modelling

MDPI

Article

Physics-Informed Neural Network for Solving a One-Dimensional Solid Mechanics Problem

Vishal Singh [1], Dineshkumar Harursampath [2], Sharanjeet Dhawan [3], Manoj Sahni [4], Sahaj Saxena [5] and Rajnish Mallick [1,*]

[1] Department of Mechanical Engineering, Thapar Institute of Engineering and Technology, Patiala 147004, Punjab, India; vsingh10_be21@thapar.edu
[2] Department of Aerospace Engineering, Indian Institute of Science, Bengaluru 560012, Karnataka, India; dineshkumar@iisc.ac.in
[3] Department of Mathematics, CCSHAU COA, Bawal 123501, Haryana, India; sharanjeet@hau.ac.in
[4] Department of Mathematics, Pandit Deendayal Energy University, Gandhinagar 382007, Gujarat, India; manoj.sahni@sot.pdpu.ac.in
[5] Department of Electrical and Instrumentation Engineering, Thapar Institute of Engineering and Technology, Patiala 147004, Punjab, India; sahaj.saxena@thapar.edu
* Correspondence: rajnish.mallick@thapar.edu

Abstract: Our objective in this work is to demonstrate how physics-informed neural networks, a type of deep learning technology, can be utilized to examine the mechanical properties of a helicopter blade. The blade is regarded as a one-dimensional prismatic cantilever beam that is exposed to triangular loading, and comprehending its mechanical behavior is of utmost importance in the aerospace field. PINNs utilize the physical information, including differential equations and boundary conditions, within the loss function of the neural network to approximate the solution. Our approach determines the overall loss by aggregating the losses from the differential equation, boundary conditions, and data. We employed a physics-informed neural network (PINN) and an artificial neural network (ANN) with equivalent hyperparameters to solve a fourth-order differential equation. By comparing the performance of the PINN model against the analytical solution of the equation and the results obtained from the ANN model, we have conclusively shown that the PINN model exhibits superior accuracy, robustness, and computational efficiency when addressing high-order differential equations that govern physics-based problems. In conclusion, the study demonstrates that PINN offers a superior alternative for addressing solid mechanics problems with applications in the aerospace industry.

Keywords: physics-informed neural network; deep neural network; artificial neural network; computational solid mechanics; partial differential equation

Citation: Singh, V.; Harursampath, D.; Dhawan, S.; Sahni, M.; Saxena, S.; Mallick, R. Physics-Informed Neural Network for Solving a One-Dimensional Solid Mechanics Problem. *Modelling* **2024**, *5*, 1532–1549. https://doi.org/10.3390/modelling5040080

Academic Editor: Josef Kiendl

Received: 5 September 2024
Revised: 4 October 2024
Accepted: 9 October 2024
Published: 18 October 2024

1. Introduction

Solid mechanics can be described as a field within applied mechanics concerned with analyzing how solid objects respond to different types of forces. It explores how materials behave under various loading conditions, offering insights into their structural integrity and performance [1]. Mechanical problems are represented using a variety of differential equations, which take different forms depending on the nature of the problem.

In recent years, computational solid mechanics has emerged as a discipline that uses numerical techniques to solve complex differential equations [2]. Solving these problems analytically is challenging and time-consuming because of the intricate equations and irregular problem domains. Over the years, various numerical techniques have been developed, such as the finite element method (FEM) [3], finite difference method (FDM) [4], element-free Galerkin method [5], and mesh-free methods [6]. Among these methods, FEM is the most widely used numerical method to solve problems in the area of solid mechanics.

In numerous cases, the problem to be solved is extensive, resulting in simulation times ranging from hours to days or even weeks. This incurs a high computational cost. If there is a requirement to change the parameter, the complete analysis must be conducted again from scratch, which is quite time-consuming [7,8].

Over the past few years, artificial neural networks (ANNs) have demonstrated remarkable performance in various fields such as image classification [9], time series forecasting [10], predictive analytics [11], genomics [12], and natural language processing [13], owing to their capacity to grasp intricate patterns and relationships from data. Artificial neural networks (ANNs) can incorporate multiple hidden layers containing neurons, granting them robust learning capabilities. This feature enables ANNs to offer an alternative method for addressing mechanics problems, diverging from conventional numerical solvers. Artificial neural networks (ANNs) have proven effective in addressing a range of challenges in fluid mechanics [14,15], fracture mechanics [16], and solid mechanics [17]. In one study, Khatir et al. (2022) [18] present a method to identify crack depth in steel beams leveraging a combination of FEM and ANNs, enhanced by the Butterfly Optimization Algorithm. Although the artificial neural network is a powerful tool, it has its limitations, such as experiencing diminished error reduction once the training process is finished, leading to suboptimal output results. Bolandi et al. (2022) [19] proposed a deep learning technique using convolutional neural networks (CNNs) to predict high-resolution stress distributions on loaded steel plates, offering a faster alternative to the traditional finite element method (FEM) and showcasing the potential of deep learning for real-time structural stress analysis. The study by Teng et al. (2022) [20] introduces a deep learning approach using convolutional neural networks (CNNs) to classify the structural state of a steel frame based on vibration signals. Le-Nguyen et al. (2022) [21] applied ANNs to predict the compressive strength of FRCM-confined concrete columns. The study conducted by Tran et al. (2022) [22] utilized three hybrid materials—gradient boosting (GB), artificial neural network (ANN), and random forest (RF)—in combination with particle swarm optimization (PSO) to predict the chloride content in concrete. Nonetheless, the performance of ANNs tends to be strongest when abundant data are available. In mechanics problems, data can be scarce, and ANNs do not incorporate the underlying physical laws of the engineering problem, resulting in reduced prediction accuracy [8,23]. Thus, the challenges associated with artificial neural networks (ANNs) motivate us to explore new ideas and methods.

One such approach widely accepted in the scientific community is a deep learning-based method known as physics-informed neural networks (PINNs) introduced by Raissi et al. [24]. PINNs represent a highly effective method for addressing problems governed by partial differential equations (PDEs). These networks are designed to directly integrate physical laws or constraints into their structure, enabling them to simulate and accurately model complex systems. The foundational component of the physics-informed neural network (PINN) framework is a multi-layer artificial neural network that is enhanced with a physics-informed loss function. This innovative loss function integrates governing differential equations, boundary conditions, initial conditions, and any available data to determine the total loss accurately. The sole distinction between an artificial neural network (ANN) and a physics-informed neural network (PINN) lies in how the loss function is implemented and calculated [25]. Artificial neural networks (ANNs) rely solely on data for learning, whereas physics-informed neural networks (PINNs) incorporate the governing equations as pre-existing knowledge [26]. PINNs can be effective in scenarios where labeled data are sparse because they utilize both the existing data as well as the inherent physical principles described in equations. Compared to ANNs, PINNs require less data. PINNs differ from traditional numerical methods like the finite difference method and finite element because they are not mesh-based. Instead, they are a mesh-free method, which allows them to handle irregular and complex geometries [24,27].

In their seminal work, Raissi et al. [24] tackled two distinct problem sets: data-driven solution and data-driven discovery within the realm of partial differential equations. Under

the data-driven solution framework, they addressed the Schrodinger Equation and the Allen–Cahn equation. Meanwhile, within the data-driven discovery approach, they delved into the Navier–Stokes equation and the Korteweg–de Vries equation. After their introduction, PINNs have been used in solving various other PDEs [28–30]. PINNs can be used for solving supervised learning problems [31] as well as unsupervised learning tasks [32]. They can also be employed for both forward and inverse problems [33]. Due to their ability to incorporate the governing equations of the problems, PINNs have been used in various fields such as fluid mechanics [34,35], heat transfer [36], healthcare [37], finance [38], and solid mechanics [39]. Several research studies have been carried out on implementing PINNs so far. In one of the studies, Haghighat et al. (2021) [39] created a PINN structure to anticipate the field variables (such as displacement and stress) associated with linear elastic and non-linear problems. In another study, Zhi et al. (2023) [40] proposed a surrogate-based physics-informed neural network (PINN) to solve representative boundary value problems based on elliptic partial differential equations (PDEs). In their study, Rao et al. (2021) [41] applied enforced initial and boundary conditions to simulate static and dynamic problems using a PINN model, which gives a mixed-variable output. One of the works carried out by J. Bai et al. (2022) [42] proposed the LSWR loss function for PINNs, which uses the Least Squares Weighted Residual (LSWR) method, solved the 2D and 3D solid mechanics problem, and showed that the performance of PINNs based on the LSWR loss function is more effective and accurate compared to PINNs utilizing either collocation or energy-based loss functions. In the study by J. Bai et al. (2023) [43], the focus was on programming methods in executing governing equations. They solved the 1-, 2-, and 3-dimensional problems by employing collocation-based and energy-based loss functions, demonstrating the effectiveness of PINN-based Computational Solid Mechanics. Abueidda et al. (2022) [44] applied PINNs to solve 3-dimensional hyperelastic problems.There is a reasonable amount of work that shows PINNs are effective in solving PDEs.

