A R C H I V E O F M E C H A N I C A L E N G I N E E R I N G

DOI: 10.24425/ame.2026.1885 Early Access; pp. 1–14

Oleksandr KUCHERENKO [1]

Physics-Informed Deep Learning for Modeling of Cylindrical Shells

Received 4 December 2025, **Revised** 18 April 2026, **Accepted** 21 May 2026, **Published online** 6 July 2026

Keywords: PINN, deep learning, loss function, ensemble, shell, deformation

This study revisits the canonical problem of the structural response of a cylindrical shell under hydrostatic pressure using physics-informed neural networks. Particular attention is given to the strong formulation of the governing differential equation, which poses significant challenges due to the need for accurate evaluation of higher-order derivatives. To enforce boundary conditions, a modified loss function with embedded hard constraints is proposed. The model hyperparameters are identified through a series of numerical experiments. The results indicate that physics-informed neural networks may exhibit inherent limitations that hinder their ability to accurately capture all features of the exact solution. To address these limitations, a hybrid approach integrating physics-based modeling and statistical learning theory via ensemble techniques is proposed. The ensemble framework demonstrates improved generalization and reliability, suggesting its potential to enhance the performance of physics-informed neural networks in solving differential equations.

Nomenclature

PINN	physics-informed neural network
DE	differential equation
ELM	extreme learning machine
BC	boundary conditions
HC	hard constrains

✉ Oleksandr KUCHERENKO, e-mail: ak_sci@proton.me

[1]The Institute of Technical Mechanics of the National Academy of Sciences of Ukraine and the State Space Agency of Ukraine, Dnipro, Ukraine

$\mathcal{NN}$	neural network
$x,\ z$	rectangular coordinates
$w,\ \widehat{w}$	deflection in the z-direction and its approximation
h	length of a shell
t	thickness of a shell
r	radius of a shell
E	modulus of elasticity
ν	Poisson's ratio
D	flexural rigidity of a shell
Z	load on a shell parallel to the z-axis
β	shell parameter
γ	weight per unit volume of the liquid
$\mathcal{L}$	loss function
θ	neural network parameters
Ω	domain
$\partial\Omega$	domain boundary
$\mathcal{F}$	discrepancy function
ℓ	ansatz

1. Introduction

In recent years, deep learning in the form of neural networks has been used in many areas of physics [1] and engineering [2, 3], including electromagnetism [4, 5], mechanics of solids [6, 7], fluid dynamics [8, 9], aerodynamics [10, 11], and materials science [12, 13]. Deep neural networks are usually employed as the dominant approach for data-driven problems to learn the nonlinear mapping from input to output. The success of deep learning can be attributed to a combination of new theoretical approaches and advances in hardware, such as general-purpose graphics processing units [14]. Usually, it requires a substantial amount of data to train a neural model for a complex problem, which leads to the need to generate and store huge datasets in order to achieve a reliable input-output mapping.

Alternatively, we can regard neural networks as DE solvers. In this context, neural networks with a sufficient number of neurons have traditionally been used. According to the universal approximation theorem [15], they can model any continuous function with arbitrary precision, thereby providing solutions to a broad class of mathematical and physical problems.

Rather than relying on the brute-force approximation of differential equations using generated arrays of data, it is possible to solve them by minimizing a loss function derived from the DE in the form of, e.g., the mean squared residuals. These

physics-informed neural networks [16–18] are mesh-free by design and, in general, do not need large training datasets. The core idea is to incorporate physical information into the model. They virtually fall into the category of weakly-supervised deep learning. Fig. 1 highlights the key distinctions between neural modeling approaches with and without access to ground-truth data. A brief historical overview of the development of deep neural network methodologies for solving differential equations can be found in [19]. Alternative numerical modeling methods include the Finite Element Method [20], Finite Volume Method, and Finite Difference Method [21], which are traditionally applied to solve numerical problems governed by differential equations. In physics-informed neural models, we exploit their property

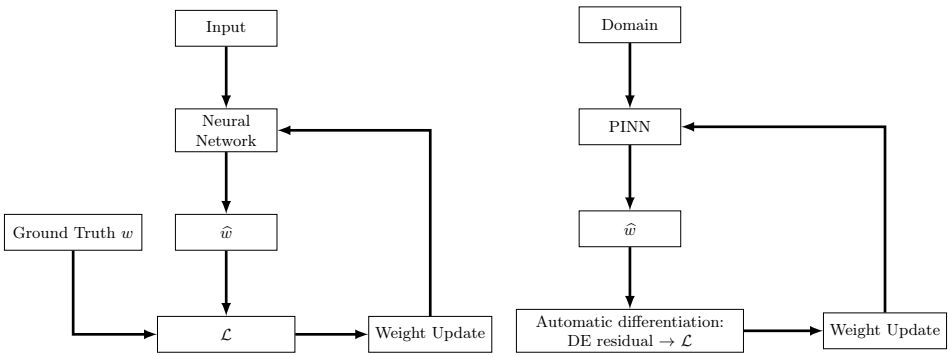

Fig. 1. Training scenarios for a neural network.

