Computers & Structures*, 244:106425, 2021.

[6] T. Yamaguchi and H. Okuda. Zooming method for FEA using a neural network. *Computers & Structures*, 247:106480, 2021.

[7] X. Liu, F. Tao, and W. Yu. A neural network enhanced system for learning nonlinear constitutive law and failure initialization criterion of composites using indirectly measurable data. *Composite Structures*, 252:112658, 2020.

[8] H. Jiang, Z. Nie, R. Yeo, A.B. Farimani, and L.B. Kara. StreeGAN: a generative deep learning model for two-dimensional stress distribution prediction. *Journal of Applied Mechanics*, 88(5):051005, 2021.

[9] C. Bisagni and L. Lanzi. Post-buckling optimisation of composite stiffened panels using neural networks. *Composite Structures*, 58(2):237–247, 2002.

[10] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar. Integrating physics-based modeling with machine learning: a survey. *arXiv Preprint*, 2003.04919, 2020.

[11] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[12] D. Zhang, L. Lu, L. Guo, and G.E. Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[13] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes. A deep learning framework for solution and discovery in solid mechanics. *arXiv preprint*, arXiv:2003.02751, 2020.

[14] L. Borkowski, C. Sorini, and A. Chattopadhyay. Recurrent neural network-based multiaxial plasticity model with regularization for physics-informed constraints. *Computers & Structures*, 258(1):106678, 2022.

[15] E. Kharazmi, Z. Zhang, and G.E. Karniadakis. Variational physics-informed neural networks for solving partial differential equations. *arXiv Preprint*, 1912.00873, 2019.

[16] W. Li, M.Z. Bazant, and J. Zhu. A physics-guided neural network framework for elastic plates: comparison of governing equations-based and energy-based approaches. *Computer Methods in Applied Mechanics and Engineering*, 383:113933, 2021.

[17] A.D. Jagtap, K. Kawaguchi, and G.E. Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.

[18] C. Anitescu, E. Atroshchenko, N. Alajlan, and T. Rabczuk. Artificial neural network methods for the solution of second order boundary value problems. *Computers, Materials and Continua*, 59(1):345–359, 2019.

[19] A.D. Jagtap and G.E. Karniadakis. Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020.

[20] V. Dwivedi and B. Srinivasan. Physics informed extreme learning machine (PIELM) - a rapid method for the numerical solution of partial differential equations. *Neurocomputing*, 391:96–118, 2020.

[21] S. Goswami, C. Anitescu, S. Chakraborty, and T. Rabczuk. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics*, 106:102447, 2020.

[22] S. Chakraborty. Transfer learning based multi-fidelity physics informed deep neural network. *Journal of Computational Physics*, 426:109942, 2021.

[23] G.B. Huang, Q.Y. Zhu, and C.K. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1–3):489–501, 2006.

[24] E. Schiassi, R. Furfaro, C. Leake, M. De Florio, H. Johnston, and D. Mortari. Extreme theory of functional connections: a fast physics-informed neural network method for solving ordinary and partial differential equations. *Neurocomputing*, 457:334–356, 2021.

[25] H. Kraus. *Thin Elastic Shells*. John Wiley & Sons, 1967.

[26] Z. Gürdal and R. Olmedo. Composite laminates with spatially varying fiber orientations: variable stiffness panel concept. In 33^{rd} *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Material Conference*, Dallas, TX, April 13–15 1992.

[27] Z. Gürdal and R. Olmedo. In-plane response of laminates with spatially varying fiber orientations-variable stiffness concept. *AIAA Journal*, 31(4):751–758, 1993.

[28] Z. Wu, P.M. Weaver, G. Raju, and B.C. Kim. Buckling analysis and optimisation of variable angle tow composite plates. *Thin-Walled Structures*, 60:163–172, 2012.

[29] P. Ribeiro, H. Akhavan, A. Teter, and J. Warmiński. A review on the mechanical behaviour of curvilinear fibre composite laminated panels. *Journal of Composite Materials*, 48(22):2761–2777, 2013.

