Effective Strategies for Parameter Tuning in PINNs: From Lower to Higher Dimensional Solid Mechanics

Flávio Valberto Barrionuevo Rodrigues[1], Paulo de Mattos Pimenta[1]

[1]*Department. of Structural and Geotechnical Engineering, University of São Paulo*
Av. Prof. Almeida Prado, 83, 05508-070, São Paulo, São Paulo, Brazil
flavio.brodrigues@usp.br, ppimenta@usp.com

Abstract. In recent years, Physics-Informed Neural Networks (PINNs) have introduced a novel approach to solving partial differential equations (PDEs) using deep learning techniques. Despite the promising results and rapid advancements in the field, there is a lack of comprehensive studies on modeling choices within the deep learning framework, particularly in solid mechanics problems. In this study, we aim to explore the influence of the number and arrangement of neurons, activation functions, and weight initialization on the accuracy of results. We focus on elasticity problems in 1D, 2D, and 3D dimensions to establish a foundation for exploring further complex scenarios. Our findings indicate that studying hyperparameters in lower dimensions is an effective strategy for optimizing performance in higher-dimensional ones.

Keywords: PINNs, solid mechanics, elasticity.

1 Introduction

Physics-Informed Neural Networks (PINNs), introduced by Raissi et al. [1], integrate physical equations into neural networks, enhancing interpretability and prediction by embedding residuals of these equations in the loss function. This allows for simultaneous updates of network weights and physical parameters, improving model performance. Unlike traditional numerical methods, PINNs avoid discretization errors due to their meshless nature, making them suitable for complex geometries Abueidda et al. [2]. They are also resilient to imperfect datasets, allowing applications in areas like geometric nonlinearity - Fuhg and Bouklas [3], hyperelasticity - Nguyen et al. [4], and fracture problems - Goswami et al. [5].

However, PINNs face challenges, including sensitivity to weight initialization, limited theoretical guidance, and difficulty in integrating heterogeneous data, leading to issues with training stability and computational efficiency. While advancements have been made by Wang and Zhong [6] and Psaros et al. [7], no unified framework for designing PINNs or selecting hyperparameters exists, limiting their broader applicability and requiring tailored approaches for different problems. Continued research is needed to improve their efficiency and address these challenges. As a result, different problems require tailored approaches, necessitating ongoing research to achieve better computational results for complex issues, which hinders the widespread application of PINNs.

In this paper, the objective is to present insights into the tuning some of hyperparameters (weights initialization, activation function, neural network architecture) in PINNs for solid mechanics problems, exploring which of them have more impact on precision compared with reference solutions. We go through problems with different dimensions leveraging results from one to another. This procedure can help authors to optimize their time searching for better parameters while facing problems in higher dimensions.

2 PINNS BUILDING BLOCKS

Building a PINN framework to solve a problem is not a difficulty task, indeed it is simple and straightforward. In this section, we present the details of the PINN methodology before providing the applications. The main ingredients in PINNs building blocks are: the selection of collocation points, the selection of neural network architecture, automatic differentiation, the loss function and the neural network training. Fig. 1 summarizes the process. For more detailed understanding and discussing on each component mentioned, we referrer to Cuomo et al. [8] and Ryu et al. [9].

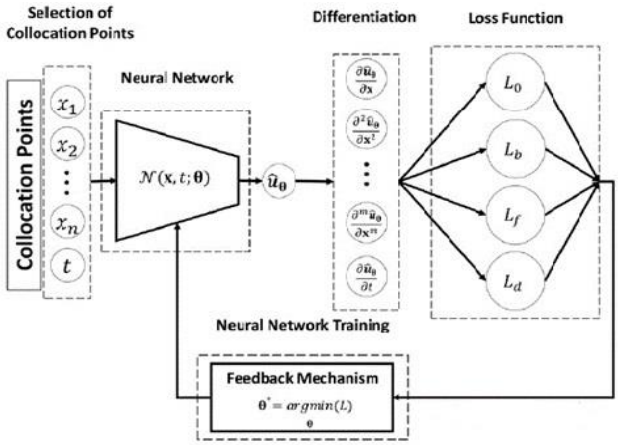


Figure 1. PINNs workflow. Adapted from Ryu et al. [9].