as universal approximators to solve DEs [22], but instead of evaluating the loss between the predicted and ground truth solutions, we incorporate the differential equation into the loss function. Basically, the loss function can be associated with initial and boundary conditions via soft or hard constraints [23]. Of particular significance is the fact that PINNs are actively used for solving forward and inverse problems [17]: for forward problems, governing DEs and boundary conditions are well-known; for inverse problems, DEs are known, whereas corresponding boundary conditions might be fully or partially unknown. Automatic differentiation [24] is employed to find the derivative of a variable of interest (as the network output) with respect to the network input.

The application of PINNs to problems of continuum mechanics has been extensively discussed. For example, the study [25] focuses on modeling of turbulent flows, demonstrating improved predictive accuracy and generalization compared to conventional data-driven approaches. In particular, two robust turbulence models based on a feedforward fully connected $\mathcal{NN}$ for large eddy simulation have been developed to capture nonlinear flow regimes. In [26], a data-driven deep learning framework is proposed for solving linear elasticity problems, enabling mesh-free approximation of displacement fields. The results demonstrate that the approach can accurately reproduce reference solutions, although limitations may include sen-

sitivity to the training strategy and difficulties in capturing complex solution features. In [27], a physics-informed deep learning approach is developed for modeling of non-associative Drucker–Prager elastoplastic behavior. It integrates constitutive equations with a multi-objective loss function, combining physical constraints with data-driven learning and allowing for accurate prediction of elastoplastic responses. The work [28] presents a data-driven, physics-constrained deep learning framework for elastoplastic problems. The method accurately predicts stress–strain responses and captures plastic deformation behavior. A shared characteristic of the last three studies is the application of transfer learning and the use of multi-objective loss functions, which may introduce sensitivity to the balancing of loss terms.

In this study, we focus on the classical problem of a cylindrical shell loaded symmetrically with respect to its axis (Fig. 2). The theoretical foundation for the application of PINNs to shell structures is detailed in [29]. In the context of related work, the authors mention Extreme Learning Machines as an alternative neural modeling approach, which can be coupled with PINN-type formulations to improve computational efficiency. ELMs are characterized by a single hidden layer with randomly assigned weights that remain fixed during training, while only the output layer is optimized. This leads to significantly reduced training time; however, such approaches may suffer from reduced accuracy, thereby limiting their applicability in purely physics-driven settings. The PINN model used by the authors in [29] is a feedforward neural network that operates in a non-Euclidean domain, with the governing equations of the linear Naghdi shell model embedded into the loss function. The authors concluded that PINNs are capable of accurately identifying the solution field in the benchmark problems when the governing equations are formulated in their weak form. However, they may fail when the equations are expressed in their strong form.

Nonetheless, this paper explicitly targets the strong formulation of the governing DE, rather than relying on weak (variational) forms that are more commonly adopted in contemporary computational frameworks. This focus is nontrivial, as such an approach requires an accurate representation of higher-order derivatives. The study, therefore, addresses a known limitation of PINNs, that is, their tendency to inadequately resolve localized features. By framing limitations within the given context, the work contributes to a more rigorous understanding of the approximation capabilities of PINNs. The study advances a hybrid approach that bridges physics-based modeling with statistical learning theory by leveraging ensemble techniques, which enables improved generalization and reliability. Finally, the work aims to establish a framework for addressing problems involving complex load patterns such as fluid-structure interaction [30], shells with initial imperfections [31], seismic loading, blast pressure effects, and other dynamic conditions.