[30] A. Pagani and A.R. Sanchez-Majano. Influence of fiber misalignments on buckling performance of variable stiffness composites using layerwise models and random fields. *Mechanics of Advanced Materials and Structures*, pages 1–16, 2020.

[31] J.N. Reddy. *Mechanics of Laminated Composite Plates and Shells: Theory and Analysis*. CRC Press, Boca Raton, 2004.

[32] K.M. Liew and C.M. Wang. pb-2 Rayleigh-Ritz method for general plate analysis. *Engineering Structures*, 15(1):55–60, 1993.

[33] R. Vescovini, V. Oliveri, D. Pizzi, L. Dozio, and P.M. Weaver. A semi-analytical approach for the analysis of variable-stiffness panels with curvilinear stiffeners. *International Journal of Solids and Structures*, 2019.

[34] R.M. Jones. *Mechanics of Composite Materials*. CRC Press, 1998.

[35] G.J. Simitses and J. Giri. Buckling of rotationally restrained orthotropic plates under uniaxial compression. *Journal of Composite Materials*, 11(3):345–364, 1977.

[36] C.W. Bert and M. Malik. Differential quadrature method in computational mechanics: A review. *Applied Mechanics Reviews*, 49(1):1–28, 1996.

[37] C. Shu. *Differential Quadrature and its Application in Engineering.* Springer Science & Business Media, 2012.

[38] I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987—1000, 1998.

[39] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[40] E. Weinan and B. Yu. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[41] J. Berg and K.Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

[42] V. Dwivedi and B. Srinivasan. Physics informed extreme learning machine (PIELM) – a rapid method for the numerical solution of partial differential equations. *Neurocomputing*, 391:96–118, 2020.

[43] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. neural networks. *Neural Networks*, 2(5):359–366, 1989.

[44] S.S. Haykin. *Neural Networks and Learning Machines.* Pearson, Upper Saddle River, NJ, USA, 2009.

[45] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[46] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

7 Appendix

Differential operators

The matrix of partial differential operators $\boldsymbol{\mathcal{K}}(\cdot)$ is defined as follows:

$$\boldsymbol{\mathcal{K}}(\cdot) = \begin{bmatrix} \kappa_1(\cdot) & \kappa_2(\cdot) & \kappa_3(\cdot) \\ \kappa_4(\cdot) & \kappa_5(\cdot) & \kappa_6(\cdot) \\ \kappa_7(\cdot) & \kappa_8(\cdot) & \kappa_9(\cdot) \end{bmatrix} \tag{46}$$

where:

$$\begin{aligned} \kappa_1(\cdot) =& A_{11}(\cdot)_{,xx} + 2A_{16}(\cdot)_{,xy} + A_{66}(\cdot)_{,yy} + \\ &+ \left(A_{11,x} + A_{16,y}\right)(\cdot)_{,x} + \left(A_{16,x} + A_{66,y}\right)(\cdot)_{,y} \end{aligned} \tag{47}$$

$$\begin{aligned} \kappa_2(\cdot) =& A_{16}(\cdot)_{,xx} + \left(A_{12} + A_{66}\right)(\cdot)_{,xy} + A_{26}(\cdot)_{,yy} + \\ &+ \left(A_{16,x} + A_{66,y}\right)(\cdot)_{,x} + \left(A_{12,x} + A_{26,y}\right)(\cdot)_{,y} \end{aligned} \tag{48}$$

$$\begin{aligned} \kappa_3(\cdot) =& - B_{11}(\cdot)_{,xxx} - 3B_{16}(\cdot)_{,xxy} - \left(B_{12} + 2B_{66}\right)(\cdot)_{,xyy} - B_{26}(\cdot)_{,yyy} + \\ &- \left(B_{11,x} + B_{16,y}\right)(\cdot)_{,xx} - 2\left(B_{16,x} + B_{66,y}\right)(\cdot)_{,xy} - \left(B_{12,x} + B_{26,y}\right)(\cdot)_{,yy} + \\ &- \frac{1}{R}\left(A_{12}(\cdot)_{x} + A_{26}(\cdot)_{y} + A_{12,x}(\cdot) + A_{26,y}(\cdot)\right) \end{aligned} \tag{49}$$