On establishing a general mathematical background which supports PINNs method, we may start with collocation points. At first, the collocation points are selected, then, a neural network architecture $\mathcal{N}(\boldsymbol{z};\boldsymbol{\theta})$, as example a fully connected network, receives as input the coordinates vector $\boldsymbol{z} = [\boldsymbol{x}, t] = [x_1, \dots, x_n, t]$, representing space, time or both of them. After that, the approximation to the desired output is computed. Such neural network approximation of the output $\boldsymbol{u}$ is:

$$\hat{u}_\theta(\boldsymbol{z}) \approx u(\boldsymbol{z}). \tag{1}$$

Where $\hat{u}_\theta$ a neural network approximation realized with a set of neural parameters $\boldsymbol{\theta}$**.** At this stage, the automatic differentiation (AD) takes place to compute the derivatives needed for the governing equation. As each problem demands a specific loss function, they can be divided into four loss terms in PINNs: L_0 for initial conditions, L_b for boundary conidiations, L_f for governing equations, and L_d for labeled data within the computational domain, if it is available.

A weighted sum of the four loss terms in eq. (2) can be used to represent the total loss function to be minimized during the neural network training.

$$L(\boldsymbol{\theta}) = \omega_0 L_0 + \omega_b L_b + \omega_f L_f + \omega_d L_d. \tag{2}$$

$$L_0 = \frac{1}{N_0}\sum_{i=1}^{N_0}\left|g(\boldsymbol{x}_0^i, t=0)\right|, \tag{3}$$

$$L_b = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|g(\boldsymbol{x}_b^i, t_b^i)\right|, \tag{4}$$

$$L_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f(\boldsymbol{x}_f^i, t_f^i)\right|, \tag{5}$$

$$L_d = \frac{1}{N_d}\sum_{i=1}^{N_d}\left|u(\boldsymbol{x}_d^i, t_d^i) - u^i\right|. \tag{6}$$

Here, g and f denote general arbitrary initial and boundary functions and also differential equations, N_0 and N_b represent the number of collocation points for initial $\{\mathbf{x}_0^i, \mathrm{t}\}_{i=1}^{N_0}$ and boundary conditions $\{\mathbf{x}_\mathrm{b}^i, \mathrm{t}_\mathrm{b}^i\}_{i=1}^{N_b}$, respectively. N_f denotes the number of collocation points in the domain, $\{\mathbf{x}_f^i, \mathrm{t}_f^i\}_{i=1}^{N_f}$. If there are additional labeled data available to include in the training, N_d is denoting the number of these points in the computational domain, $\{\mathbf{x}_d^i, \mathrm{t}_d^i\}_{i=1}^{N_d}$.

According on the type of physics problem to be solved and the availability of data, some weights can be zero.

Finally, to update the weights during the learning processes, PINNs need to define and optimizer. In the initial work of Raissi et al. [1], the Adam optimizer was chosen, which is a type of stochastic gradient descent algorithms. Moreover, BFGS, L-BFGS and L-BFGS-B can be applied, and the combination between them is also possible. Now, the solution of the governing equation is given by the optimal neural network parameter set $\boldsymbol{\theta}^*$, that minimizes the loss function, eq. (7).

$$\boldsymbol{\theta}^* = arg\ min_{\theta} L(\boldsymbol{\theta}). \tag{7}$$

As we described the general PINN framework, Tab.1 briefly summarize how the change in each component enhance or disturb the performance. For readers interested in more details about those components' details, Ruy et al. [8] must-know reference. It is important to mention, that the computational cost refers to training time and number of operations to reach the desirable accuracy. However, most important is to define a proper neural network architecture. As we can see, most of issues are related to this item, indicating that a good understanding about neural networks architectures can save time and computational effort.