Beams are the structural elements designed to support loads that are perpendicular to the axis, and are used in various industries such as aerospace, automotive, construction, etc. In our paper, we are focusing on utilizing PINNs for a particular beam with its application in aerospace engineering. Over the years, there have been many studies utilizing PINNs to solve beam-related problems. The work by Kapoor et al. (2023) [45] used PINNs to simulate complex beam systems, solving both forward and inverse problems. The study by Faroughi et al. (2023) [46] investigates the use of mixed formulation PINNs to solve the problem of a non-uniform beam resting on an elastic foundation under arbitrary external loads. They used several different beam structures under various types of complicated boundary and foundation conditions to show the application of the proposed method. Another study by Liu et al. (2024) [47] presents a physics-informed neural network that incorporates modal transformation equations for improved performance with minimal data. A modified Kalman filter reduces noise sensitivity. The framework's effectiveness and the benefits of the KF-UI algorithm are demonstrated through experiments on a cantilever beam and a wing structure. Bazmara et al. (2023) [48] present a physics-informed neural network framework for analyzing the non-linear buckling behavior of a 3D functionally graded (FG) porous, slender beam on a Winkler–Pasternak foundation. Trinh et al. (2024) [49] present a PINN model for analyzing functionally graded thin-walled beams with doubly symmetric I- and channel-sections. The approach, based on energy methods, is used to predict vertical displacements and angles of twist in these beams. Verma et al. (2024) [50] use PINNs to simulate the behavior of a cantilever beam subjected to a uniform loading. Moreover, Mouratidou et al. (2024) [51] present a computational method for solving 2D elasticity problems in solid mechanics using an ensemble of PINNs. It addresses strain–displacement relations, equilibrium equations, and isotropic constitutive relations under various boundary conditions. There is extensive research demonstrating the effectiveness of PINNs in solving beam problems.

In our study, we analyze the mechanical characteristics (such as deflection, etc.) of a helicopter blade treated as a prismatic (constant EI) cantilever beam subjected to triangular

loading. When a lateral force is applied over a beam, the beam's longitudinal axis undergoes deformation, resulting in a curvature known as the deflection curve [1]. Understanding these mechanical behaviors is essential in designing helicopter blades, ensuring optimal flight performance and safety.

While it is well-documented that classical methods like the finite element method (FEM) are computationally expensive and time-consuming [7,8], a similar issue arises with PINNs if there are any changes in the initial parameters of the partial differential equation or boundary conditions that would require retraining the PINNs. However, the recent advantages of parametric training provide a promising solution [52]. In the paper by Choo et al. (2024) [52] the limitations of traditional PINNs are highlighted, and these involve repetitive and time-consuming training. To solve this, the authors propose a novel method called parameterized physics-informed neural networks. Other work by Anton et al. (2024) [53] uses parametric PINNs for constitutive model calibration from full-field displacement data, offering fast, near real-time evaluations. Other methods also came into the picture in recent years called neural operators [54] that map between infinite-dimensional function spaces. Different neural operators such as the Fourier Neural Operator (FNO) [55,56], Physics-Informed Neural Operator (PINO) [57,58], etc. Instead of learning a single solution, neural operators learn an operator that can map any input function (e.g., parameters or initial/boundary conditions) to its corresponding output. However, these aspects are beyond the current scope of this research and present opportunities for future work. Currently, this research is dedicated to investigating the capabilities of physics-informed neural networks (PINNs) within beam mechanics, emphasizing their applicability and significance in aerospace design and analysis.

This paper is organized as follows: Section 2 thoroughly explains the theory and architecture of ANNs and PINNs. Section 3 gives a brief overview of the problem to be solved and defines the governing equation, boundary conditions, and the formulation of the loss function for PINN. Section 4 presents a detailed exploration of the training process and a brief overview of the results obtained. Ultimately, the study is concluded in Section 5.

2. Physics-Informed Neural Network

This section provides a detailed exploration of artificial neural networks (ANNs) and physics-informed neural networks (PINNs), focusing on their theoretical frameworks and methodologies and how they are used to address challenges in computational solid mechanics.

2.1. Artificial Neural Network

Inspired by the biological neurons present in the human brain [59], the artificial neural network is a computational model that aims to replicate the functions of neurons, allowing them to learn patterns and relationships from the data inputs and make a decision or prediction based on the acquired knowledge [60].

The artificial neural network (ANN) architecture, illustrated in Figure 1, comprises several network layers, including an input layer, hidden layer(s), and an output layer. Depending on the requirements of the problem, each neural network layer can contain multiple neurons or nodes. Data or information is inputted into the neural network via the input layer and then moves forward through the network, passing through the adjacent layers, which are connected to each other via weights, biases, and activation functions [26]. Typically, an L-layer artificial neural network can be mathematically expressed as [43,61]:

$$\phi^{(0)} = x, \tag{1}$$

$$\phi^{(l)} = w^{(l)} \cdot \phi^{(l-1)} + b^{(l)}, \quad \text{for} \quad l = 1, 2, \ldots, L-1, \tag{2}$$

$$\phi^{(l+1)} = \alpha(\phi^{(l)}), \tag{3}$$

$$v = \phi^{(L)} = w^{(L-1)} \cdot \phi^{(L-1)} + b^{(L-1)}. \tag{4}$$

where x is the input vector which is fed into the input layer $\phi^{(0)} = x$; adjacent layers in the neural network are denoted as $\phi^{(l)}$ and $\phi^{(l+1)}$. The activation function is represented by α, and the final output of the artificial neural network (ANN) is denoted by v.

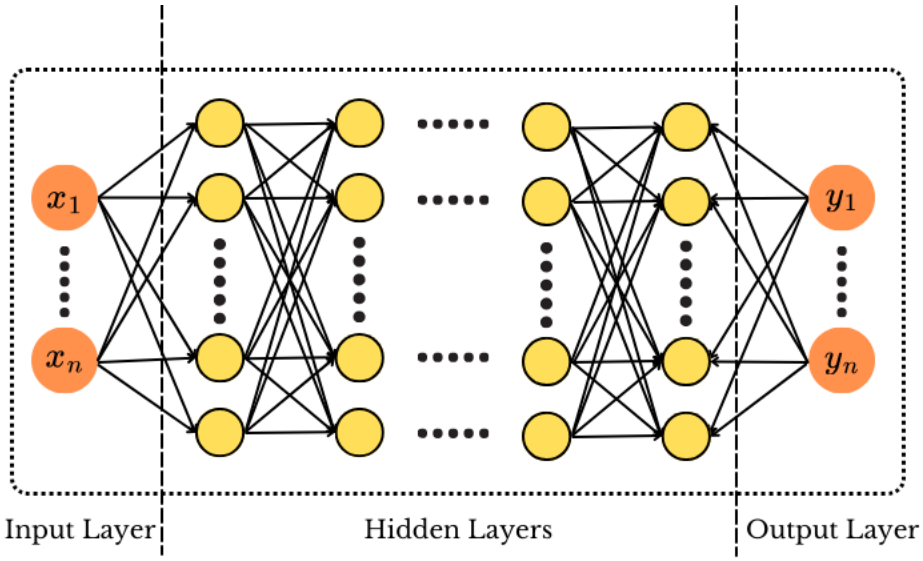


Figure 1. Artificial neural network architecture.

A neural network tries to establish a relationship between the input data (x) and output data (y) by learning an underlying function (f). This relationship is defined by a function $v = f(x, \theta)$, where θ refers to the learnable parameters of the network [62]. The network works to optimize these parameters by reducing a loss function:

$$L_{\mathrm{NN}} = \frac{1}{N} \sum_{i=1}^{N} |v - v^*|^2 \tag{5}$$

where N is the number of data points; it measures the difference between the network's predictions v and the actual values v^*. The neural network can effectively model the desired function by adjusting its parameters during the training process. Finally, the prediction or result is delivered through the output layer.

