Journal of Science and Technology – Smart Systems and Devices
Volume x, Issue x, Month Year, ppp-ppp

GradNorm Physics-Informed Neural Networks for Linear Elasticity: Adaptive Loss Balancing and Comparative Finite Element Analysis

Phuong Cuc Hoang, Thi Thanh Mai Ta, Nam Nguyen Canh*

Faculty of Mathematics and Informatics, Hanoi University of Science and Technology, Ha Noi, Vietnam
**Corresponding author, email: nam.nguyencanh@hust.edu.vn*

Abstract

This study presents a comparative analysis between the classical Finite Element Method (FEM) and Physics-Informed Neural Networks (PINNs) for linear elasticity. A well-known challenge in PINN formulations stems from imbalances among loss components associated with governing equations and boundary conditions, which often induce training instabilities and ill-conditioned optimization dynamics. To address this issue, we employ GradNorm, a gradient-based adaptive loss-balancing strategy, to dynamically adjust the weighting parameters during training, thereby mitigating optimization stiffness and improving convergence. The proposed PINN approach is systematically evaluated against high-fidelity FEM benchmarks across various geometries and loading conditions. Numerical results demonstrate that, while FEM remains a highly computationally efficient method for linear elastic problems, PINNs equipped with GradNorm-based adaptive weighting constitute a robust, mesh-free alternative with comparable accuracy. These findings underscore the efficacy of adaptive loss-balancing strategies for enhancing the reliability of PINNs in computational mechanics.

Keywords: Finite element method, linear elasticity, forward problem, partial differential equations, physics-informed neural networks.

1. Introduction

Linear elasticity is a cornerstone of continuum mechanics, providing a mathematical framework to model the deformation and stress in solid bodies under external loads. Based on the assumptions of small deformations and a linear stress-strain relationship, the theory simplifies the more complex nonlinear elasticity. This balance of simplicity and accuracy makes it invaluable across engineering and applied sciences, from the structural analysis of buildings and bridges to modeling biological tissues and geological phenomena. By formulating physical laws as partial differential equations, linear elasticity offers a robust and practical method for analyzing complex systems, enabling engineers and researchers to predict performance and optimize designs.

The Finite Element Method (FEM) [1] originated in the 1950s, when aeronautical engineers first employed numerical techniques to address structural mechanics problems. Since then, FEM has developed into a highly successful and widely adopted framework for solving Partial Differential Equations (PDEs). Today, the term "Finite Element Method" denotes not only the mere interpolation technique it is, but also a broad class set of PDEs and approximation techniques [2]. Its enduring success is attributed to a rigorous mathematical foundation, which ensures theoretical guarantees of convergence and accuracy. Experience shows that failure to produce an approximate solution with acceptable accuracy is almost invariably linked to departure from the mathematical foundations. Nevertheless, the method faces several limitations. Real-world problems are often high-dimensional and computationally demanding; as the dimensionality increases, the associated cost can grow exponentially, restricting FEM's applicability to large-scale or high-dimensional systems. Moreover, FEM computations are resource-intensive and prone to various sources of numerical error, arising from the complexity of the governing PDEs, the discretization and interpolation methods employed, and the iterative procedures necessary for achieving convergence.

In recent years, deep learning-based approaches-most notably Physics-Informed Neural Networks (PINNs) [3] have emerged as viable alternatives to traditional numerical solvers for partial differential equations (PDEs). PINNs leverage the universal approximation capability of neural networks by incorporating governing physical laws directly into the loss function, thereby enabling data-efficient learning of PDE solutions. This framework benefits significantly from advances in modern computational hardware, particularly GPUs, which facilitate the efficient approximation of highly nonlinear functions [4]. PINNs have since been successfully applied to a broad range of problems in fluid mechanics [5–7], heat transfer [8, 9], where they have demonstrated high accuracy and stability. More recently, elastic deformation problems have also attracted growing interest, with several studies reporting encouraging results [10–12]. Despite these advantages, PINNs still face several unresolved

p-ISSN 3093-3285
e-ISSN 3093-3315
https://doi.org/10.51316/jst.xxx.ssad.xxxx.xx.x.x
Received: Jan 10, 2026; Revised: Jan 29, 2026;
Accepted: Jan 29, 2026; Online: Feb 24, 2026

challenges. Common issues include convergence to suboptimal local minima, imbalance between data-driven and physics-based loss components, and limited scalability to complex or multi-domain problems [13]. Moreover, theoretical questions concerning convergence rates and generalization capabilities remain largely open [14]. Elasticity, in particular, poses additional difficulties: the vector–tensor nature of elliptic operators, the complexity of mixed boundary conditions, susceptibility to ill-conditioning, and the emergence of sharp stress or displacement gradients often hinder training and reduce robustness [15]. Nevertheless, the mesh-free nature and inherent flexibility of PINNs have rendered them increasingly competitive with conventional numerical methods in specific contexts.

This work presents a systematic comparative study of the FEM and PINNs for linear elasticity problems. However, in contrast to previous studies on PINNs for elasticity, which typically rely on heuristic or empirically tuned schemes to balance the different components of the loss function, the present work employs a principled, gradient-based weighting strategy, namely GradNorm [16], to update the weight parameters λ during training adaptively. Existing literature on PINNs for solid mechanics commonly adopts alternative strategies such as residual-based adaptive weighting [17], self-adaptive loss balancing [18], or NTK-inspired weighting schemes [19]. To the best of our knowledge, the use of GradNorm within the standard PINN framework for elasticity is not yet widely adopted and has received limited attention in the existing literature. Incorporating this mechanism offers a new perspective on improving training stability and alleviating optimization challenges inherent to elasticity problems. The aim is to evaluate their respective performance in terms of accuracy, computational cost, and scalability. Similar to our previous study on elastic deformation analysis in bimetallic materials [20], numerical experiments are conducted across a diverse set of materials, highlighting the distinctive physical characteristics of each material type. They are carried out in both two- and three-dimensional domains to provide quantitative and qualitative assessments, thereby identifying application scenarios where each method exhibits particular strengths or weaknesses.