The paper is structured as follows: in Section 2, we briefly introduce the mathematical representation of the benchmark problem. Section 3 addresses key aspects of PINNs relevant to the problem (loss functions, boundary conditions,

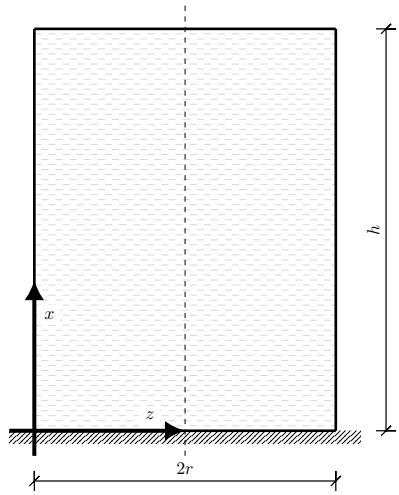

Fig. 2. Cylindrical shell under hydrostatic pressure.

automatic differentiation, architecture). Section 4 details the training process of the PINNs, aimed at solving the governing equation. Finally, in Section 5, we present and analyze the results, followed by a concluding summary.

2. Cylindrical Shell Loaded Symmetrically

Symmetrical deformation of a circular cylindrical shell w (Fig. 2) under the load Z, assuming constant shell thickness t, is governed by the following equation [32]:

$$D\frac{d^4w}{dx^4} + \frac{Et}{r^2}w = Z, \tag{1}$$

where $D = \dfrac{Et^3}{12(1-\nu^2)}$.

Assuming that $\beta^4 = \dfrac{3(1-\nu^2)}{(rt)^2}$ and $Z = -\gamma(h-x)$, we can simplify Eq. (1):

$$\frac{d^4w}{dx^4} + 4\beta^4 w = -\frac{\gamma(h-x)}{D}. \tag{2}$$

The general solution to this equation is of the following form:

$$w = e^{\beta x}(C_1 cos\beta x + C_2 sin\beta x) + e^{-\beta x}(C_3 cos\beta x + C_4 sin\beta x) - \frac{\gamma(h-x)r^2}{Et}, \tag{3}$$

where $C_1, ..., C_4$ are the constants of integration. In fact, t is small compared to r and h, so we can consider the shell to be infinitely long, which gives us $C_1 = C_2 = 0$.

The boundary conditions for the lower edge of the shell built into a rigid foundation are:

$$\begin{aligned} w_{BC} &= w_{x=0} = 0, \\ \left(\frac{dw}{dx}\right)_{x=0} &= 0. \end{aligned} \tag{4}$$

Then, C_3 and C_4 can be expressed as follows:

$$C_3 = \frac{\gamma r^2 h}{Et}, \quad C_4 = \frac{\gamma r^2}{Et}(h - \frac{1}{\beta}). \tag{5}$$

Finally, taking into account Eq. (5), we obtain:

$$w = -\frac{\gamma r^2}{Et}(h - x - e^{-\beta x}(h\ cos\ \beta x + (h - \frac{1}{\beta})\ sin\ \beta x)). \tag{6}$$

For the sake of simplicity, we treat this problem as 1D with a single input parameter x, which is the coordinate along the x axis (see Fig. 2). In our numerical experiments, we assume that $r = 5.215\,\text{m}$, $h = 11.92\,\text{m}$, $t = 4 \cdot 10^{-3}\,\text{m}$, $\nu = 0.3$, $E = 2 \cdot 10^{11}\,\text{Pa}$, $\gamma = 9810\,\frac{\text{N}}{\text{m}^3}$. Eq. (6) serves as the ground truth for the deep models considered in this study. We use this closed-form solution to evaluate the performance of the PINNs.

3. Model architecture

In physics-informed deep learning, a deep neural network is employed to approximate a solution of a differential equation via automatic differentiation capabilities of such frameworks as Tensorflow or Pytorch [33]. Deflection $\hat{w}$ can be approximated in the following way:

$$\hat{w}(x) = \mathcal{NN}(x; \boldsymbol{\theta}), \tag{7}$$

where $\mathcal{NN}(x; \boldsymbol{\theta})$ is the deep neural network with adjusted parameters.

To embed governing physical laws into the neural network training procedure, we need to modify the network loss function so that it reflects the principles of complex physical phenomena. The loss function of the PINN might include soft or hard constraints in order to satisfy boundary conditions of the problem (4).

The PINN loss function with soft constraints may be defined as follows:

$$\begin{aligned} &\mathcal{L} = \mathcal{L}_{BC} + \lambda \mathcal{L}_{DE}, \\ &\mathcal{L}_{BC} = \mathcal{F}(\hat{w}(\boldsymbol{x}),\ w_{BC}(\boldsymbol{x})),\ \boldsymbol{x} \in \partial\Omega, \\ &\mathcal{L}_{DE} = \mathcal{F}\left(\frac{d^4\hat{w}(\boldsymbol{x})}{dx^4} + 4\beta^4\hat{w}(\boldsymbol{x}) + \frac{\gamma(h - \boldsymbol{x})}{D},\ \mathbf{0}\right),\ \boldsymbol{x} \in \Omega, \end{aligned} \tag{8}$$

where $\boldsymbol{x}$ is a set of points in the domain Ω with the boundary $\partial\Omega$, λ is the fixed penalty coefficient of the constraint (a weighting factor), and $\mathcal{F}$ measures the discrepancy between its vector arguments.