2.2. Physics-Informed Neural Networks (PINNs)

Physics-informed neural networks (PINNs), as proposed by Raissi et al. [24], offer a novel approach to solving problems governed by partial differential equations (PDEs). This technique seamlessly integrates the core principles and equations of physics into the training process of artificial neural networks (ANNs).

The general representation of a governing equation for a physical process is given by [8]:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathcal{D}(\mathbf{u}) + \mathcal{P}(\mathbf{u}). \tag{6}$$

Here, $\mathcal{D}$ denotes the non-linear differential operator, $\mathcal{P}$ represents the linear differential operator, and $\mathbf{u}$ stands for the unknown solution being analyzed, which satisfies the differential equation. The differential equation loss, L_{DE}, is as follows:

$$L_{\mathrm{DE}} = \frac{1}{N_{\mathrm{DE}}} \sum_{i=1}^{N_{\mathrm{DE}}} \left| \frac{\partial u(x_n, t_n)}{\partial t} - \mathcal{D}u(x_n, t_n) - \mathcal{P}u(x_n, t_n) \right|^2, \tag{7}$$

where (x_n, t_n) denotes the collocation points where differential equation loss is calculated and N_{DE} gives the total number of these collocation points. Additionally, the boundary and initial condition loss is given as:

$$L_{\text{BC}} = \frac{1}{N_{\text{BC}}} \sum_{i=1}^{N_{\text{BC}}} |\mathcal{B}(u(x_b, t_b))|^2 \tag{8}$$

$$L_{\text{in}} = \frac{1}{N_{\text{in}}} \sum_{i=1}^{N_{\text{in}}} |(u(x_i, 0) - u_i(x_i, 0))|^2 \tag{9}$$

where (x_b, t_b) are the boundary points, N_{BC} is the total number of boundary points, and $\mathcal{B}$ is the boundary operator corresponding to Dritchlet, Robin, periodic or Neumann boundary conditions [63]. Moreover, $(x_i, 0)$ are the initial points where the initial loss is calculated, N is the total number of initial points, and u_i is the defined initial condition for the problem. Additionally, the data loss is defined as:

$$L_{\text{data}} = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} |u(x_d, t_d) - u_{\text{es}}(x_d, t_d)|^2 \tag{10}$$

where (x_d, t_d) are the data points for training the ANN, N_{data} are the total number of these data points, and u_{es} is the exact solution of the problem.

In the PINN approach, the aim is to effectively approximate the solution $\mathbf{u}$ of a given problem using a neural network. To achieve this, the network optimizes its parameters, which include weights and biases, by minimizing a defined loss function. This loss function is designed to ensure that the network accurately represents the underlying physics of the problem while also fitting the available data. In this study, our focus is solely on the differential equation loss (L_{DE}), boundary loss (L_{BC}), and data loss (L_{data}). Therefore, the total loss is defined as follows:

$$L_{\text{total}} = L_{\text{DE}} + L_{\text{BC}} + L_{\text{Data}}, \tag{11}$$

The architecture utilized in this study, as illustrated in Figure 2, centers around the artificial neural network (ANN) as its primary component. This ANN comprises interconnected layers of artificial neurons, responsible for processing input data denoted as x and propagating information throughout the network to generate an output prediction, denoted as u. Subsequently, the output u is employed to compute derivative terms, which are obtained analytically through automatic differentiation methods [64]. These derivatives are then utilized to calculate the boundary and differential equation loss. The data loss is also directly computed based on the output u. Finally, the total loss, which requires minimization for practical training, is determined by considering all these factors.

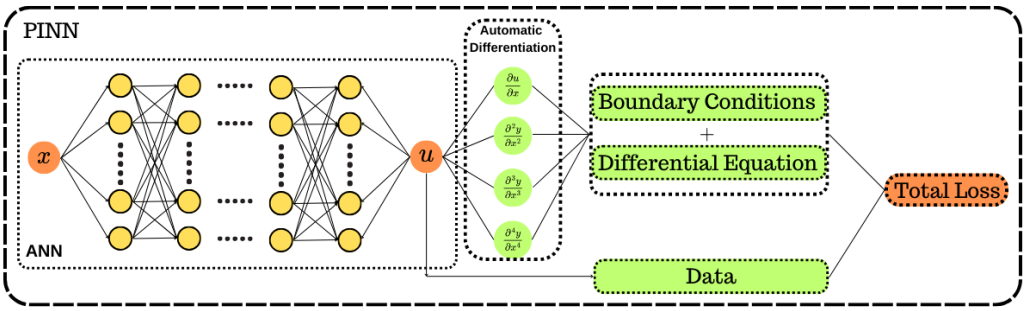


Figure 2. Physics-informed neural network architecture.

3. PINN for One-Dimensional Solid Mechanics Problem

In this section, we will discuss the problem that physics-informed neural networks (PINNs) aim to solve. We will explain the differential and exact equations, as well as the

boundary conditions that govern the problem. Furthermore, we elaborate on the formulation of the loss function integrated into PINNs for addressing this particular problem.

3.1. Problem Definition

In this demonstration, we present the implementation of a physics-informed neural network designed for solving a beam mechanics problem. We considered the helicopter blade as a cantilever beam which is fixed at point A, and B is the free end, subjected to triangular loading, as illustrated in Figure 3.

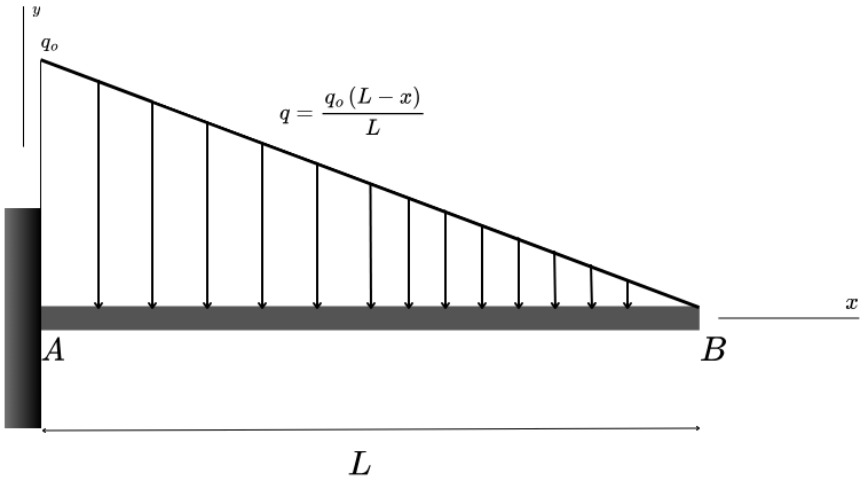


Figure 3. Cantilever beam with triangular loading.

The load intensity along the beam's length L is defined by the equation:

$$q = \frac{q_o(L-x)}{L}. \tag{12}$$

Here, q_o represents the maximum load applied at the fixed end, L denotes the length of the beam, and x signifies the position along the beam's length.

3.2. Governing Equations

For a prismatic cantilever beam, the governing fourth-order differential equation according to Euler–Bernoulli theory is given by [1]:

$$EI\frac{d^4v}{dx^4} = -q, \tag{13}$$

$$EI\frac{d^4v}{dx^4} = -\frac{q_o(L-x)}{L}. \tag{14}$$

Here, E represents Young's Modulus, I denotes the Moment of Inertia of the beam, and q stands for the load intensity. Upon integrating Equation (14) once, we obtain the expression for shear force (V) within the beam:

$$V = EI\frac{d^3v}{dx^3} = \frac{q_o(L-x)^2}{2L} + C_1. \tag{15}$$

As we know, at the free end of the beam, i.e., at point B, that $x = L$ the shear force is zero. From this boundary condition, we have the following relation:

$$\left.\frac{d^3v}{dx^3}\right|_{x=L} = 0.$$

By using this condition and using Equation (15), we obtain $C_1 = 0$. Therefore, the shear force is given by,

$$V = EI\frac{d^3v}{dx^3} = \frac{q_o(L-x)^2}{2L}. \tag{16}$$

Integrating Equation (14) twice yields the subsequent equation for the bending moment M of the beam:

$$EI\frac{d^2v}{dx^2} = \frac{-q_o(L-x)^3}{6L} + C_2. \tag{17}$$

At the free end, located at x=L, the bending moment is zero. This condition translates into the boundary condition:

$$\left.\frac{d^2v}{dx^2}\right|_{x=L} = 0. \tag{18}$$

By using the above boundary condition and Equation (17), we obtain $C_2 = 0$; therefore, the bending moment M is

$$M = EI\frac{d^2v}{dx^2} = \frac{-q_o(L-x)^3}{6L}. \tag{19}$$

Continuing with the integration of Equation (14) for the third time, we derive the equation representing the slope.