The remainder of this paper is structured as follows. Section 2 introduces the governing equations of linear elasticity for solid materials. Sections 3 and 4 are devoted to methodological foundations, presenting the FEM and PINNs, respectively. Numerical experiments and benchmark test cases are reported in Section 5 to evaluate the accuracy, efficiency, and scalability of both approaches. Finally, Section 6 discusses the main findings, highlighting the strengths and limitations of the two methods, and provides perspectives for future research.

2. Setting of Problem

Let $\Omega \subset \mathbb{R}^d$ $(d = 2;3)$ be a bounded domain with Lipschitz boundary $\partial\Omega$, representing the reference configuration of a deformed elastic body. This body is subjected to an external force field $\mathbf{f} : \Omega \to \mathbb{R}^d$ and specified boundary conditions. Under the assumption of small deformations, we aim to determine the displacement field $\mathbf{u} : \Omega \to \mathbb{R}^d$ at equilibrium, as governed by the theory of linear elasticity.

For a linear isotropic elastic material, the relationship between the stress tensor $\mathscr{A}(\mathbf{u})$ and the infinitesimal strain tensor $\varepsilon(\mathbf{u})$ is described by Hooke's law:

$$\mathscr{A}(\mathbf{u}) = \lambda \mathrm{tr}(\varepsilon(\mathbf{u}))\mathscr{I} + 2\mu\varepsilon(\mathbf{u}) \tag{1}$$

where λ and μ are the so-called *Lamé coefficients*, and $\mathscr{I}$ is the identity matrix. The strain tensor is defined in terms of the displacement field as:

$$\varepsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T) \tag{2}$$

The equilibrium equation for the stress tensor $\mathscr{A}(\mathbf{u})$ is given by:

$$-\nabla \cdot \mathscr{A}(\mathbf{u}) = \mathbf{f} \quad \text{in} \quad \Omega \tag{3}$$

By substituting the expressions for the stress and strain tensors into the equilibrium equation, we obtain the Navier-Cauchy equation, which is the governing partial differential equation for linear elasticity. The boundary is decomposed as $\partial\Omega = \Gamma_D \cup \Gamma_N$, where Γ_D is the Dirichlet boundary and Γ_N is the Neumann boundary on which a traction $\mathbf{g}_N : \Gamma_N \to \mathbb{R}^d$ is imposed. We get the following *linear elastic system*

$$\begin{cases} -\nabla \cdot \mathscr{A}(\mathbf{u}) = \mathbf{f} & \text{in} \quad \Omega, \\ \mathbf{u} = 0 & \text{on} \quad \Gamma_D, \\ \mathscr{A}(\mathbf{u}) \cdot \mathbf{n} = \mathbf{g}_N & \text{on} \quad \Gamma_N \end{cases} \tag{4}$$

Remark 2.1.

(i) The Lamé coefficients reflect the mechanical properties of the material. In particular, $(\lambda + \frac{2}{3}\mu)$ governs compressibility: a very large value corresponds to almost incompressible materials.

(ii) For convenience, the properties of the material can alternatively be expressed in terms of the Young modulus E and the Poisson ratio ν. These quantities are related to the Lamé coefficients by

$$E = \mu\frac{3\lambda + 2\mu}{\lambda + \mu} \quad \text{and} \quad \nu = \frac{1}{2}\frac{\lambda}{\lambda + \mu}$$

Consequently, ν satisfies $-1 \leq \nu < \frac{1}{2}$. With assumption $\lambda \geq 0$, we obtain $\nu \geq 0$. Almost incompressible materials correspond to ν very close to $\frac{1}{2}$.

3. Finite Element Method

3.1. Weak Formulation

The FEM approximates the solution of partial differential equations by first reformulating them into a variational (weak) form. This approach relaxes the continuity requirements on the solution, allowing for approximation within a function space where derivatives are not necessarily continuous.

To derive the weak formulation of the equilibrium equation, we multiply the equation by a test function $\mathbf{v} \in (H^1_{\Gamma_D}(\Omega))^d$ and integrate over the domain Ω. Applying the Green's formula to the left-hand side yields:

$$-\int_\Omega (\nabla \cdot \mathscr{A}(\mathbf{u})) \cdot \mathbf{v} \mathrm{d}x = \int_\Omega \mathscr{A}(\mathbf{u}) \cdot \varepsilon(\mathbf{v}) \mathrm{d}x - \int_{\partial\Omega} (\mathscr{A}(\mathbf{u}) \cdot \mathbf{n}) \mathbf{v} \mathrm{d}s \tag{5}$$

where we have used the identity $\mathscr{A}(\mathbf{u}) \cdot \nabla \mathbf{v} = \mathscr{A}(\mathbf{u}) \cdot \varepsilon(\mathbf{v})$ due to the symmetry of the stress tensor $\mathscr{A}(\mathbf{u})$.