The best form of the function $\mathcal{F}$ is defined by the cross-validation procedure. The penalty coefficient λ is defined experimentally as well. Fig. 3 illustrates a common issue with soft constraints: one loss (in this case, $\mathcal{L}_{BC}$) often dominates the others, requiring careful adjustment of λ. Furthermore, a large penalty coefficient may result in an ill-conditioned optimization problem, which fails to converge. To mitigate the stated issue, the boundary conditions may be imposed through the loss function by applying hard constraints [34]. Let us write the solution in Ω as

$$\hat{w}_{HC}(x) = g(x) + \ell(x)\mathcal{NN}(x;\boldsymbol{\theta}), \tag{9}$$

where $g(x)$ satisfies some initial values, and ℓ satisfies the following conditions:

$$\begin{aligned} &\ell(x) = 0,\ x \in \partial\Omega, \\ &\ell(x) > 0,\ x \in \Omega - \partial\Omega. \end{aligned} \tag{10}$$

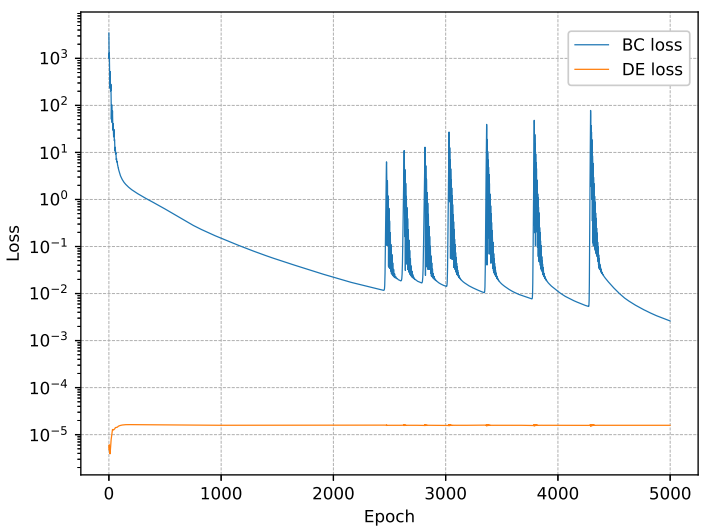

Fig. 3. Soft constraints—example of BC and DE losses.

For simple geometries, the function ℓ can be chosen analytically. For $\Omega = [0, h]$, $\partial\Omega = \{0, h\}$, we can choose $\ell(x) = (x-0)(h-x)$. Considering Eq. (9, 10), we express the loss function with hard constraints as follows:

$$\mathcal{L}_{HC} = \mathcal{F}(\hat{w}_{HC}(\boldsymbol{x}),\ \boldsymbol{0}),\ \boldsymbol{x} \in \Omega. \tag{11}$$

We will use this option further in our analysis.

Fig. 4 depicts the schematic representation of the PINN, namely a fully connected feedforward network for one-dimensional Eq. (2), where the input x represents the spatial coordinate, and the output $\hat{w}$ represents the approximated solution. The deep neural network $\mathcal{NN}$, which comprises k hidden layers, each consisting of n neurons with nonlinear activation functions a (the activation function of the output layer is always linear), is an approximator that estimates the solution $\hat{w}(x)$ of Eq. (2). We use PyTorch's torch.autograd.grad() function for automatic differentiation.

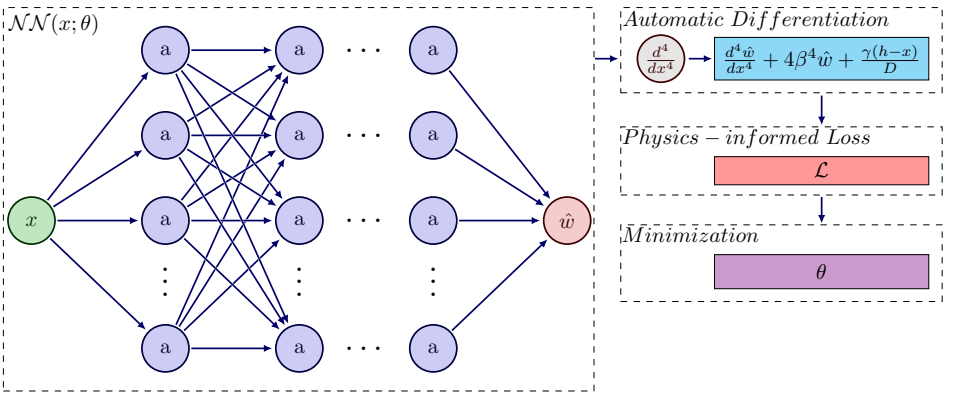

Fig. 4. Schematic of a PINN solving Eq. (2).