$$\begin{aligned} \kappa_4(\cdot) =& A_{16}(\cdot)_{,xx} + \left(A_{12} + A_{66}\right)(\cdot)_{,xy} + A_{26}(\cdot)_{,yy} + \\ &+ \left(A_{16,x} + A_{12,y}\right)(\cdot)_{,x} + \left(A_{66,x} + A_{26,y}\right)(\cdot)_{,y} \end{aligned} \tag{50}$$

$$\begin{aligned} \kappa_5(\cdot) =& A_{66}(\cdot)_{,xx} + 2A_{26}(\cdot)_{,xy} + A_{22}(\cdot)_{,yy} + \\ &+ \left(A_{66,x} + A_{26,y}\right)(\cdot)_{,x} + \left(A_{26,x} + A_{22,y}\right)(\cdot)_{,y} \end{aligned} \tag{51}$$

$$\begin{aligned} \kappa_6(\cdot) =& - B_{16}(\cdot)_{,xxx} - \left(B_{12} + 2B_{66}\right)(\cdot)_{,xxy} - 3B_{26}(\cdot)_{,xyy} - B_{22}(\cdot)_{,yyy} + \\ &- \left(B_{16,x} + B_{12,y}\right)(\cdot)_{,xx} - 2\left(B_{66,x} + B_{26,y}\right)(\cdot)_{,xy} - \left(B_{26,x} + B_{22,y}\right)(\cdot)_{,yy} + \\ &- \frac{1}{R}\left(A_{26}(\cdot)_{,x} + A_{22}(\cdot)_{,y} + A_{26,x}(\cdot) + A_{22,y}(\cdot)\right) \end{aligned} \tag{52}$$

$$\begin{aligned}\kappa_7(\cdot) =& B_{11}(\cdot)_{,xxx} + 3B_{16}(\cdot)_{,xxy} + (B_{12}+2B_{66})(\cdot)_{,xyy} + B_{26}(\cdot)_{,yyy} + \\ &+ 2\left(B_{11,x}+B_{16,y}\right)(\cdot)_{,xx} + 2\left(2B_{16,x}+B_{66,y}+B_{12,y}\right)(\cdot)_{,xy} + 2\left(B_{66,x}+B_{26,y}\right)(\cdot)_{,yy} + \\ &+ \left(B_{11,xx}+2B_{16,xy}+B_{12,yy}\right)(\cdot)_{,x} + \left(B_{16,xx}+2B_{66,xy}+B_{26,yy}\right)(\cdot)_{,y}\end{aligned} \tag{53}$$

$$\begin{aligned}\kappa_8(\cdot) =& B_{16}(\cdot)_{,xxx} + (B_{12}+2B_{66})(\cdot)_{,xxy} + 3B_{26}(\cdot)_{,xyy} + B_{22}(\cdot)_{,yyy} + \\ &+ 2\left(B_{16,x}+B_{66,y}\right)(\cdot)_{,xx} + 2\left(B_{12,x}+2B_{26,y}+B_{66,x}\right)(\cdot)_{,xy} + 2\left(B_{26,x}+B_{22,y}\right)(\cdot)_{,yy} + \\ &+ \left(B_{16,xx}+2B_{66,xy}+B_{26,yy}\right)(\cdot)_{,x} + \left(B_{12,xx}+2B_{26,xy}+B_{22,yy}\right)(\cdot)_{,y}\end{aligned} \tag{54}$$