Table 1. PINNs components and their main related issues.

Components	Related issues
Selection of collocation points	Precision and computational cost
Neural Network Architecture	Problem's nature, scalability, generalization, training efficiency, training parameters, and computational cost
Loss function	Precision and computational cost
Differentiation	Computational cost
Training method	Generalization and computational cost

3 PINNS IN SOLID MECHANICS AND COMPLEX GEOMETRIES

The field of computational solid mechanics relies heavily on numerical methods, as exact solutions are available only for linear and a few nonlinear problems. One well-established numerical method is the Finite Element Method (FEM), which can handle both linear and nonlinear problems. However, FEM struggles with incorporating data into the analysis, requiring more time to process and complete the analysis. PINNs are designed to solve either problems with or without data.

Due to this, and other advantages, the application of PINNs in solid mechanics is an open study field. One of the first applications was made by Haghighat et al. [9] on modeling linear elasticity using PINNs. The same author proposed the mixed variables (displacement and stress) with individual networks for each. Moving to material properties, Abueidda et al. [10] integrated the potential energy functional and the residuals of the governing equations for hyperelasticity.

For an extensive review of applications in solid mechanics, the authors refer to Faroughi et al. [11], Kim and Lee [12], and Cuomo et al. [13]. These researchers point out many application scenarios with different neural network architectures. Furthermore, we would like to address some other interesting examples. Bastek and Kochman [14] and Kairanda et al. [15] proved that PINNs are applicable to shells.

Despite the advances and the great effort to broaden the applications, PINNs still face difficulties in more complex geometric domains, which are crucial for real-world problems. While methods such as signed distance functions and domain decomposition have been explored to introduce complexity, Faroughi et al. [11], they are mostly applied to 2D shapes. Extending these approaches to 3D surfaces remains challenging, especially in ensuring vector fields stay tangent to surfaces, as noted by Cuomo et al. [13].

PINNs are still in their infancy and are no match for the FEM. Grossman et al. [16] compared both methods in a variety of benchmark partial differential equations and had a special treatment to arrange a fair set up for it to provide coherent results. They concluded PINNs are not able to beat FEM considering processing time and accuracy. Nevertheless, they pointed out that PINNs were good at the transition into higher dimensions: there was no increment in computational cost from the Poisson equation in 2D and 3D. This hints at the efficiency of PINNs in certain high-dimensional settings, in which classical techniques are expensive.

4 NUMERICAL EXPERIMENTS

In this section, we present the main results related to the selected problems. Numerical examples are provided to demonstrate the performance PINNs while varying some hyperparameters. The examples are proposed in a way ones can infer relations between hypermeters and precision. Then, leverage the results for a high dimension problem. Thus, computational time and cost can be optimized. In these examples, we followed Haghighat et al.[9] scheme, matching each output field with his own neural network.

The linear solid mechanics problems have the following governing equation

$$\sigma_{ij,j} + f_i = 0, x \in \boldsymbol{\Omega}. \tag{8}$$

Where σ is the Cauchy stress tensor, f is the body force per unit of mass. Alson considering small deformations, the stress tensor is calculated by