The near-optimal hyperparameters of the PINN are selected using the holdout method, which effectively performs an ablation-like analysis by comparing configurations with added or removed components. Table 1 lists the hyperparameters, combinations of which are considered in this article.

Table 1. Hyperparameters of PINN

Layers	Neurons	Epochs	Activation function	Loss function $\mathcal{F}$
4, 8, 16	8, 16, 32	6000	ReLU, SELU	L1, MSE, Huber

4. PINN training

In this study, we employ the PyTorch machine learning framework. The neural networks are trained using the Adam optimization algorithm, with a variable learning rate dynamically adjusted by the ReduceLROnPlateau scheduler to enhance convergence. For all experiments, we use a batch size of 512. The training data are obtained by randomly sampling 2000 points from the one-dimensional domain $\Omega = [0, h]$ along the x-axis. Network weights are initialized using the Kaiming uniform initialization method to promote convergence and mitigate vanishing or exploding gradient issues. As shown in Table 1, each PINN is trained for a total of

6000 epochs. The training process is accelerated using an Nvidia Tesla P100 GPU to enhance computational efficiency.

The network architectures of the three top-performing PINNs, optimized using Huber, MSE, and L1 loss functions, are feedforward with a single input and a single output. The number of layers, neurons per layer, and activation functions for each model are summarized in Table 2. Fig. 5 depicts the training loss curves for these PINNs, showing their progressive decrease over iterations, which indicates convergence of the models toward a solution.

Table 2. Hyperparameters for the best-performing PINNs

#	Layers	Neurons in each layer	Activation	Loss $\mathcal{F}$	Value
1	16	32	ReLU	Huber	0.001006
2	8	8	ReLU	MSE	0.516541
3	16	32	ReLU	L1	0.089040

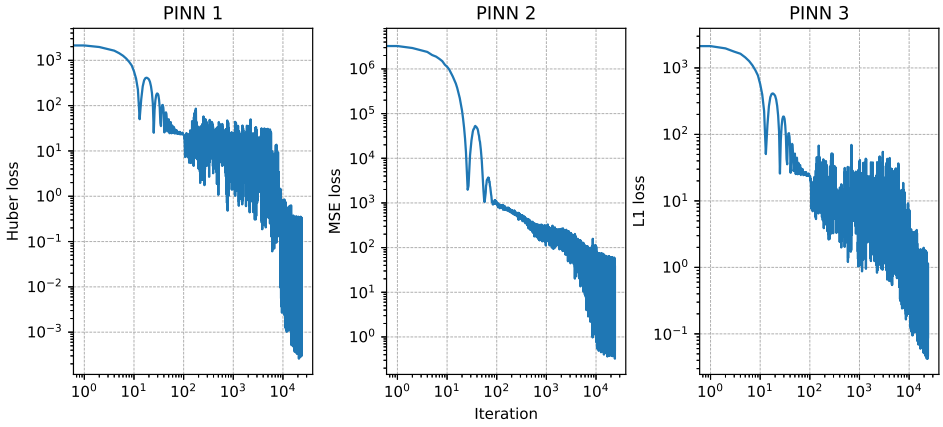

Fig. 5. Training losses for the three best-performing PINNs with different loss functions.

5. Results

In total, we trained 54 models. Fig. 6 illustrates the solutions to the benchmark problem formulated in strong form, obtained using the three top-performing PINNs (Table 2), in comparison with the exact solution given by Eq. (6). None of the models was able to capture the features of the exact solution in the vicinity of the point of maximum deflection, demonstrating inherent limitations despite the relatively complex structure of each neural network. Moreover, PINN2 (with MSE loss function) introduced notable noise into the part of the solution, whereas the neural network models trained with the Huber and L1 loss functions exhibited a systematic overestimation of the maximum deflection.