$$EI\frac{dv}{dx} = \frac{q_o(L-x)^4}{24L} + C_3, \tag{20}$$

and as the fixed support slope is zero, so

$$\left.\frac{dv}{dx}\right|_{x=0} = 0, \tag{21}$$

which gives us

$$C_3 = \frac{-q_oL^3}{24}. \tag{22}$$

After substituting the value of C_3 into Equation (20) and simplifying, we obtain the equation defining the slope as follows:

$$\frac{dv}{dx} = -\frac{q_ox}{24LEI}(4L^3 - 6L^2x + 4Lx^2 - x^3). \tag{23}$$

Integrating Equation (14) four times yields the equation for deflection, which can be expressed as follows:

$$EIv = \frac{-q_o(L-x)^5}{120L} + C_3x + C_4. \tag{24}$$

At the root of the beam, the deflection is zero; which gives the following boundary condition:

$$v|_{x=0} = 0. \tag{25}$$

By using the above boundary condition, we obtain the value of,

$$C_4 = \frac{q_oL^4}{120}. \tag{26}$$

Upon substituting the values of C_3 and C_4 and performing some simplifications, we obtain the exact equation that describes the deflection of the beam.

$$v = -\frac{q_ox^2}{120LEI}(10L^3 - 10L^2x + 5Lx^2 - x^3). \tag{27}$$

3.3. Loss-Defined

In the preceding section, we derived the governing differential equation and the boundary conditions, laying the foundation for constructing the loss function for the physics-informed neural network. The governing differential equation and boundary conditions are as follows:

$$EI\frac{d^4v}{dx^4} = -\frac{q_o(L-x)}{L}, \tag{28}$$

$$v(0) = 0, \tag{29}$$

$$v'(0) = 0, \tag{30}$$

$$v''(L) = 0, \tag{31}$$

$$v'''(L) = 0. \tag{32}$$

PINN is trained to approximate the solution to the differential equation over the boundary and collocation points, denoted as:

$$v_{\text{PINN}} \approx v, \tag{33}$$

Using Equations (28)–(32), we formulate our boundary loss and physics-based loss, which enable learning of the neural network parameters by minimizing the total loss defined as [25]:

$$L_{\text{total}} = L_{\text{DE}} + L_{\text{BC}} + L_{\text{Data}}, \tag{34}$$

where L_{DE} represents the differential equation loss:

$$L_{\text{DE}} = \frac{1}{N_{\text{DE}}}\sum_{i=1}^{N_{\text{DE}}} |EI\frac{d^4v_{\text{PINN}}(x_n)}{dx^4} + \frac{q_o(L-x)}{L}|^2. \tag{35}$$

In this expression, x_n denotes collocation points along the length of the beam for which L_{DE} is calculated, and N_{DE} represents the total number of collocation points.

As for the noundary loss L_{BC}, it can be expressed as:

$$L_{\text{BC}} = \frac{1}{N_{\text{BC}}}\sum_{i=1}^{N_{\text{BC}}}, (L_{\text{B1}} + L_{\text{B2}} + L_{\text{B3}} + L_{\text{B4}}), \tag{36}$$

$$L_{\text{B1}} = |v_{\text{PINN}}(x_b = 0) - 0|^2, \tag{37}$$

$$L_{\text{B2}} = |v'_{\text{PINN}}(x_b = 0) - 0|^2, \tag{38}$$

$$L_{\text{B3}} = |v''_{\text{PINN}}(x_b = L) - 0|^2, \tag{39}$$

$$L_{\text{B4}} = |v'''_{\text{PINN}}(x_b = L) - 0|^2. \tag{40}$$

where x_b denotes the boundary points and N_{BC} denotes the number of boundary points for the beam. L_{B1}, L_{B2}, L_{B3}, and L_{B4} correspond to the boundary conditions defined in Equations (30)–(32), with L representing the length of the beam.

The data loss, L_{Data}, measures the deviation or difference between predicted values from the exact values and is defined as:

$$L_{\text{Data}} = \frac{1}{N_{\text{Data}}}\sum_{i=1}^{N_{\text{Data}}} |v_{\text{PINN}}(x_n) - v^*(x_n)|^2, \tag{41}$$

where N_{Data} are the total number of collocation points on the beam used for calculating data loss, x_n denotes the collocation points along the length of the beam, and v_{PINN} is the predicted or approximated solution and v^* is the exact solution.

Thus, to train the PINN model and learn the neural network parameters, the total loss (L_{total}) is minimized as much as possible. By minimizing this total loss, we fine-tune the model to make more accurate predictions, essentially improving its performance.

4. Results and Discussion

We considered a one-dimensional cantilever beam of length L = 1 m, subjected to a maximum load at point A, $q_0 = 1.0$ N as shown in Figure 3. For the simplification of the problem, we assume the value of Young's modulus, $E = 1.0$ Pa, and Moment of Inertia, $I = 1.0$ Kgm^2.

In this study, we analyzed 51 collocation points spaced at intervals of 0.02 m along the length of the beam as shown in Figure 4. At the boundary points, we selected two positions: one at the fixed end ($x = 0$) and another at the free end ($x = L$) as shown in Figure 4. Thus, we have $N_{\text{De}} = N_{\text{data}} = 51$ and $N_{\text{BC}} = 2$, with collocation points x_n ranging from 0 to 1 with increments of 0.02 and the boundary points $x_b \, \epsilon \, [0.00, 1.00]$.

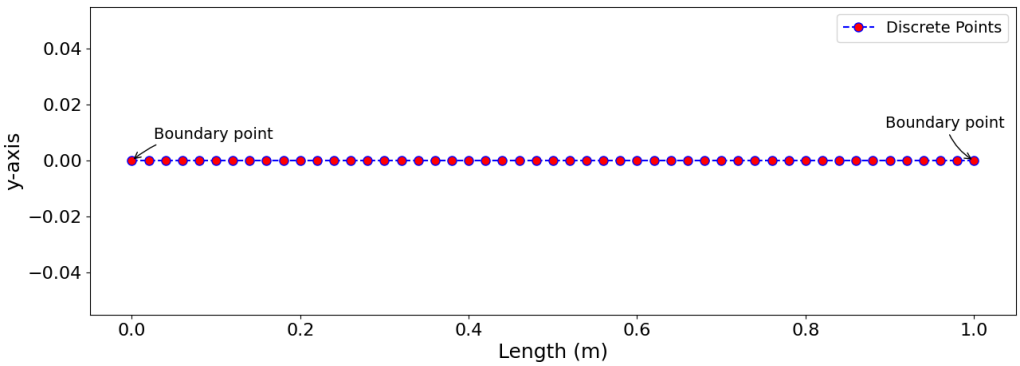


Figure 4. Points over the length of beam.

We trained the PINN model for 300 epochs at a learning rate of $\alpha = 0.001$, employing ADAM [65] as the optimizer. The model consists of an architecture with one input layer, five hidden layers, and one output layer, with each hidden layer containing 50 neurons. For the activation function, we implemented the tanh function [66] as shown in Equation (42).

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{42}$$

We also trained an artificial neural network (ANN) with an identical network architecture for comparison purposes. This includes the same number of epochs, learning rate, optimizer, and the tanh activation function, allowing for a direct comparison between the PINN model and the ANN. Both the models were developed from scratch using PyTorch [67] version 2.2.1. We have summarized all the details of our models in Table 1.

Table 1. Neural network configuration.

Parameters	Values
Epochs	300
Learning Rate	0.001
Optimizer	Adam
Input Layer	1
Hidden Layers	5
Output Layer	1
Number of Neurons	50
Activation Function	Tanh

The ANN model underwent training using the configuration outlined in Table 1. Upon completion of 300 epochs of training, a loss curve for the model was generated, visually represented in Figure 5. This curve serves as a valuable tool for assessing the model's convergence and performance throughout the training process.