$$\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right), \tag{9}$$

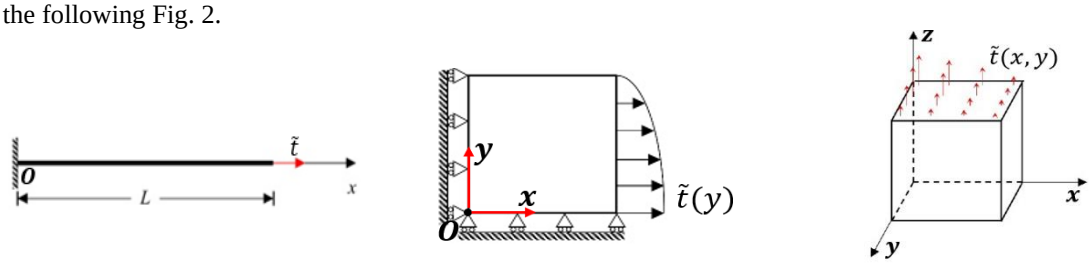
$$\sigma_{ij} = \lambda\delta_{ij}\varepsilon_{kk} + 2\mu\varepsilon_{ij}. \tag{10}$$

Where λ and μ are the Lamé constants and δ_{ij} is the Kronecker delta function. ε_{ij} is the small strain tensor and u is the displacement. To solve the eq. (8), closed boundary conditions are required, which can be written as

$$u_i = \bar{u}_i, x \in \Gamma_u, \tag{11}$$

$$\sigma_{ij}n_j = \bar{t}_i, x \in \Gamma_t. \tag{12}$$

Displacement and force are represented by u and t on the corresponding boundaries respectively, and n denotes the unit normal vector on the corresponding boundaries. So that, the problems to be studied are presented in the following Fig. 2.



(a) (b) (c)

Figure 2. a) bar under axial force, b) plate under stretching force, c) cube under stretching force.

The loss function is one of the main ingredients of the problem. In this work, we adopted the collocation loss function, however there is the energy-based loss function Fugh and Bouklas [3]. The collocation loss function is straightforward and enforces the boundary conditions directly, what is important in PINNs-based computational solid mechanics problems.

Thus, the collocation loss function is built summing up the Mean Square Error (MSE) from the governing equation and the boundary conditions, considering n as the total number of sample points, and m as the number of sample points on the traction boundary.

$$L = \frac{1}{n}\sum_{i=1}^{n}\left|\sigma_{ij,j}^{a} + f_i^a\right| + \frac{1}{m_t}\sum_{i=1}^{m_t}\left|\sigma_{ij}^{a}n_j - \bar{t}_i^a\right|. \tag{13}$$

To quantify the prediction error in PINN scheme, we define the relative mean square (ReMS) error as follows:

$$e_{RMS} = \frac{1}{n}\sum_{i}^{n}\left(\frac{\varphi_{ref,i} - \varphi_{PINN,i}}{\varphi_{ref,i}}\right)^2. \tag{14}$$

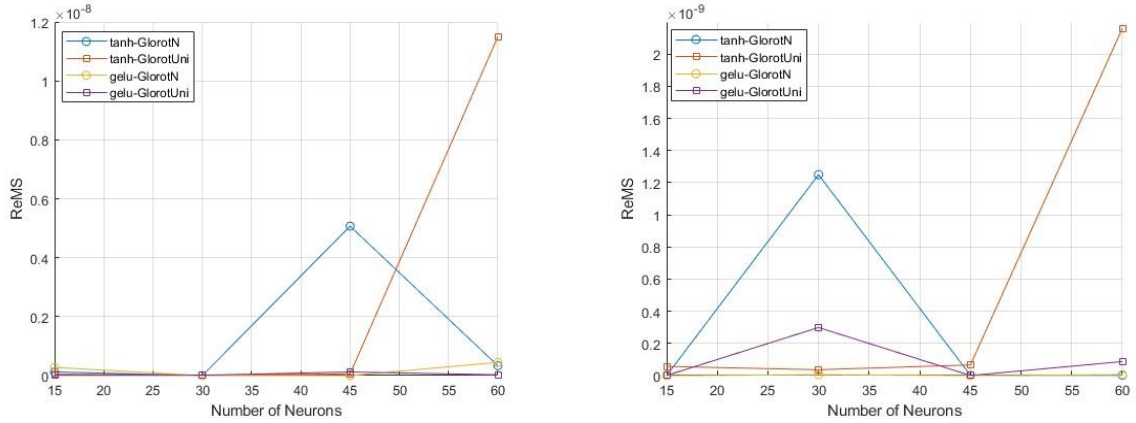
Where $\varphi_{\mathrm{ref,i}}$ and $\varphi_{\mathrm{PINN,i}}$ are the reference values, from analytical or FEM solutions, and PINN prediction variables respectively. The optimizer for the examples is L-BFGS-B. The next results were all tested on a 64-bit Windows system with an Intel® seventh gen Core i5™ -7200U CPU (2.50GHz) and 16 GB memory ram.