A possible solution to the problem may lie in a change of the form of the governing equations, imposing additional constraints on the loss function, or utilizing more sophisticated deep learning architectures. We propose to use one of the ensemble methods [35] in order to filter out the erroneous components of the solution. Since we have already trained multiple models ($18 * 3$) in the process of

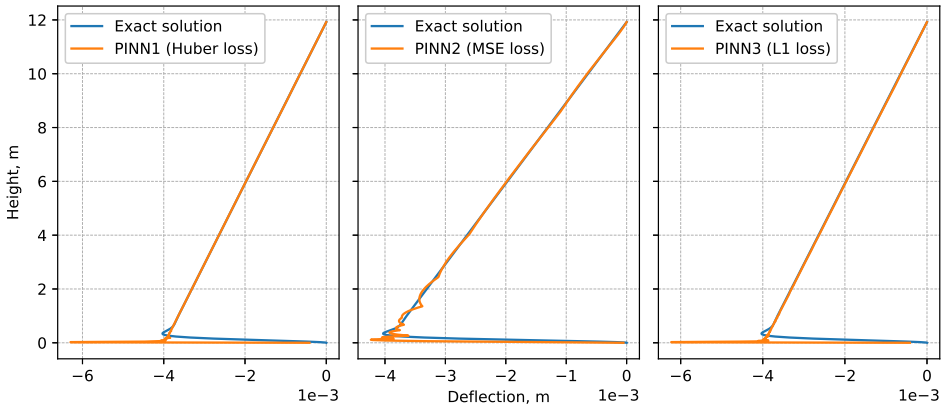

Fig. 6. Solutions obtained by the top-performing PINNs.

hyperparameter optimization, we can reuse these. One of the simplest ensemble methods is ensemble averaging, where several, less satisfactory networks are combined to produce an improved output. A supplementary advantage of this approach is the reduction of variance at no cost. Additionally, we propose the use of the median function instead of simple averaging, as it provides increased robustness to outliers. Fig. 7 shows the general computational process involving a median-based ensemble model.

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \longrightarrow \begin{bmatrix} \mathcal{NN}_1 \\ \mathcal{NN}_2 \\ \vdots \\ \mathcal{NN}_m \end{bmatrix} \longrightarrow \begin{bmatrix} \hat{w}_{11} & \dots & \hat{w}_{1m} \\ \vdots & \ddots & \vdots \\ \hat{w}_{n1} & \dots & \hat{w}_{nm} \end{bmatrix} \longrightarrow \boxed{\begin{array}{c}\text{Median-based} \\ \text{Ensemble}\end{array}} \longrightarrow \hat{\boldsymbol{w}} = \begin{bmatrix} \hat{w}_1 \\ \hat{w}_2 \\ \vdots \\ \hat{w}_n \end{bmatrix}$$

Fig. 7. Computational process with a median-based ensemble model.

Fig. 8 illustrates the solutions obtained by applying a median-based ensemble model. All ensemble models demonstrate a significant performance improvement compared to the top-performing PINNs with corresponding loss functions. The use of an ensemble modeling approach improved the estimation of the maximum deflection and suppressed spurious oscillations.

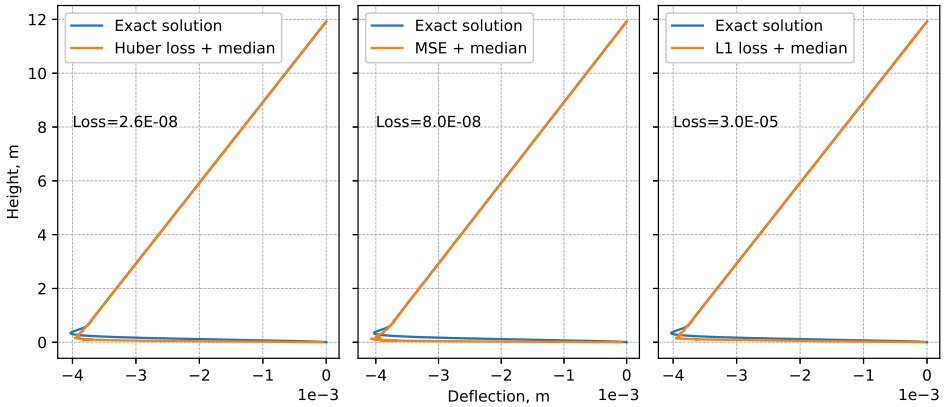

Fig. 8. Solutions obtained by median-based ensemble.

6. Conclusions

In this study, we employed the physics-informed neural network architecture with hard constraints and set up a sequence of numerical experiments to solve the benchmark problem governed by the fourth-order differential equation (1). Numerical experiments were designed to find near-optimal hyperparameters of the PINN.