This leads to the following weak formulation: Seek the displacement field $\mathbf{u} \in (H^1_{\Gamma_D}(\Omega))^d$ such that:

$$a(\mathbf{u}, \mathbf{v}) = L(\mathbf{v}) \quad \forall \mathbf{v} \in (H^1_{\Gamma_D}(\Omega))^d \tag{6}$$

where the bilinear form $a(\mathbf{u}, \mathbf{v})$ and the linear form $L(\mathbf{v})$ are defined as:

$$a(\mathbf{u}, \mathbf{v}) = \int_\Omega (\lambda \nabla \cdot \mathbf{u} \nabla \cdot \mathbf{v} + 2\mu \varepsilon(\mathbf{u}) \cdot \varepsilon(\mathbf{v})) \mathrm{d}x \tag{7}$$

$$L(\mathbf{v}) = \int_\Omega \mathbf{f} \cdot \mathbf{v} \mathrm{d}x + \int_{\Gamma_N} \mathbf{g}_N \cdot \mathbf{v} \mathrm{d}s \tag{8}$$

Here, the surface integral terms vanish on the Dirichlet boundary Γ_D due to the choice of the test function space, and the traction boundary condition $\mathscr{A}(\mathbf{u}) \cdot \mathbf{n} = \mathbf{g}_N$ is applied on the Neumann boundary Γ_N.

In establishing the well-posedness of the weak formulation, we first recall that variational problems may be treated within the general analytical framework furnished by the Banach-Nečas-Babuška (BNB) theorem [2, Theorem 2.6], which provides a robust criterion applicable even in nonsymmetric or noncoercive settings. In the specific context of linear elasticity, however, the associated bilinear form $a(\mathbf{u}, \mathbf{v})$ enjoys both continuity and coercivity on the underlying Hilbert space. These structural properties position the problem (6) squarely within the scope of the classical Lax–Milgram lemma [2, Lemma 2.2], from which the existence and uniqueness of the weak solution follow directly.

3.2. Galerkin Approximation

The fundamental idea of Galerkin methods is to replace the Hilbert space V by a *finite-dimensional subspace* $V_h \subset V$ associated with a triangulation $\mathscr{T}_h$ of the domain Ω, where the index h denotes the mesh size (see Fig. 1).

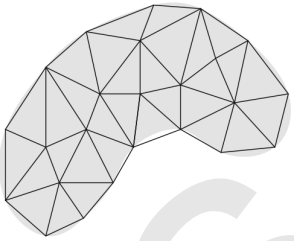


Fig. 1. Triangulation of the domain Ω

In its most general form, the Galerkin method approximates the solution $\mathbf{u}$ of the problem (6) by solving the following approximate problem:

$$\begin{cases} \text{Seek } \mathbf{u}_h \in V_h \text{ such that} \\ a(\mathbf{u}_h, \mathbf{v}_h) = L(\mathbf{v}_h) \quad \forall \mathbf{v}_h \in V_h \end{cases} \tag{9}$$

Let $\{\varphi_1, \varphi_2, \ldots, \varphi_N\}$ be a basis of V_h. We can express the approximate solution $\mathbf{u}_h$ as a linear combination of the basis functions:

$$\mathbf{u}_h = \sum_{i=1}^{N} U_i \varphi_i \tag{10}$$

where $\mathbf{U} = (U_i)_{1 \le i \le N} \in \mathbb{R}^N$ denotes the coordinate vector of $\mathbf{u}_h$. By substituting this expansion into the discrete variational problem and choosing the test function $\mathbf{v}_h$ to be each basis function φ_j in turn, we obtain a system of linear algebraic equations. This system is written in matrix form as:

$$\mathbf{AU} = \mathbf{F} \tag{11}$$

where the stiffness matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ and the load vector $\mathbf{F} \in \mathbb{R}^N$ are given componentwise by $A_{ij} = a(\varphi_i, \varphi_j)$ and $F_j = L(\varphi_j)$, for all $1 \le i, j \le N$. Solving this linear system for $\mathbf{U}$ yields the Galerkin approximation $\mathbf{u}_h$ to the weak solution of the linear elasticity problem (6).

Under standard approximation properties of the finite element space V_h, Céa's lemma [2, Lemma 2.28] guarantees the quasi-optimality of the Galerkin approximation.

4. Physics-Informed Neural Networks

4.1. Introduction to PINNs

Physics-Informed Neural Networks (PINNs), first introduced by Raissi *et al.* (2019) [21], represent a pioneering approach that integrates the physical laws articulated through Partial Differential Equations (PDEs) within deep learning frameworks. By embedding

the governing equations directly into the training process, PINNs enable the resolution of PDE-based problems in scenarios where available data is limited. This methodology not only enhances the predictive capabilities of neural networks but also ensures compliance with underlying physical principles.

Let $\mathbf{NN}(\mathbf{x};W,b) : \mathbb{R}^{d_\mathbf{x}} \to \mathbb{R}^{d_\mathbf{y}}$ denote an L-layer feed-forward neural network, where the input is a vector $\mathbf{x} \in \mathbb{R}^{d_\mathbf{x}}$, the corresponding output is $\mathbf{y} \in \mathbb{R}^{d_\mathbf{y}}$, and W, b the network parameters. The forward pass through the network is defined recursively as:

$$\mathbf{h}^{(l)} = \sigma\Big(W^{(l)}\mathbf{h}^{(l-1)} + b^{(l)}\Big), \quad l = 1,\dots,L \tag{12}$$

with $\mathbf{h}^{(0)} = \mathbf{x}$ and $\mathbf{h}^{(L)} = \mathbf{y}$ representing the input and output of the model, respectively, $W^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}, b^{(l)} \in \mathbb{R}^{n_l}$ are parameters of each layer l, known as weights and biases, respectively. The nonlinear activation functions $\sigma(\cdot)$ enable the network to approximate highly nonlinear mappings (e.g., ReLU or tanh). The final network output approximating the PDE solution is:

$$\mathbf{u}(\mathbf{x}) \approx \hat{\mathbf{u}}(\mathbf{x};\theta) := \mathbf{NN}(\mathbf{x};W,b), \tag{13}$$

$$\mathbf{NN}(\mathbf{x};W,b) = W^{(L+1)}\mathbf{h}^{(L)} + b^{(L+1)} \tag{14}$$

where the full set of learnable parameters is:

$$\theta = \Big\{W^{(l)}, b^{(l)}\Big\}_{l=1}^{L+1} \tag{15}$$

Since the network inputs correspond to Cartesian coordinates, derivatives of the network output with respect to spatial variables can be computed directly via automatic differentiation. In contrast to finite-difference schemes, which are prone to accumulating numerical errors, particularly in deep architectures, modern computational frameworks (e.g., Theano, TensorFlow, MXNet) employ graph-based automatic differentiation to evaluate derivatives with machine-level precision. This functionality enables the consistent application of differential operators $\mathscr{P}$ to the neural-network approximation $\hat{\mathbf{u}}$, thereby facilitating the incorporation of PDE residuals into the learning objective.

For notational convenience, for any quantity g we define the discrete mean-squared residual:

$$|g| := \frac{1}{N}\sum_{i=1}^{N} g(x_i)^2 \tag{16}$$

where $\{x_i\}$ denotes the set of collocation points sampled from Ω (or from its boundary or initial surface, whenever appropriate).

With this notation in place, the total loss function takes the form:

$$\mathscr{L} = \lambda_r \mathscr{L}_{\text{PDE}} + \lambda_b \mathscr{L}_{\text{BC}} + \lambda_i \mathscr{L}_{\text{IC}} \tag{17}$$

where the three components respectively measure the discrepancy with the governing PDE, with the prescribed boundary data, and with the imposed initial condition. Their explicit expressions are:

$$\mathscr{L}_{\text{PDE}} = \big|\, \mathscr{P}(\hat{\mathbf{u}}) - 0^* \,\big| = \frac{1}{N_r}\sum_{i=1}^{N_r} \big\| \mathscr{P}(\hat{\mathbf{u}}(x_i;\theta)) - 0^* \big\|_2^2 \tag{18}$$

$$\mathscr{L}_{\text{BC}} = \big|\, \hat{\mathbf{u}} - \mathbf{u}^* \,\big|_{\partial\Omega} = \frac{1}{N_b}\sum_{i=1}^{N_b} \big\| \hat{\mathbf{u}}(x_i;\theta) - \mathbf{u}^*(x_i) \big\|_2^2 \tag{19}$$

$$\mathscr{L}_{\text{IC}} = \big|\, \hat{\mathbf{u}}(\cdot,0;\theta) - \mathbf{u}_0^* \,\big| = \frac{1}{N_i}\sum_{i=1}^{N_i} \big\| \hat{\mathbf{u}}(x_i,0;\theta) - \mathbf{u}_0^*(x_i) \big\|_2^2 \tag{20}$$

where, $\partial\Omega$ denotes the boundary of the computational domain, and 0^* is the target value of the PDE residual (typically zero). The training consists of minimizing $\mathscr{L}$ with respect to θ using gradient-based optimization.

4.2. Application to Linear Elasticity

For linear elasticity problem (4), the generic loss (17) is specialized into:

$$\mathscr{L} = \lambda_r \mathscr{L}_{\text{PDE}} + \lambda_D \mathscr{L}_D + \lambda_N \mathscr{L}_N \tag{21}$$

where λ_r, λ_D and λ_N are weights that balance the loss contributions. With:

$$\mathscr{L}_{\text{PDE}} = \frac{1}{N_r}\sum_{i=1}^{N_r} \Big\| \nabla\cdot\mathscr{A}(\hat{\mathbf{u}}(x_i;\theta)) + \mathbf{f}(x_i) \Big\|_2^2 \tag{22}$$

$$\mathscr{L}_{\text{D}} = \frac{1}{N_b}\sum_{j=1}^{N_b} \Big\| \hat{\mathbf{u}}(x_j;\theta) - \mathbf{u}_D(x_j) \Big\|_2^2 \tag{23}$$

$$\mathscr{L}_{\text{N}} = \frac{1}{N_b}\sum_{j=1}^{N_b} \Big\| \mathscr{A}(\hat{\mathbf{u}}(x_j;\theta))\cdot\mathbf{n}_j - \mathbf{g}_N(x_j) \Big\|_2^2 \tag{24}$$

where N_r and N_b are the number of collocation points inside the domain and on the boundary, respectively; $\mathbf{u}_D$ and $\mathbf{g}_N$ denote Dirichlet and Neumann boundary data; and $\mathbf{n}_j$ is the outward unit normal at boundary point x_j.

Algorithm 1 Training PINN for Linear Elasticity

Initialize network parameters θ
while not converged **do**
  Sample interior points $x_r \in \Omega$
  Sample boundary points $x_D \in \Gamma_D$, $x_N \in \Gamma_N$
  Compute predicted displacement $\hat{\mathbf{u}}(x;\theta)$
  Compute loss:

$$\mathscr{L} = \lambda_r \mathscr{L}_{\text{PDE}} + \lambda_D \mathscr{L}_D + \lambda_N \mathscr{L}_N$$

  Update $\theta \leftarrow \theta - \eta\nabla_\theta \mathscr{L}$
end while

This construction allows PINN to solve forward elasticity problems without labeled data, relying solely on the governing physics, and thus providing good generalization to unseen domains. Fig. 2 illustrates the PINN architecture for linear elasticity.