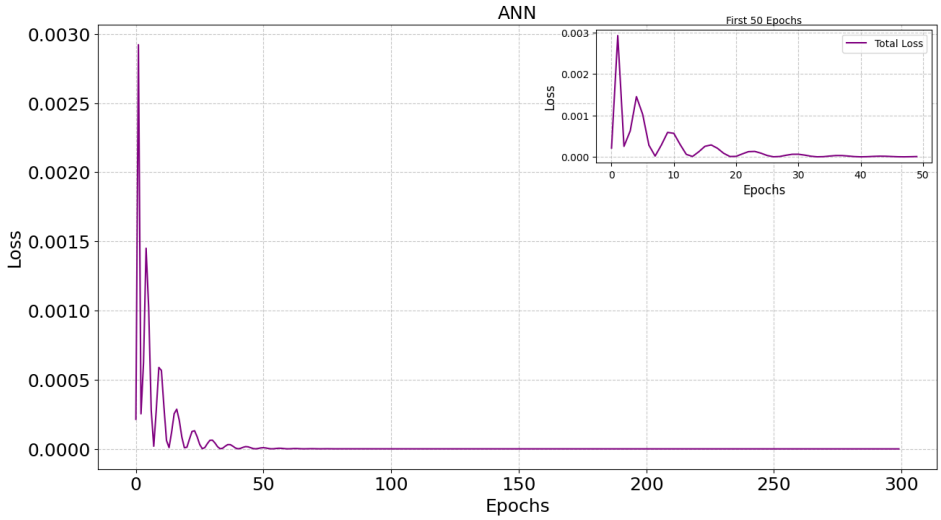


Figure 5. Loss curve for the ANN model.

Additionally, after training the PINN model for 300 epochs, we acquired the loss curve, depicted in Figure 6. This graph illustrates the convergence of various components, including PDE loss, boundary loss, data loss, and the overall total Loss.

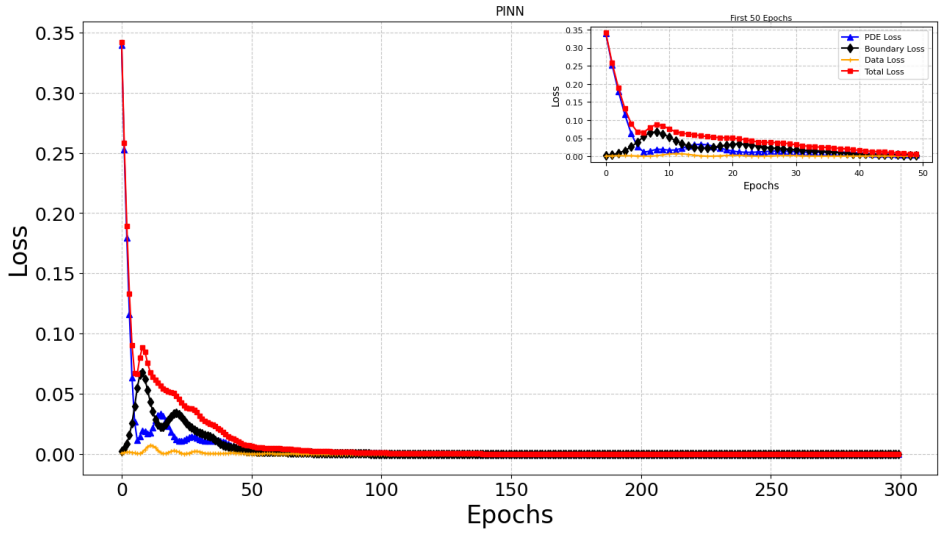


Figure 6. Loss curve for the PINN model.

Upon completing its training, the PINN model can accurately predict/approximate the deflection, slope, bending moment, and shear force along the length of a beam as shown in Figures 7 and 8. This detailed analysis enables a precise evaluation of the beam's mechanical response.

Additionally, we have calculated the Mean Squared Error (MSE) values between the predicted and the exact solution for both the models as shown in Table 2. It is calculated as the average of the squared errors as shown in Equation (43). It serves as a metric to evaluate the precision of a predictive model.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{43}$$

where n is the number of data points, $\hat{y}_i$ is the predicted value, and y_i is the actual/exact solution. A lower MSE value signifies enhanced performance, denoting that the predictions generated by the model are in closer approximation to the exact values.

As observed from Table 2, it is evident that the PINN model exhibits superior performance in approximating the solution of the differential equation when compared to the ANN model. Now, we delve into the outcomes achieved, presenting a comparative analysis of the results derived from PINN and ANN. This comparison is visually represented in Figures 7 and 8.

As illustrated in Figure 7, the deflection curve predicted by the PINN overlaps with the exact deflection curve (obtained from the exact deflection equation, Equation (27)), demonstrating a high degree of accuracy. Conversely, the deflection curve derived from the ANN exhibits a noticeable deviation from the exact solution. In our method, we are giving the data points (i.e., collocation points generated along the length of the beam) as input to the neural network in both models, which tries to map a function between the data points and the deflection over the beam, but the difference arises in the implementation of the loss function. In PINNs, we include scientific or physical principles with the fitting of available data within the loss function framework, thereby ensuring a more holistic model. Conversely, in the ANN, the loss function's formulation is exclusively based on empirical data without integrating physical laws or constraints. The differential equation for deflection (Equation (14)) of the cantilever beam is of the fourth order, which we are approximating through PINNs and ANNs. Additionally, the loss curve for both models, as depicted in Figures 5 and 6, demonstrates good convergence. However, the complex nature of the governing equation, with its high order and non-linearity, poses challenges for the ANN model in accurately approximating the solution. As a result, the model exhibits a deviation from the exact deflection curve.

Additionally, we have predicted the slope, bending moment, and shear force experienced by the beam along its length under triangular loading. The comparative analysis among the curves representing the exact solution, PINN solution, and the solution obtained through ANN is illustrated in Figure 8. As observed in Figure 8, the solution derived from the PINN perfectly matches the exact solution. Conversely, the solution obtained through ANN exhibits significant deviations. The calculation of slope, bending moment, and shear force is achieved through the differentiation of the output provided by the neural network in both models.

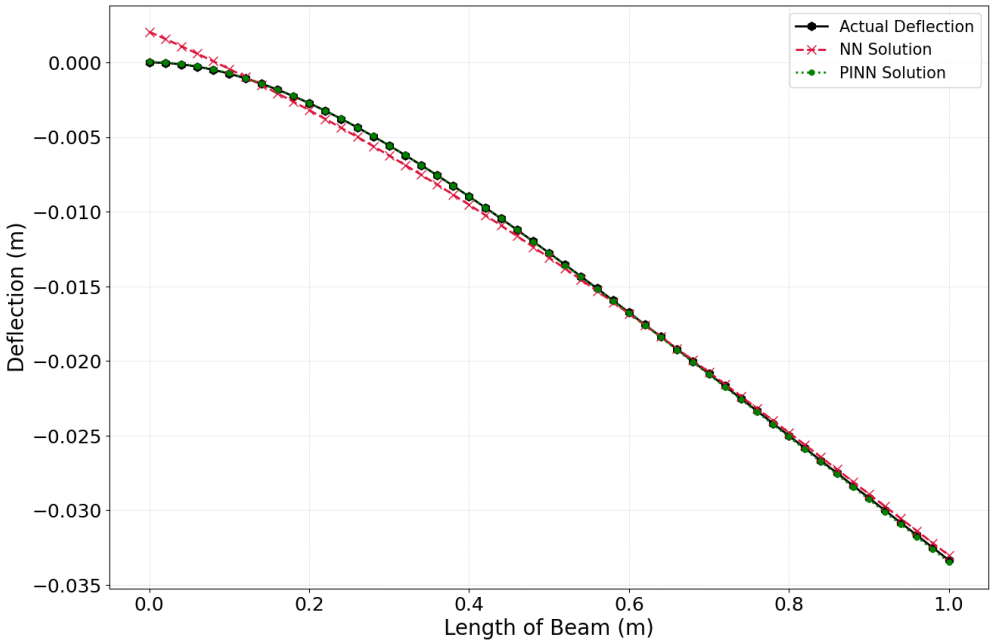
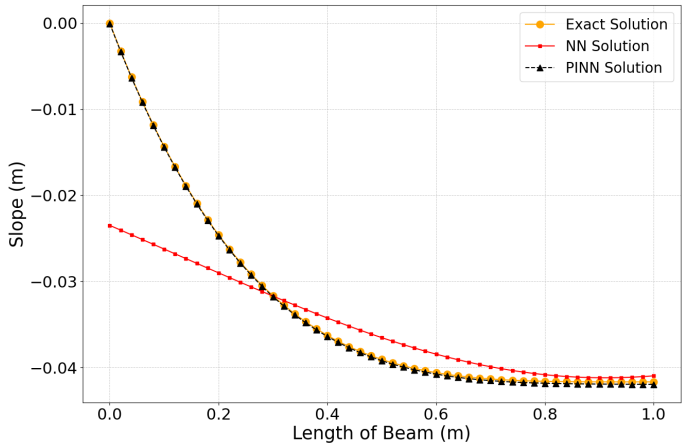
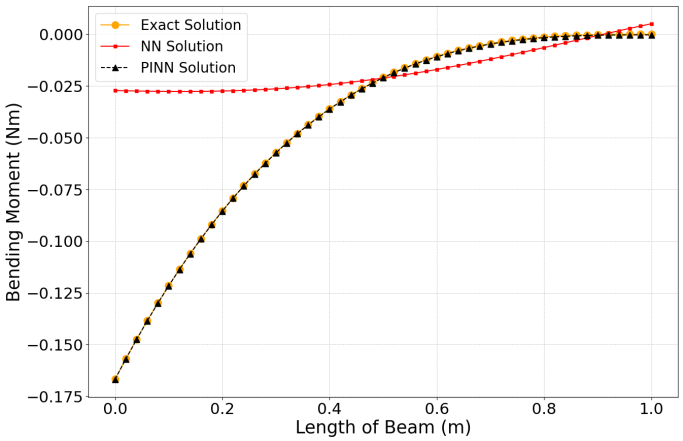