Although PINNs with hard constraints, as considered in this study, tend to converge faster and yield more accurate predictions than those with soft constraints, they are inherently limited by the requirement to construct an explicit ansatz [36] for embedding boundary conditions. This requirement reduces the architectural flexibility of hard-constrained PINNs and may introduce additional challenges when extending the framework to higher-dimensional or more geometrically complex domains. In this context, one possible approach is the use of geometry-aware methods based on R-functions [37].

Despite the theoretical universal approximation capability of neural networks, the conducted numerical experiments revealed that none of the trained models fully captured all salient features of the exact solution to the benchmark problem, which supports the hypothesis that PINNs may exhibit significant limitations when governing equations are presented in their strong form, especially in the presence of higher-order derivatives.

To address this issue, an ensemble-based strategy was investigated. The results show that even a simple aggregation technique, such as median filtering, yields a noticeable improvement in predictive accuracy and robustness compared to individual models. This suggests that ensemble learning provides an effective pathway for mitigating model deficiencies and enhancing generalization, despite the increased model complexity it entails.

These findings motivate further research on advanced ensemble techniques and geometry-aware methods aimed at improving the robustness and accuracy of PINNs.

References

[1] P.R. Wiecha, A. Arbouet, C. Girard, and O.L. Muskens. Deep learning in nano-photonics: inverse design and beyond. *Photon. Res.*, 9(5):B182–B200, 2021. doi: 10.1364/PRJ.415960.

[2] W. Cao, J. Song, and W. Zhang. Solving high-dimensional parametric engineering problems for inviscid flow around airfoils based on physics-informed neural networks. *Journal of Computational Physics*, 516:113285, 2024. doi: 10.1016/j.jcp.2024.113285.

[3] R. Liang, W. Liu, Y. Fu, and M. Ma. Physics-informed deep learning for structural dynamics under moving load. *International Journal of Mechanical Sciences*, 284:109766, 2024. doi: 10.1016/j.ijmecsci.2024.109766.

[4] H. Chang, J. Fan, R. Wang, and B. Wang. Solving electromagnetic problems with pinn based on scattering equivalent source method. In *2024 IEEE International Symposium on Antennas and Propagation and INC/USNC-URSI Radio Science Meeting (AP-S/INC-USNC-URSI)*, pages 1025–1026, 2024. doi: 10.1109/AP-S/INC-USNC-URSI52054.2024.10686927.

[5] M. Baldan, P. Di Barba, and D. A. Lowther. Physics-informed neural networks for inverse electromagnetic problems. *IEEE Transactions on Magnetics*, 59(5):1–5, 2023. doi: 10.1109/TMAG.2023.3247023.

[6] D.W. Abueidda, Q. Lu, and S. Koric. Meshless physics-informed deep learning method for three-dimensional solid mechanics. *International Journal for Numerical Methods in Engineering*, 122(23):7182–7201, 2021. doi: 10.1002/nme.6828.

[7] H. Hu, L. Qi, and X. Chao. Physics-informed neural networks (PINN) for computational solid mechanics: Numerical frameworks and applications. *Thin-Walled Structures*, 205:112495, 2024. doi: 10.1016/j.tws.2024.112495.

[8] C. Zhao, F. Zhang, W. Lou, X. Wang, and J. Yang. A comprehensive review of advances in physics-informed neural networks and their applications in complex fluid dynamics. *Physics of Fluids*, 36(10):101301, 2024. doi: 10.1063/5.0226562.

[9] R. Sun, H. Jeong, J. Zhao, Y. Gou, E. Sauret, Z. Li, and Y. Gu. A physics-informed neural network framework for multi-physics coupling microfluidic problems. *Computers & Fluids*, 284:106421, 2024. doi: 10.1016/j.compfluid.2024.106421.

[10] V. Sekar, M. Zhang, C. Shu, and B.C. Khoo. Inverse design of airfoil using a deep convolutional neural network. *AIAA Journal*, 57(3):993–1003, 2019. doi: 10.2514/1.J057894.

[11] Y. Sun, U. Sengupta, and M. Juniper. Physics-informed deep learning for flow modelling and aerodynamic optimization. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1149–1155, 2022. doi: 10.1109/SSCI51031.2022.10022215.

[12] K. Guo, Z. Yang, C.H. Yu, and M.J. Buehler. Artificial intelligence and machine learning in design of mechanical materials. *Mater. Horiz.*, 8:1153–1172, 2021. doi: 10.1039/D0MH01451F.

[13] Y. Diao, J. Yang, Y. Zhang, D. Zhang, and Y. Du. Solving multi-material problems in solid mechanics using physics-informed neural networks based on domain decomposition technology. *Computer Methods in Applied Mechanics and Engineering*, 413:116120, 2023. doi: 10.1016/j.cma.2023.116120.