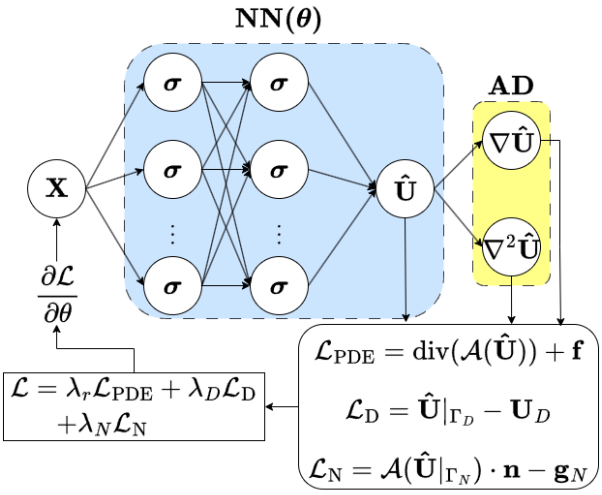


Fig. 2. PINN for linear elasticity problem

5. Experimental Results

This section presents a series of numerical experiments aimed at assessing the accuracy and computational efficiency of the two numerical schemes under consideration, namely the Finite Element Method (FEM) and Physics-Informed Neural Networks (PINNs). These examples are examined in both two- and three-dimensional settings. The PINN architectures employed in the corresponding experiments are listed in Table 1, while all FEM simulations are performed using the FreeFEM++ [22].

Table 1. PINN hyperparameters

Hidden layers	Hidden size	Learning rate	Optimizer	Activation function
6	30	1×10^{-3}	Adam + L-BFGS	Tanh

According to (21), the PINN loss function consists of two components: the PDE residual and the boundary losses. At the beginning of training, the weight parameters λ_r, λ_D, and λ_N are all initialized to 1. During the Adam optimization phase, these weights are adaptively adjusted using the gradient–norm (GradNorm) strategy [16] in order to maintain an appropriate balance between the PDE residual and boundary losses. In the subsequent L–BFGS refinement phase, the weights are kept fixed, and the optimizer primarily serves to stabilize the training process and improve the conditioning of the optimization landscape [15]. For the L–BFGS method, we employ a learning rate of 1.0, a memory size of 100, and a strong Wolfe line search.

Finally, to quantitatively compare the FEM and PINN solutions obtained in these experiments, we evaluate the following error metric

$$\mathscr{E} := \frac{1}{N}\sum_{i=1}^{N} \left\| \mathbf{u}_{\text{PINN}}^{(i)} - \mathbf{u}_{\text{FEM}}^{(i)} \right\|_2^2 \tag{25}$$

where N is the total number of test points.

5.1. 2D Space-Time Examples

This example examines the elastic deformation of a homogeneous, isotropic metal bar subjected to a prescribed body force field. In particular, the bar, fabricated from one of the materials listed in Table 2, is analyzed under a uniform body force $\vec{\mathbf{f}} = (0,-1)^\top$. The geometric configuration together with the applied boundary conditions is illustrated in Fig. 3.

Table 2. Elastic material properties

Materials	E (MPa)	ν
Steel	2.01 E+5	0.291
Copper	1.17 E+5	0.33
Aluminium	0.69 E+5	0.33

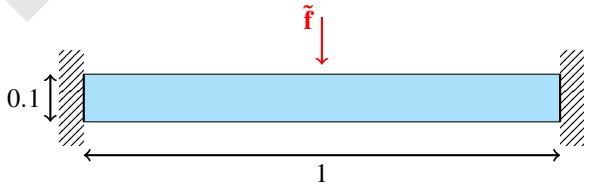
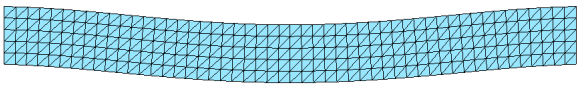


Fig. 3. Initial two-dimensional geometry of the bar

On the boundaries $x = 0$ and $x = 1$, where the metal bar is clamped to a rigid support, we impose homogeneous Dirichlet boundary data for the displacement field. In the PINN training, the interior of the computational domain Ω is sampled on a 200×50 training grid, while the boundary is further refined using a refinement factor `densBnd`=10 in order to improve the resolution near $\partial\Omega$ and mitigate potential boundary underfitting. To maintain consistency between the two numerical schemes, the FEM discretization is constructed so that the number of interior and boundary nodes is comparable to the number of collocation points used in the PINN. This alignment ensures that both approaches operate on similar spatial resolutions across the computational domain.

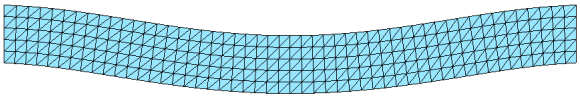
Next, the deformation of the metal bar simulated by FEM and PINN is presented.



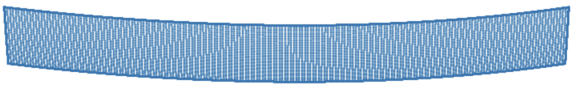
(a) Deformation of the steel bar obtained by FEM

(b) Deformation of the steel bar obtained by PINN

Fig. 4. Simulation results of the steel bar deformation

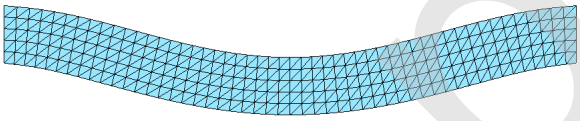


(a) Deformation of the copper bar obtained by FEM

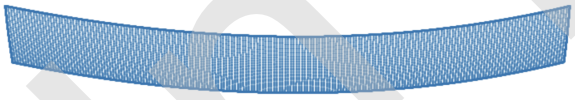


(b) Deformation of the copper bar obtained by PINN

Fig. 5. Simulation results of the copper bar deformation



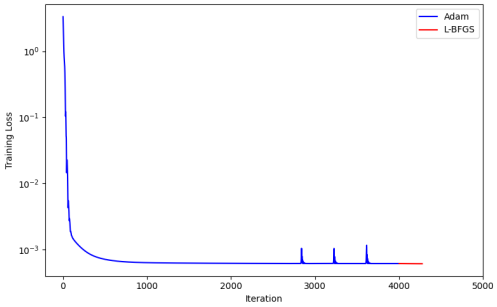
(a) Deformation of the aluminium bar obtained by FEM