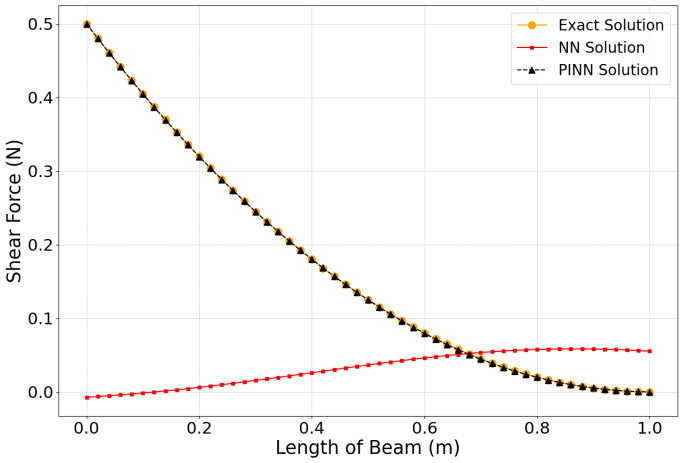
Figure 7. Comparing the solutions from PINN and ANN for predicting deflection.



(**a**)



(**b**)



(**c**)

Figure 8. Comparing the solutions from PINN and ANN for predicting slope (**a**), bending moment (**b**), and shear force (**c**).

Table 2. Comparison of MSE values between the exact and predicted solution for ANN and PINN model.

	ANN	**PINN**
Deflection	3.070×10^{-7}	3.060×10^{-9}
Slope	4.500×10^{-5}	2.935×10^{-8}
Bending Moment	0.002	5.517×10^{-8}
Shear Force	0.049	1.127×10^{-7}

The slope is calculated as the first derivative of the deflection equation (Equation (27)). In Figure 8a, the PINN solution aligns with the exact solution, while the ANN solution does not achieve this level of accuracy. We aim to establish a function that maps the data points to the deflection of the curve using both PINN and ANN models. The output from the neural networks of both models is differentiated for the first time to determine the slope of the cantilever beam. However, the result from the PINN model is significantly better than the ANN models.

The bending moment of the cantilever beam is determined by taking the second derivative of the deflection equation (Equation (27)). The comparison in Figure 8b shows that the PINN solution closely aligns with the exact solution, while the ANN solution does not. Here, the output from the neural network of both models is differentiated two times to obtain the bending moment of the cantilever beam under triangular loading. However, the PINN solution demonstrates superior performance compared to the ANN solution.

By taking the third derivative of the bending equation (Equation (27)), we can determine the shear force in the cantilever beam under triangular loading. As shown in Figure 8c, the PINN solution closely matches the exact solution, while the predicted solution by the ANN does not. Here also, the output from the neural network of both models is differentiated three times, which gives us the shear force. Once again, the predicted solution from the PINN model outperforms the ANN solution.

From the above results in predicting deflection, slope, bending moment, and shear force, it is clear that the PINN approach performs much better as compared to the ANNs because PINNs incorporate physical constraints into their loss function, giving them an advantage over conventional ANNs. Unlike PINNs, ANNs depend exclusively on the dataset provided and thus encounter challenges in accurately predicting the solution.

The predicted solution given by the ANN model, in comparison to the deflection curve (Figure 7), deviated less as compared to the slope curve, bending moment curve, and shear force curve (Figure 8). The deviation in the deflection curve is less because we are directly mapping a function between the data points (as input) and deflection (i.e., exact solution) over the length of the beam (as output) in ANN. However, there are still errors in predicting the deflection curve, and the ANN model fails to accurately predict the deflection due to the complex and non-linear nature of the equation. When differentiating the ANN's output (i.e., predicted deflection) to calculate the slope, bending moment, and shear force, any initial errors in the predicted deflection are amplified. This amplification occurs because differentiation inherently magnifies errors with increasing order of derivative. This means that as the order of the derivative increases, the error also increases, posing a challenge for the accurate prediction of slope (first order), bending moment (second order), and shear force (third order), as illustrated in Table 2. Moreover, the lack of physical information (such as boundary condition and differential equation) in the ANN's loss function adds to the compounded inaccuracies in these derived quantities. However, with PINN, such issues do not occur. The PINN model accurately predicts deflection, slope, bending moment, and shear force.

In traditional methods like the finite element method (FEM), achieving higher accuracy comes at the cost of increased computational expense, as more refined grids are required to improve results [52,68]. The need for finer grids leads to greater computational costs. To address these limitations, researchers have turned to machine learning approaches [25,69]. Physics-informed neural networks (PINNs) offer an innovative approach to solving partial differential equations (PDEs) with reduced computational time compared to FEM. The key advantage of this shift lies in the time invested in training the model, which allows machine learning algorithms to potentially outperform traditional FEMs in engineering problem solving [70]. Thus, PINNs provide a solid and effective framework for solving computational mechanics problems governed by differential equations.

5. Conclusions

In this study, we have demonstrated the application of physics-informed neural networks (PINNs) to computational mechanics problems, particularly with applications in the aerospace sector. We considered the helicopter blade as a cantilever beam subjected to triangular loading. We employ PINNs to approximate or predict the deflection of the cantilever beam. Additionally, we leverage PINNs to estimate the corresponding slope, bending moment, and shear force, providing a comprehensive analysis of the beam's mechanical behavior. We have successfully trained a PINN model and, for comparison, have also trained an ANN model using identical parameters such as epochs of 300, learning rate of 0.001, optimizer as ADAM, and five hidden layers with 50 neurons in each layer. The outcomes derived from the PINN model demonstrate a high degree of accuracy, as the predicted solution aligns precisely with the exact or analytical solution. Conversely, the solution predicted by the ANN model exhibits a noticeable deviation from the exact solution. The results obtained from PINNs achieved very low MSE values as compared to the ANN results as shown in Table 2. This comparative analysis highlights the improved effectiveness of the PINN framework in capturing the fundamental physical principles that govern the differential equation, resulting in more precise and dependable approximations.

It can be concluded that physics-informed neural networks (PINNs) offer an efficient and precise approach for solving computational mechanics challenges, with significant applications in the aerospace sector. In the field of aerospace engineering, the simulation of systems operating under complex conditions through conventional solvers incurs significant computational expenses. As an alternative, physics-informed neural networks (PINNs) offer solutions that are not only more accurate but also markedly more efficient and robust.

For future work, we plan to extend the application of PINNs to solve a broader range of computational mechanics problems within the aerospace sector. This will include tackling more complex geometries and incorporating dynamic loading conditions. Overall, our findings underscore the transformative potential of PINNs in aerospace applications, paving the way for more efficient and accurate simulations that can significantly advance the field.

Author Contributions: Conceptualization, R.M., V.S., D.H. and S.S.; Funding acquisition, S.S. and R.M.; Investigation, V.S. and R.M.; Methodology, R.M., M.S. and S.D.; Project administration, R.M., D.H. and S.S.; Software, V.S.; Supervision, R.M.; Visualization, V.S.; Writing—original draft, V.S., R.M. and S.S.; Writing—review & editing, R.M., D.H., S.S., S.D. and M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Anusandhan National Research Foundation (ANRF) Erstwhile Science and Engineering Research Board, DST, India, under the MATRICS Scheme, file number M.T.R./2022/001029.