[14] M.A. Shafique, A. Munir, and J. Kong. Deep learning performance characterization on gpus for various quantization frameworks. *AI*, 4(4):926–948, 2023. doi: 10.3390/ai4040047.

[15] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. doi: 10.1016/0893-6080(89)90020-8.

[16] Z.K. Lawal, H. Yassin, D.T.C. Lai, and A.C. Idris. Physics-informed neural network (PINN) evolution and beyond: A systematic literature review and bibliometric analysis. *Big Data and Cognitive Computing*, 6(4), 2022. doi: 10.3390/bdcc6040140.

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

[18] W. Chen, Q. Wang, J. Hesthaven, and C. Zhang. Physics-informed machine learning for reduced-order modeling of nonlinear problems. *Journal of Computational Physics*, 446:110666, 2021. doi: 10.1016/j.jcp.2021.110666.

[19] N. Yadav, A. Yadav, and Kumar M. *An Introduction to Neural Network Methods for Differential Equations*. Springer Dordrecht, 2015. doi: 10.1007/978-94-017-9816-7.

[20] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The Finite Element Method: its Basis and Fundamentals (Seventh Edition)*. Butterworth-Heinemann, 2013. doi: 10.1016/C2009-0-24909-9.

[21] S. Mazumder. *Numerical Methods for Partial Differential Equations: Finite Difference and Finite Volume Methods*. Academic Press, 2015.

[22] I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998. doi: 10.1109/72.712178.

[23] S. Chen, Z. Liu, W. Zhang, and J. Yang. A hard-constraint wide-body physics-informed neural network model for solving multiple cases in forward problems for partial differential equations. *Applied Sciences*, 14(1), 2024. doi: 10.3390/app14010189.

[24] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, and J.M. Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18(1):5595–5637, 2017.

[25] R. Bose and A.M. Roy. Invariance embedded physics-infused deep neural network-based sub-grid scale models for turbulent flows. *Engineering Applications of Artificial Intelligence*, 128:107483, 2024. doi: 10.1016/j.engappai.2023.107483.

[26] A.M. Roy, R. Bose, V. Sundararaghavan, and R. Arroyave. Deep learning-accelerated computational framework based on physics informed neural network for the solution of linear elasticity. *Neural Networks*, 162:472–489, 2023. doi: 10.1016/j.neunet.2023.03.014.

[27] A.M. Roy, S. Guha, V. Sundararaghavan, and R. Arroyave. Physics-infused deep neural network for solution of non-associative drucker–prager elastoplastic constitutive model. *Journal of the Mechanics and Physics of Solids*, 185:105570, 2024. doi: 10.1016/j.jmps.2024.105570.

[28] A.M. Roy and S. Guha. A data-driven physics-constrained deep learning computational framework for solving von Mises plasticity. *Engineering Applications of Artificial Intelligence*, 122:106049, 2023. doi: 10.1016/j.engappai.2023.106049.

[29] J.H. Bastek and D. Kochmann. Physics-informed neural networks for shell structures. *European Journal of Mechanics - A/Solids*, 97:104849, 2023. doi: 10.1016/j.euromechsol.2022.104849.

[30] Y. Yegorov and O. Kucherenko. Cylindrical shells in a wind flow: regression analysis of the wind pressure distribution coefficient. *Strength of Materials and Theory of Structures*, 114:165–172, 2025. doi: 10.32347/2410-2547.2025.114.165-172.

[31] Y. Yegorov and O. Kucherenko. Computer modelling of thin-walled shell structures with geometric imperfections. *Strength of Materials and Theory of Structures*, 111:205–213, 2023. doi: 10.32347/2410-2547.2023.111.205-213.

[32] S. Timoshenko and S. Woinowsky-Kreiger. *Theory of plates and shells.* McGraw-Hill, 1959.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 8026–8037, 2019. doi: 10.48550/arXiv.1912.01703.

[34] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S.G. Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021. doi: 10.1137/21M1397908.

[35] Z.H. Zhou. *Ensemble Methods. Foundations and Algorithms*. Chapman & Hall, 2025.

[36] E. Haghighat, A.C. Bekar, E. Madenci, and R. Juanes. A nonlocal physics-informed deep learning framework using the peridynamic differential operator. *Computer Methods in Applied Mechanics and Engineering*, 385:114012, 2021. doi: 10.1016/j.cma.2021.114012.

[37] N. Sukumar and A. Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022. doi: 10.1016/j.cma.2021.114333.