4.1 Stretching linear rod

Herein, to complete the analysis, a 1D tensile load in x direction is applied on a linear elastic rod. The problem configuration is given in Fig. 2a**.** The length of the rod is 1 m, the Young's modulus E = 10 Pa, the area (A) is equal to 1 m². A stretching force $\tilde{t}$ = 1 N is applied at the right end of the rod. So that, the equilibrium equations and the applied boundary conditions are $u(x=0)=0$, $\sigma_x(x=l)=1$ and $\sigma_{x,x}=0, x \in [0,1]$. In this case, we have the analytical solution, eq. (11), in terms of displacement u(x)

$$u(x)=\frac{x\tilde{t}}{EA}. \tag{15}$$

The Fig. 3 summarizes the main between distribution of neurons (width or deep) and ReMS, also including different weights initialization methods and activation functions. The simulation was carried out with 51 equally spaced sample points.



(a) (b)

Figure 3. Rod simulation with a) width neural networks, b) deep neural networks.

4.2 Stretching square plate

The program is extended to a 2D plate under stretching in plane stress condition. The configuration of the problem is shown in the Fig. 2(b) below. The length of the plate is 2 m and a distributed force $\tilde{t}(y)$is applied on the right side of the plate.

$$\tilde{t}(y)=sin\left(\pi-\frac{\pi y}{2}\right). \tag{16}$$

The boundary conditions are $u(0,y)=v(x,0)=0$. In this problem, the Lamé constants are calculated through eq. (17)

$$\lambda=\frac{E\upsilon}{(1+\upsilon)(1-\upsilon)}, \mu=\frac{E}{2(1+\upsilon)}. \tag{17}$$

Setting 7 Pa and 0.3 to Young's modulus and Poisson's ratio. To compare the results with FEM numerical software, a fine mesh from ABAQUS was used as reference, where quadratic elements are 0.005 m equally spaced. This problem is interesting to explore because the stress concentration of σ_y at the bottom right corner of the plate can represent a test for PINNs accuracy while facing these features.

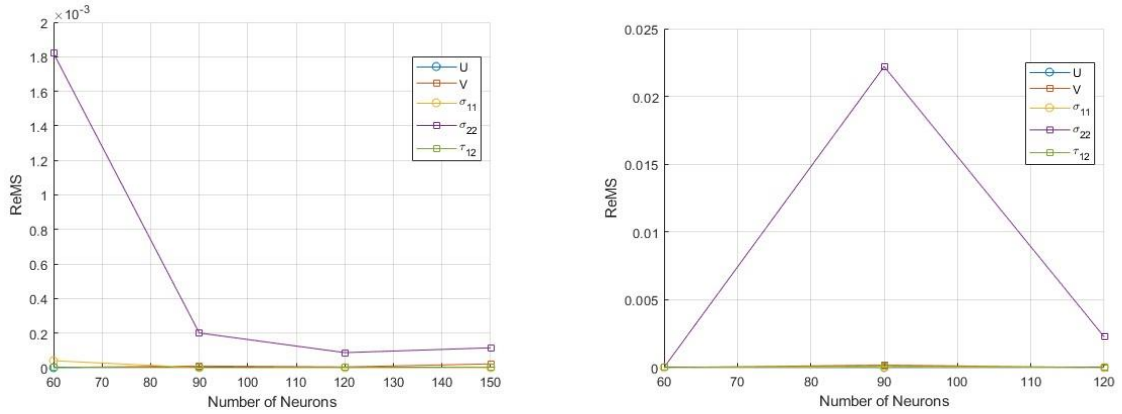
The main results are presented in Fig. 4**.** Leveraging the results in section 4.1, we only simulated the combination between *gelu* activation function and *Glorot Normal* weights initialization for the neural network architecture, expecting to extract hints about stresses and displacements fields. The domain was subdivided into 2601 sample points.