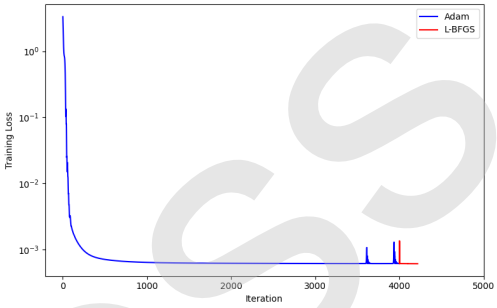
(b) Deformation of the aluminium bar obtained by PINN

Fig. 6. Simulation results of the aluminium bar deformation

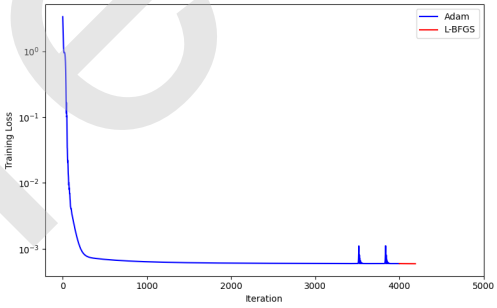
The deformation results of the metal bar presented in Fig. 4, Fig. 5, and Fig. 6 provide clear evidence of the governing role of Young's modulus E in determining structural response. As anticipated, materials with higher Young's modulus exhibit greater stiffness and therefore undergo less deformation. Consequently, the deformation magnitude follows the order: aluminium, copper, and steel. These results are in complete agreement with the intrinsic mechanical properties of the materials, which supports the validity of the simulation outcomes. The convergence process of the loss function shown in Fig. 7 exhibits a smooth and stable convergence behavior, with no noticeable



(a) 2D-The steel bar



(b) 2D-The copper bar



(c) 2D-The aluminium bar

Fig. 7. Convergence of the PINN training loss in the 2D example

oscillations or stagnation, thereby confirming the numerical stability of the training process. This behavior provides further evidence of the stabilizing role of the L-BFGS algorithm, underscoring its effectiveness in enhancing the robustness, convergence reliability, and numerical stability of the neural network training process.

Moreover, the deformation results obtained for each material using both the FEM and the PINN demonstrate a commendable degree of proportional consistency, with the resulting deformation amplitudes being nearly identical. The quantitative error values shown in Table 3 further confirm that the discrepancies between the two schemes are minor, indicating that the PINNs framework can reproduce the overall deformation behavior with reasonable accuracy. However, a closer inspection shows that FEM yields more accurate and spatially consistent displacement fields than PINN. In particular, the PINN solution exhibits noticeable deviations near Dirichlet

boundaries, where the prescribed displacements are only enforced softly through the loss function rather than exactly, as in FEM. This limitation, together with the strong sensitivity of PINNs to hyperparameters-such as network depth, number of hidden layers, and optimization strategy-explains the inferior local accuracy of PINN despite its ability to capture the overall deformation pattern.

Table 3. Training summary for 2D examples

Material	Epochs	Δt	Total Loss	$\mathscr{E}$
Steel	>4000	0.115	6.091×10^{-4}	3.452×10^{-6}
Copper	>4000	0.117	6.099×10^{-4}	7.255×10^{-6}
Aluminium	>4000	0.116	5.932×10^{-4}	2.198×10^{-5}

5.2. 3D Space-Time Examples

For a more realistic illustration, this example presents a three-dimensional simulation of metal bar deformation. Building upon the two-dimensional case, the metal bar model is examined in a three-dimensional setting, where the deformation behavior is analyzed using both the FEM and PINN. In this scenario, a traction $\vec{\mathbf{g}}_N = (0,0,-1)^\top$ is applied on the face $x = 1$, whereas the opposite face at $x = 0$ is rigidly fixed, corresponding to a homogeneous Dirichlet condition. The initial configuration of the metal bar is illustrated in Fig. 8.

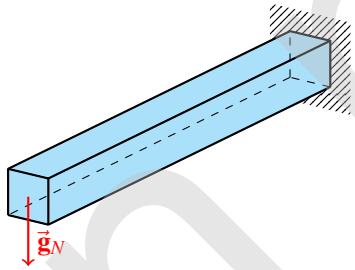
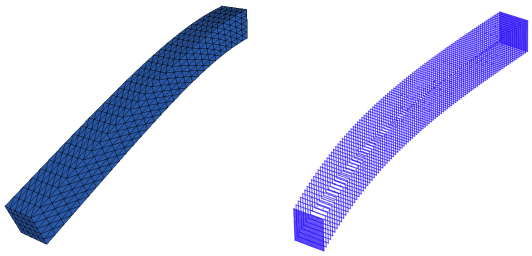


Fig. 8. Initial three-dimensional geometry of the bar

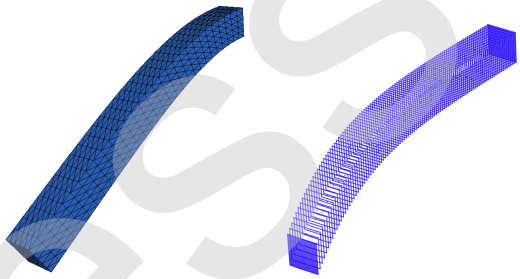
A structured $12\times6\times6$ grid is used to sample interior points for PINN training, while the boundary is refined using a density factor `densBnd`=10 to ensure adequate sampling along $\partial\Omega$. To ensure a consistent comparison between the two numerical schemes, the FEM mesh is generated with a comparable number of interior and boundary nodes to those used in the PINN collocation set, thereby aligning the effective spatial resolution of both discretizations. The corresponding deformation results of the metal bar for each considered material are shown in Fig. 9, Fig. 10, and Fig. 11.