$$\begin{aligned}\kappa_9(\cdot) =& -D_{11}(\cdot)_{,xxxx} - 4D_{16}(\cdot)_{,xxxy} - 2(D_{12}-2D_{66})(\cdot)_{,xxyy} + 4D_{26}(\cdot)_{,xyyy} - D_{22}(\cdot)_{,yyyy} + \\ &- 2\left(D_{11,x}+D_{16,y}\right)(\cdot)_{,xxx} - 2\left(3D_{16,x}+2D_{66,y}+D_{12,y}\right)(\cdot)_{,xxy} - 2\left(D_{12,x}+3D_{26,y}+2D_{66,x}\right)(\cdot)_{,xyy} + \\ &- 2\left(D_{26,x}+D_{22,y}\right)(\cdot)_{,yyy} - \left(D_{11,xx}+2D_{16,xy}+D_{12,yy}\right)(\cdot)_{,xx} - 2\left(D_{16,xx}+2D_{66,xy}+D_{26,yy}\right)(\cdot)_{,xy} + \\ &- \left(D_{12,xx}+2D_{26,xy}+D_{22,yy}\right)(\cdot)_{,yy} - \frac{1}{R}\left(B_{12}(\cdot)_{,xx} + 2B_{26}(\cdot)_{,xy} + B_{22}(\cdot)_{,yy}\right) \\ &+ \left[2\left(B_{12,x}+B_{26,y}\right)(\cdot)_{,x} + 2\left(B_{26,x}+B_{22,y}\right)(\cdot)_{,y} + \left(B_{12,xx}+2B_{26,xy}+B_{22,yy}\right)(\cdot)\right]\end{aligned} \tag{55}$$

The matrix of partial differential operators $\boldsymbol{\mathcal{M}}(\cdot)$ is defined as follows:

$$\boldsymbol{\mathcal{M}}(\cdot) = \begin{bmatrix} I_0(\cdot) & 0(\cdot) & -I_1(\cdot)_{,x} \\ 0(\cdot) & I_0(\cdot) & -I_1(\cdot)_{,y} \\ I_1(\cdot)_{,x} & I_1(\cdot)_{,y} & I_0(\cdot) - I_2\left((\cdot)_{,xx} + (\cdot)_{,yy}\right) \end{bmatrix} \tag{56}$$

The matrix of partial differential operators $\boldsymbol{\mathcal{G}}(\cdot)$ is defined as follows:

$$\boldsymbol{\mathcal{G}}(\cdot) = \begin{bmatrix} 0(\cdot) & 0(\cdot) & 0(\cdot) \\ 0(\cdot) & 0(\cdot) & 0(\cdot) \\ 0(\cdot) & 0(\cdot) & \bar{N}_{xx}(\cdot)_{,xx} + 2\bar{N}_{xy}(\cdot)_{,xy} + \bar{N}_{yy}(\cdot)_{,yy} \end{bmatrix} \tag{57}$$

Navier solutions

Deflected shape due to sinusoidal transverse pressure $q_z = \hat{q}_z \sin\left(\frac{\pi x}{a}\right) \sin\left(\frac{\pi y}{b}\right)$:

$$w^{\text{bend,ext}} = \frac{\hat{q}_z b^4}{\pi^4 \left(D_{11} r^4 + 2\left(D_{12} + 2D_{66}\right) r^4 + D_{22}\right)} \sin\left(\frac{\pi x}{a}\right) \sin\left(\frac{\pi y}{b}\right) \tag{58}$$

Natural frequencies:

$$\omega_{mn}^{\text{ext}} = \sqrt{\frac{\pi^4}{\tilde{I}_0 b^4} \left(D_{11} m^4 r^4 + 2\left(D_{12} + 2D_{66}\right) m^2 n^2 r^2 + D_{22} n^4\right)} \tag{59}$$

Buckling multiplier for uniform biaxial compression, i.e. $\overline{N}_{xx} = \overline{N}_{yy}$ and $\overline{N}_{xy} = 0$:

$$\lambda_{mn}^{\text{ext}} = \frac{D_{11}\alpha^4 + 2\left(D_{12} + 2D_{66}\right)\alpha^2\beta^2 + D_{22}\beta^4}{\left(\alpha^2 + \beta^2\right)} \tag{60}$$

where $r = b/a$ is the aspect ratio of the plate, $\tilde{I}_0 = I_0 + I_2\left[\alpha^2 + \beta^2\right]$, $\alpha = m\pi/a$, $\beta = n\pi/b$, while m and n are the number of haft-waves in the x and y direction, respectively.