4.3 Stretching in 3D

A cube under axial tension is taken as model to verify PINN's performance in space and symmetrical properties. The configuration is shown in Fig.2(c). The cube size is 2 m and a distributed force $\tilde{t}(x, y)$,eq. (18), is applied on the surfaces.

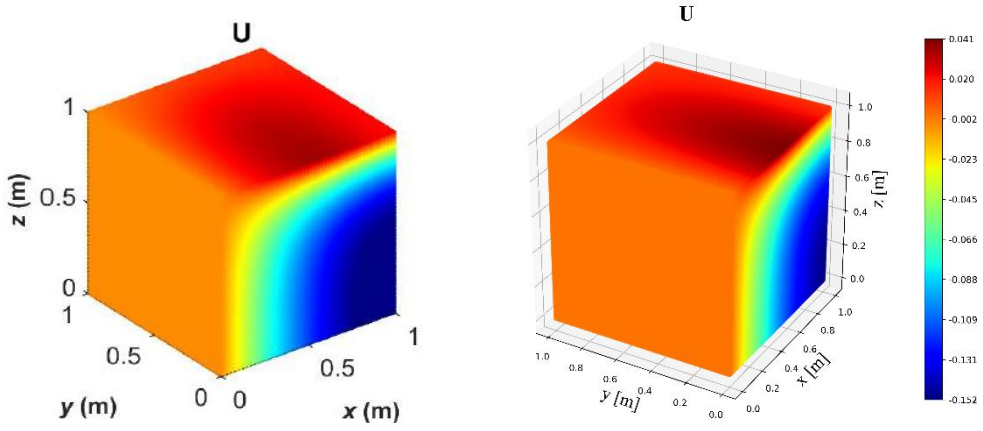
$$\tilde{t}(x, y) = sin\left(\pi - \frac{\pi x}{2}\right) sin\left(\pi - \frac{\pi y}{2}\right). \tag{18}$$

The boundary conditions are given as $u(0, y, z) = v(x, 0, z) = w(x, y, 0) = 0$. The material parameters are 10 Pa and 0.25 for Young's modulus and Poisson's ratio. The weights initialization method (*Glorot Normal*), activation function (*gelu*), number of layers (5) and neurons per layers (16) were customized based on the results from previous 2 sections. Overall, 9260 sample points were generated. A fine mesh, element size equal to 0.01, was applied in the commercial software ABAUQUS for comparison. The total simulation time was about 4512 s for a total number of iterations in 5001, the final loss converged to 0.2906. The results are summarized in Fig. 5.



(a) (b)

Figure 4: Plate problem results for: a) width neural networks, b) deep neural networks.



(a) (b)

Figure 5: 3D problem displacement field U results for (a) FEM, (b) PINN.

5 CONCLUSIONS AND DISCUSSIONS

Tuning parameters in neural networks is a challenging field, particularly when identifying the optimal settings

for different problem types. In this study, we explored neural network configurations by analyzing progressively complex toy problems. We used separate neural networks for each displacement field, which provided control over neuron count and allowed extension to asymmetrical cases. Deeper neural networks consistently outperformed wider ones, and using the *gelu* activation function with *Glorot Normal* weight initialization yielded stable outcomes.