The simulation outcomes, along with the error values reported in Table 4, align well with the trends observed in the preceding two-dimensional experiments. This agreement confirms the ability of the PINN model to reliably reproduce the essential features of linear



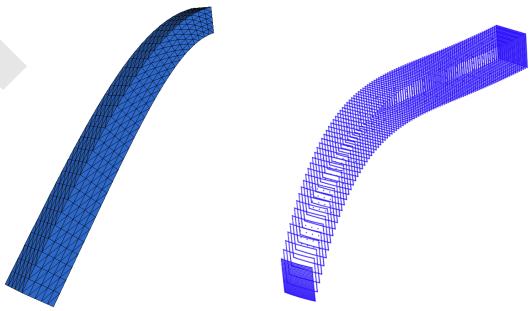
(a) FEM (b) PINN

Fig. 9. Three-dimensional simulation results of the steel bar deformation



(a) FEM (b) PINN

Fig. 10. Three-dimensional simulation results of the copper bar deformation



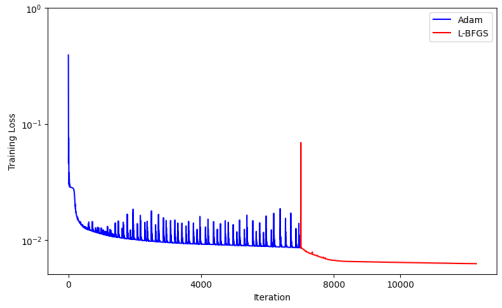
(a) FEM (b) PINN

Fig. 11. Three-dimensional simulation results of the aluminium bar deformation

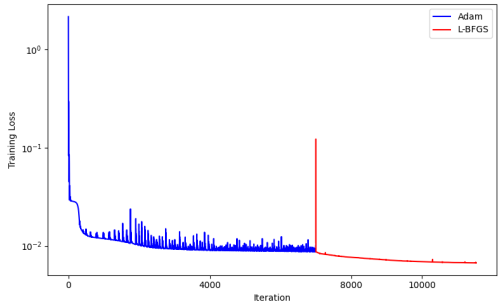
Table 4. Training summary for 3D examples

Material	Epochs	Δt	Total Loss	$\mathscr{E}$
Steel	>10000	0.467	6.714×10^{-3}	1.146×10^{-4}
Copper	>9000	0.468	6.483×10^{-3}	3.435×10^{-4}
Aluminium	>12000	0.453	6.277×10^{-3}	1.491×10^{-3}

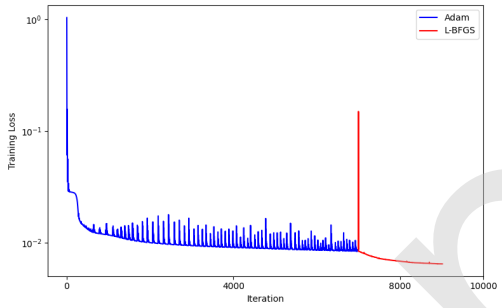
elastic deformation behavior across different material types. As illustrated in Fig. 12, the L-BFGS optimizer contributes significantly to stabilizing the convergence behavior. Moreover, the consistency between the 2D and 3D results highlights the robustness of the PINN formulation when extended to higher-



(a) 3D-The aluminium bar



(b) 3D-The steel bar



(c) 3D-The copper bar

Fig. 12. Convergence of the PINN training loss in the 3D example

dimensional settings, demonstrating its effectiveness in delivering accurate displacement predictions even in more geometrically complex configurations.

6. Conclusion

In this study, we applied both the Finite Element Method (FEM) and Physics-Informed Neural Networks (PINNs) to solve linear elastic deformation problems. Our numerical results exhibit close quantitative agreement with the benchmark solutions provided in the official FreeFEM++ documentation, thereby validating the correctness of our FEM implementation. As expected from a classical and extensively validated numerical framework, FEM consistently demonstrated high accuracy. PINNs, while showing promising performance, exhibit sensitivity to hyperparameters, network architecture, and optimization strategies, particularly in higher-dimensional settings.

One of the primary strengths of PINNs lies in their ability to assimilate multiple sources, such as sparse measurements, heterogeneous boundary data, and coupled multi-physics constraints, within a unified loss-function framework. This flexibility stands in contrast to classical FEM, which does not naturally accommodate such multifaceted data integration. However, the basic PINN formulation still struggles to enforce boundary conditions in a strong sense, a limitation that has motivated the development of more advanced variants, including XPINN [23], gPINN [24], and various domain-decomposition and gradient-augmented approaches.

A further practical challenge is the computational cost: the training of PINN models remains substantially more time-consuming than running FEM solvers, especially for large-scale or three-dimensional problems governed by stiff PDEs. Reducing this training burden is essential for enabling PINNs to become a competitive alternative in industrial and scientific applications. As part of our future research agenda, we plan to apply these advanced PINN variants to more complex problems, including elastic deformation in bimetallic materials [20] and shape optimization [25, 26], where traditional methods have already proven successful.