Data Availability Statement: The data presented in this study are available on request. The data are not publicly available due to privacy.

Acknowledgments: The authors extend their appreciation to the AgAutomate Pvt. Ltd. for supporting this work through a Consultancy Grant under AgA/RT/CG/2022/0101.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gere, J.; Goodno, B. *Mechanics of Materials, Eighth Edition*; Cengage Learning: Boston, MA, USA, 2012.
2. Curnier, A. *Computational Methods in Solid Mechanics*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; Volume 29.
3. Bykiv, N.; Yasniy, P.; Lapusta, Y.; Iasnii, V. Finite element analysis of reinforced-concrete beam with shape memory alloy under the bending. *Procedia Struct. Integr.* **2022**, *36*, 386–393. [CrossRef]
4. Ma, G.; Jiang, Q.; Zong, X.; Wang, J. Identification of flexural rigidity for Euler–Bernoulli beam by an iterative algorithm based on least squares and finite difference method. *Structures* **2023**, *55*, 138–146. [CrossRef]

5. Chehel Amirani, M.; Khalili, S.; Nemati, N. Free vibration analysis of sandwich beam with FG core using the element free Galerkin method. *Compos. Struct.* **2009**, *90*, 373–379. [CrossRef]
6. Liu, G. *Meshfree Methods: Moving Beyond the Finite Element Method*, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2009.
7. Kononenko, O.; Kononenko, I. Machine Learning and Finite Element Method for Physical Systems Modeling. *arXiv* **2018**, arXiv:1801.07337. [CrossRef]
8. Kag, V.; Gopinath, V. Physics-informed neural network for modeling dynamic linear elasticity. *arXiv* **2024**, arXiv:2312.15175. [CrossRef]
9. Kaymak, S.; Helwan, A.; Uzun, D. Breast cancer image classification using artificial neural networks. *Procedia Comput. Sci.* **2017**, *120*, 126–131. [CrossRef]
10. Sako, K.; Mpinda, B.N.; Rodrigues, P.C. Neural Networks for Financial Time Series Forecasting. *Entropy* **2022**, *24*, 657. [CrossRef]
11. Sinha, D.; Sarangi, P.K.; Sinha, S. Efficacy of Artificial Neural Networks (ANN) as a Tool for Predictive Analytics. In *Analytics Enabled Decision Making*; Sharma, V., Maheshkar, C., Poulose, J., Eds.; Springer: Singapore, 2023; pp. 123–138. [CrossRef]
12. Yue, T.; Wang, Y.; Zhang, L.; Gu, C.; Xue, H.; Wang, W.; Lyu, Q.; Dun, Y. Deep Learning for Genomics: From Early Neural Nets to Modern Large Language Models. *Int. J. Mol. Sci.* **2023**, *24*, 15858. [CrossRef]
13. Otter, D.W.; Medina, J.R.; Kalita, J.K. A Survey of the Usages of Deep Learning for Natural Language Processing. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 604–624. [CrossRef]
14. McCracken, M.F. Artificial Neural Networks in Fluid Dynamics: A Novel Approach to the Navier-Stokes Equations. In Proceedings of the Practice and Experience on Advanced Research Computing, Pittsburgh, PA, USA, 22–26 July 2018; PEARC '18. [CrossRef]
15. Morimoto, M.; Fukami, K.; Zhang, K.; Fukagata, K. Generalization techniques of neural networks for fluid flow estimation. *Neural Comput. Appl.* **2021**, *34*, 3647–3669. [CrossRef]
16. Hsu, Y.C.; Yu, C.H.; Buehler, M.J. Using deep learning to predict fracture patterns in crystalline solids. *Matter* **2020**, *3*, 197–211. [CrossRef]
17. Mianroodi, J.R.; Siboni, N.H.; Raabe, D. Teaching Solid Mechanics to Artificial Intelligence: A fast solver for heterogeneous solids. *arXiv* **2021**, arXiv:2103.09147. [CrossRef]
18. Khatir, A.; Capozucca, R.; Khatir, S.; Magagnini, E. Vibration-based crack prediction on a beam model using hybrid butterfly optimization algorithm with artificial neural network. *Front. Struct. Civ. Eng.* **2022**, *16*, 976–989. [CrossRef]
19. Bolandi, H.; Li, X.; Salem, T.; Boddeti, V.N.; Lajnef, N. Bridging finite element and deep learning: High-resolution stress distribution prediction in structural components. *Front. Struct. Civ. Eng.* **2022**, *16*, 1365–1377. [CrossRef]
20. Teng, S.; Chen, G.; Wang, S.; Zhang, J.; Sun, X. Digital image correlation-based structural state detection through deep learning. *Front. Struct. Civ. Eng.* **2022**, *16*, 45–56. [CrossRef]
21. Le-Nguyen, K.; Minh, Q.C.; Ahmad, A.; Ho, L.S. Development of deep neural network model to predict the compressive strength of FRCM confined columns. *Front. Struct. Civ. Eng.* **2022**, *16*, 1213–1232. [CrossRef]
22. Tran, V.Q.; Giap, V.L.; Vu, D.P.; George, R.C.; Ho, L.S. Application of machine learning technique for predicting and evaluating chloride ingress in concrete. *Front. Struct. Civ. Eng.* **2022**, *16*, 1153–1169. [CrossRef]
23. Diao, Y.; Yang, J.; Zhang, Y.; Zhang, D.; Du, Y. Solving multi-material problems in solid mechanics using physics-informed neural networks based on domain decomposition technology. *Comput. Methods Appl. Mech. Eng.* **2023**, *413*, 116120. [CrossRef]
24. Raissi, M.; Perdikaris, P.; Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
25. Cuomo, S.; di Cola, V.S.; Giampaolo, F.; Rozza, G.; Raissi, M.; Piccialli, F. Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next. *arXiv* **2022**, arXiv:2201.05624. [CrossRef]
26. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
27. Lu, L.; Meng, X.; Mao, Z.; Karniadakis, G.E. DeepXDE: A Deep Learning Library for Solving Differential Equations. *SIAM Rev.* **2021**, *63*, 208–228. [CrossRef]
28. Kadeethum, T.; Jørgensen, T.M.; Nick, H.M. Physics-informed neural networks for solving nonlinear diffusivity and Biot's equations. *PLoS ONE* **2020**, *15*, e0232683. [CrossRef] [PubMed]
29. Meng, X.; Li, Z.; Zhang, D.; Karniadakis, G.E. PPINN: Parareal physics-informed neural network for time-dependent PDEs. *Comput. Methods Appl. Mech. Eng.* **2020**, *370*, 113250. [CrossRef]
30. Sharma, P.; Evans, L.; Tindall, M.; Nithiarasu, P. Stiff-PDEs and physics-informed neural networks. *Arch. Comput. Methods Eng.* **2023**, *30*, 2929–2958. [CrossRef]
31. Muller, A.P.O.; Costa, J.C.; Bom, C.R.; Klatt, M.; Faria, E.L.; de Albuquerque, M.P.; de Albuquerque, M.P. Deep pre-trained FWI: Where supervised learning meets the physics-informed neural networks. *Geophys. J. Int.* **2023**, *235*, 119–134. [CrossRef]
32. Rebai, A.; Boukhris, L.; Toujani, R.; Gueddiche, A.; Banna, F.A.; Souissi, F.; Lasram, A.; Rayana, E.B.; Zaag, H. Unsupervised physics-informed neural network in reaction-diffusion biology models (Ulcerative colitis and Crohn's disease cases) A preliminary study. *arXiv* **2023**, arXiv:2302.07405. [CrossRef]
33. Sahin, T.; von Danwitz, M.; Popp, A. Solving Forward and Inverse Problems of Contact Mechanics using Physics-Informed Neural Networks. *arXiv* **2023**, arXiv:2308.12716. [CrossRef]
34. Eivazi, H.; Tahani, M.; Schlatter, P.; Vinuesa, R. Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations. *Phys. Fluids* **2022**, *34*, 075117. [CrossRef]