In the plate problem, we observed instability for tension in the y-direction with a deep neuron configuration. This insight informed the neural architecture selection for subsequent problems. Higher-dimensional problems presented significant computational challenges due to the "curse of dimensionality." Though our 3D example produced good results, computational time remains a concern.

Studying lower-dimensional problems to inform hyperparameter tuning for more complex cases may offer an efficient strategy for reducing effort. Future work should focus on refining the tuning process for PINNs to enable their application to more complex problems.

Acknowledgements. F.V.B. Rodrigues acknowledges the support by Comissão de Aperfeiçoamento Pessoal de Nível Superior - Brazil (CAPES). P.M Pimenta also acknowledges the support by CNPq under the grant 308142/2017-7, abd the Alexander von Humboldt Foundation for the Georg Foster Award that made possible his stays at the University of Duisburg-Essen and Hannover – Germany.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

1. Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys. 2019 Feb;378:686–707.
2. Abueidda DW, Lu Q, Koric S. Meshless physics-informed deep learning method for three-dimensional solid mechanics. Int J Numer Methods Eng. 2021 Dec 15;122(23):7182–201.
3. Fuhg JN, Bouklas N. The mixed Deep Energy Method for resolving concentration features in finite strain hyperelasticity. J Comput Phys. 2022 Feb;451:110839.
4. Nguyen-Thanh VM, Zhuang X, Rabczuk T. A deep energy method for finite deformation hyperelasticity. Eur J Mech - ASolids. 2020 Mar;80:103874.
5. Goswami S, Anitescu C, Chakraborty S, Rabczuk T. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. Theor Appl Fract Mech. 2020 Apr;106:102447.
6. Wang Y, Zhong L. NAS-PINN: Neural architecture search-guided physics-informed neural network for solving PDEs. J Comput Phys. 2024 Jan;496:112603.
7. Psaros AF, Kawaguchi K, Karniadakis GE. Meta-learning PINN loss functions. J Comput Phys. 2022 Jun;458:111121.
8. Cuomo S, Di Cola VS, Giampaolo F, Rozza G, Raissi M, Piccialli F. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. J Sci Comput. 2022 Sep;92(3):88.
9. Ryu I, Park GB, Lee Y, Choi DH. Physics-informed neural network for engineers: a review from an implementation aspect. J Mech Sci Technol. 2024 Jul;38(7):3499–519.
10. Haghighat E, Raissi M, Moure A, Gomez H, Juanes R. A deep learning framework for solution and discovery in solid mechanics [Internet]. arXiv; 2020 [cited 2024 Jul 20]. Available from: http://arxiv.org/abs/2003.02751
11. Abueidda DW, Koric S, Guleryuz E, Sobh NA. Enhanced physics-informed neural networks for hyperelasticity. Int J Numer Methods Eng. 2023 Apr 15;124(7):1585–601.
12. Faroughi SA, Pawar NM, Fernandes C, Raissi M, Das S, Kalantari NK, et al. Physics-Guided, Physics-Informed, and Physics-Encoded Neural Networks and Operators in Scientific Computing: Fluid and Solid Mechanics. J Comput Inf Sci Eng. 2024 Apr 1;24(4):040802.
13. Kim D, Lee J. A Review of Physics Informed Neural Networks for Multiscale Analysis and Inverse Problems. Multiscale Sci Eng. 2024 Mar;6(1):1–11.
14. Bastek JH, Kochmann DM. Physics-Informed Neural Networks for shell structures. Eur J Mech - ASolids. 2023 Jan;97:104849.
15. Kairanda N, Habermann M, Theobalt C, Golyanik V. NeuralClothSim: Neural Deformation Fields Meet the Thin Shell Theory [Internet]. arXiv; 2024 [cited 2024 Jul 20]. Available from: http://arxiv.org/abs/2308.12970
16. Grossmann TG, Komorowska UJ, Latz J, Schönlieb CB. Can physics-informed neural networks beat the finite element method? IMA J Appl Math. 2024 Jun 21;89(1):143–74.