References

[1] R. L. Taylor and O. C. Zienkiewicz, The Finite Element Method, Butterworth-Heinemann Oxford, UK, 2013.

[2] A. Ern and J.-L. Guermond, Theory and Practice of Finite Elements, Springer New York, 2004.

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

[4] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, vol. 1, MIT Press Cambridge, 2016.

[5] L. Sun, H. Gao, S. Pan, and J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Computer Methods in Applied Mechanics and Engineering, vol. 361, Art. no. 112732, Apr. 2020. https://doi.org/10.1016/j.cma.2019.112732

[6] Z. Xiang, W. Peng, X. Liu, and W. Yao, Self-adaptive loss balanced physics-informed neural networks, Neurocomputing, vol. 496, pp. 11–34, July 2022. https://doi.org/10.1016/j.neucom.2022.05.015

[7] N.-D. Pham, N. Nguyen Canh, and T. T. M. Ta, The physics-informed neural networks approach to parameter identification problems in unsteady Navier-Stokes equations, pp. 164–175, Oct. 2025. https://doi.org/10.1007/978-3-032-08384-5_14

[8] Z. Zhao, Y. Wang, W. Zhang, Z. Ba, and L. Sun, Physics-informed neural networks in heat transfer-dominated multiphysics systems: A comprehensive review, Engineering Applications of Artificial Intelligence, vol. 157, Art. no. 111098, Oct. 2025.
https://doi.org/10.1016/j.engappai.2025.111098

[9] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, Physics-informed neural networks for heat transfer problems, Journal of Heat Transfer, vol. 143, no. 6, Apr. 2021.
https://doi.org/10.1115/1.4050542

[10] E. Fuzaro de Almeida, S. da Silva, and A. Barbosa da Cunha Junior, Physics-informed neural networks for solving elasticity problems, in Proceedings of the 27th International Congress of Mechanical Engineering, 2025.

[11] D. W. Abueidda, S. Koric, E. Guleryuz, and N. A. Sobh, Enhanced physics-informed neural networks for hyperelasticity, International Journal for Numerical Methods in Engineering, vol. 124, no. 7, pp. 1585–1601, Nov. 2022.
https://doi.org/10.1002/nme.7176

[12] H. Moon, D. Park, H. Cho, H.-K. Noh, J. H. Lim, and S. Ryu, Physics-informed neural network-based discovery of hyperelastic constitutive models from extremely scarce data, Computer Methods in Applied Mechanics and Engineering, vol. 446, Art. no. 118258, Nov. 2025.
https://doi.org/10.1016/j.cma.2025.118258

[13] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, Scientific machine learning through physics–informed neural networks: Where we are and what's next, Journal of Scientific Computing, vol. 92, no. 3, July 2022.
https://doi.org/10.1007/s10915-022-01939-z

[14] N. Doumèche, G. Biau, and C. Boyer, On the convergence of PINNs, Bernoulli, vol. 31, no. 3, Aug. 2025.
https://doi.org/10.3150/24-bej1799

[15] P. Rathore, W. Lei, Z. Frangella, L. Lu, and M. Udell, Challenges in training PINNs: A loss landscape perspective, in Proceedings of the 41st International Conference on Machine Learning, July 2024, pp. 42159–42191.

[16] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks, in Proceedings of the 35th International Conference on Machine Learning (ICML), 2018, pp. 794–803.

[17] S. Wang, Y. Teng, and P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM Journal on Scientific Computing, vol. 43, no. 5, pp. A3055–A3081, Jan. 2021.
https://doi.org/10.1137/20m1318043

[18] L. D. McClenny and U. M. Braga-Neto, Self-adaptive physics-informed neural networks, Journal of Computational Physics, vol. 474, Art. no. 111722, Feb. 2023.
https://doi.org/10.1016/j.jcp.2022.111722

[19] S. Wang, X. Yu, and P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, Journal of Computational Physics, vol. 449, Art. no. 110768, Jan. 2022.
https://doi.org/10.1016/j.jcp.2021.110768

[20] T. T. M. Ta and P. C. Hoang, The elastic deformation process of bimetallic objects, International Journal of Mathematics for Industry, vol. 14, no. 01, Dec. 2022.
https://doi.org/10.1142/s266133522250006x

[21] M. Raissi, P. Perdikaris, and G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics, vol. 378, pp. 686–707, Feb. 2019.
https://doi.org/10.1016/j.jcp.2018.10.045

[22] F. Hecht, New development in FreeFEM++, J. Numer. Math. vol. 20, no. 3-4, 2012.
https://doi.org/10.1515/jnum-2012-0013

[23] A. D. J. Ameya D. Jagtap and G. E. K. George Em 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, vol. 28, no. 5, pp. 2002–2041, Jan. 2020.
https://doi.org/10.4208/cicp.oa-2020-0164

[24] J. Yu, L. Lu, X. Meng, and G. E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Computer Methods in Applied Mechanics and Engineering, vol. 393, Art. no. 114823, Apr. 2022.
https://doi.org/10.1016/j.cma.2022.114823

[25] T. T. M. Ta, V. C. Le, and H. T. Pham, Shape optimization for Stokes flows using sensitivity analysis and finite element method, Applied Numerical Mathematics, vol. 126, pp. 160–179, Apr. 2018.
https://doi.org/10.1016/j.apnum.2017.12.009

[26] V. C. Le and T. T. M. Ta, Constrained shape optimization problem in elastic mechanics, Computational and Applied Mathematics, vol. 40, no. 7, Sept. 2021.
https://doi.org/10.1007/s40314-021-01632-1