35. Hu, B.; McDaniel, D. Applying Physics-Informed Neural Networks to Solve Navier–Stokes Equations for Laminar Flow around a Particle. *Math. Comput. Appl.* **2023**, *28*, 102. Available online: https://www.mdpi.com/2297-8747/28/5/102 (accessed on 4 September 2024). [CrossRef]
36. Jalili, D.; Jang, S.; Jadidi, M.; Giustini, G.; Keshmiri, A.; Mahmoudi, Y. Physics-informed neural networks for heat transfer prediction in two-phase flows. *Int. J. Heat Mass Transf.* **2024**, *221*, 125089. [CrossRef]
37. Mukhmetov, O.; Zhao, Y.; Mashekova, A.; Zarikas, V.; Ng, E.Y.K.; Aidossov, N. Physics-informed neural network for fast prediction of temperature distributions in cancerous breasts as a potential efficient portable AI-based diagnostic tool. *Comput. Methods Programs Biomed.* **2023**, *242*, 107834. [CrossRef] [PubMed]
38. Dhiman, A.; Hu, Y. Physics Informed Neural Network for Option Pricing. *arXiv* **2023**, arXiv:2312.06711. [CrossRef]
39. Haghighat, E.; Raissi, M.; Moure, A.; Gomez, H.; Juanes, R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Comput. Methods Appl. Mech. Eng.* **2021**, *379*, 113741. [CrossRef]
40. Zhi, P.; Wu, Y.; Qi, C.; Zhu, T.; Wu, X.; Wu, H. Surrogate-Based Physics-Informed Neural Networks for Elliptic Partial Differential Equations. *Mathematics* **2023**, *11*, 2723. Available online: https://www.mdpi.com/2227-7390/11/12/2723 (accessed on 4 September 2024). [CrossRef]
41. Rao, C.; Sun, H.; Liu, Y. Physics-informed deep learning for computational elastodynamics without labeled data. *J. Eng. Mech.* **2021**, *147*, 04021043. [CrossRef]
42. Bai, J.; Rabczuk, T.; Gupta, A.; Alzubaidi, L.; Gu, Y. A physics-informed neural network technique based on a modified loss function for computational 2D and 3D solid mechanics. *Comput. Mech.* **2022**, *71*, 543–562. [CrossRef]
43. Bai, J.; Jeong, H.; Batuwatta-Gamage, C.P.; Xiao, S.; Wang, Q.; Rathnayaka, C.M.; Alzubaidi, L.; Liu, G.R.; Gu, Y. An Introduction to Programming Physics-Informed Neural Network-Based Computational Solid Mechanics. *Int. J. Comput. Methods* **2023**, *20*, 2350013. [CrossRef]
44. Abueidda, D.W.; Koric, S.; Guleryuz, E.; Sobh, N.A. Enhanced physics-informed neural networks for hyperelasticity. *Int. J. Numer. Methods Eng.* **2022**, *124*, 1585–1601. [CrossRef]
45. Kapoor, T.; Wang, H.; Núñez, A.; Dollevoet, R. Physics-informed neural networks for solving forward and inverse problems in complex beam systems. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *35*, 5981–5995. [CrossRef]
46. Faroughi, S.; Darvishi, A.; Rezaei, S. On the order of derivation in the training of physics-informed neural networks: Case studies for non-uniform beam structures. *Acta Mech.* **2023**, *234*, 5673–5695. [CrossRef]
47. Liu, Y.; Wang, L.; Ng, B.F. A hybrid model-data-driven framework for inverse load identification of interval structures based on physics-informed neural network and improved Kalman filter algorithm. *Appl. Energy* **2024**, *359*, 122740. Available online: https://www.sciencedirect.com/science/article/pii/S0306261924001235 (accessed on 4 September 2024). [CrossRef]
48. Bazmara, M.; Mianroodi, M.; Silani, M. Application of physics-informed neural networks for nonlinear buckling analysis of beams. *Acta Mech. Sin.* **2023**, *39*, 422438. [CrossRef]
49. Trinh, D.T.N.; Luong, K.A.; Lee, J. An analysis of functionally graded thin-walled beams using physics-informed neural networks. *Eng. Struct.* **2024**, *301*, 117290. Available online: https://www.sciencedirect.com/science/article/pii/S0141029623017054 (accessed on 4 September 2024). [CrossRef]
50. Verma, A.; Mallick, R.; Harursampath, D.; Sahay, P.; Mishra, K.K. Physics-Informed Neural Networks with Application in Computational Structural Mechanics. Available online: https://scml.jp/2024/paper/29/CameraReady/scml2024.pdf (accessed on 4 September 2024).
51. Mouratidou, A.D.; Drosopoulos, G.A.; Stavroulakis, G.E. Ensemble of Physics-informed Neural Networks for Solving Plane Elasticity Problems with Examples. *Acta Mech.* **2024**, *235*, 6703–6722. [CrossRef]
52. Cho, W.; Jo, M.; Lim, H.; Lee, K.; Lee, D.; Hong, S.; Park, N. Parameterized Physics-informed Neural Networks for Parameterized PDEs. *arXiv* **2024**, arXiv:2408.09446. [CrossRef]
53. Anton, D.; Tröger, J.-A.; Wessels, H.; Römer, U.; Henkes, A.; Hartmann, S. Deterministic and statistical calibration of constitutive models from full-field data with parametric physics-informed neural networks. *arXiv* **2024**, arXiv:2405.18311. [CrossRef]
54. Kovachki, N.; Li, Z.; Liu, B.; Azizzadenesheli, K.; Bhattacharya, K.; Stuart, A.; Anandkumar, A. Neural operator: Learning maps between function spaces with applications to PDEs. *J. Mach. Learn. Res.* **2023**, *24*, 1–97.
55. Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; Anandkumar, A. Fourier Neural Operator for Parametric Partial Differential Equations. *arXiv* **2021**, arXiv:2010.08895. [CrossRef]
56. Rashid, M.M.; Pittie, T.; Chakraborty, S.; Krishnan, N.A. Learning the stress-strain fields in digital composites using Fourier neural operator. *iScience* **2022**, *25*, 105452. Available online: https://www.sciencedirect.com/science/article/pii/S2589004222017242 (accessed on 4 September 2024). [CrossRef]
57. Li, Z.; Zheng, H.; Kovachki, N.; Jin, D.; Chen, H.; Liu, B.; Azizzadenesheli, K.; Anandkumar, A. Physics-Informed Neural Operator for Learning Partial Differential Equations. *ACM/IMS J. Data Sci.* **2024**, *1*, 9. [CrossRef]
58. Rosofsky, S.G.; Al Majed, H.; Huerta, E.A. Applications of physics informed neural operators. *Mach. Learn. Sci. Technol.* **2023**, *4*, 025022. [CrossRef]
59. Dell'Aversana, P. Artificial neural networks and deep learning: A simple overview. In *A Global Approach to Data Value Maximization. Integration, Machine Learning and Multimodal Analysis*; Cambridge Scholars Publishing: Newcastle upon Tyne, UK, 2019; Volume 12.
60. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 4 September 2024).

61. Liu, G.R. *Machine Learning with Python*; World Scientific: Singapore, 2022. [CrossRef]
62. Liquet, B.; Moka, S.; Nazarathy, Y. *Mathematical Engineering of Deep Learning*; CRC Press: Boca Raton, FL, USA, 2024.
63. Schäfer, V. Generalization of Physics-Informed Neural Networks for Various Boundary and Initial Conditions. PhD Thesis, Technische Universität Kaiserslautern, Kaiserslautern, Germany, 2022.
64. Baydin, A.G.; Pearlmutter, B.A.; Radul, A.A.; Siskind, J.M. Automatic differentiation in machine learning: A survey. *arXiv* **2018**, arXiv:1502.05767. [CrossRef]
65. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980. [CrossRef]
66. Dubey, S.R.; Singh, S.K.; Chaudhuri, B.B. Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark. *arXiv* **2022**, arXiv:2109.14545. [CrossRef]
67. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv* **2019**, arXiv:1912.01703. [CrossRef]
68. Patidar, K.C. Nonstandard finite difference methods: Recent trends and further developments. *J. Differ. Equ. Appl.* **2016**, *22*, 817–849. [CrossRef]
69. Karniadakis, G.E.; Kevrekidis, I.G.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L. Physics-informed machine learning. *Nat. Rev. Phys.* **2021**, *3*, 422–440. [CrossRef]
70. Santos, L. Deep and Physics-Informed Neural Networks as a Substitute for Finite Element Analysis. In Proceedings of the 2024 9th International Conference on Machine Learning Technologies, Oslo, Norway, 24–26 May 2024; pp. 84–90.