Received: 1 July 2023 | Revised: 30 August 2023 | Accepted: 25 September 2023

DOI: 10.1002/pamm.202300222

RESEARCH ARTICLE

Effects of variational formulations on physics-informed neural network performance in solid mechanics

Nils Radin | **Sven Klinkel** | **Okyay Altay**

Chair of Structural Analysis and Dynamics, RWTH Aachen University, Aachen, Germany

Correspondence
Nils Radin, Chair of Structural Analysis and Dynamics, RWTH Aachen University, Mies-van-der-Rohe-Str.1, 52074 Aachen, Germany.
Email: radin@lbb.rwth-aachen.de

Abstract

In recent years, *Physics-Informed Neural Networks* (PINNs) have emerged as a promising method for solving partial differential equations (PDEs) in a variety of fields. To this end, an artificial neural network is used to approximate the unknown variables while the differential equation is embedded in the loss function to penalize the network's output. The present contribution focuses on the application of PINNs and its derivatives in the field of solid mechanics. We formulate the PINN framework by incorporating the strong form of the corresponding PDEs alongside initial and boundary conditions into the loss function of the neural network. The loss function poses as a residual, effectively constructing a minimization problem to solve the PDEs. The nature of the residual is further reformulated into the variations (weak) form by applying test functions and integration by parts. This extension leads to *Variational Physics-Informed Neural Networks* (VPINNs), which impose a lower stringency on the solution. We demonstrate the performance of PINNs and VPINNs on a numerical solid mechanics problem.

1 | INTRODUCTION

Solving partial differential equations (PDEs) that describe physical phenomena is an enduring topic in various fields, including solid mechanics. Besides the established methods, for example, finite element analysis, we may also use artificial neural networks to approximate the solution of the PDEs. Using neural networks over traditional methods can have several advantages, for example in ill-posed problems, inverse modelling, that is discovering system parameter from data, and problems with high dimensionality. Classical methods still have limitations in these areas [1, 2]. Traditionally, the NNs require a large amount of expensive training data, which might not always be available.

An alternative machine learning approach are *Physics-Informed Neural Networks* (PINNs). They incorporate the governing equations into the loss function of the network, thus eliminating the need of data to learn the underlying physics [3]. PINNs have been used to model both forward and inverse problems in a variety of fields, such as fluid mechanics [4, 5], high-dimensional stochastic PDEs [6], optics [7] and solid mechanics [8]. PINNs typically incorporate the strong form of the PDEs to penalize the networks output.

A different approach are so called *Variational Physics-Informed Neural Networks* (VPINNs), which employ the variational (weak) formulation of the PDEs in the loss function. VPINNs can be compared to a *Petrov-Galerkin* method where

the neural network poses as the trial space while the test space is introduced as a suitable set of test functions [9]. The variational form is obtained by multiplying the strong form with the test function and integrating-by-parts. This framework has several advantages over PINNs. The integration-by-parts reduces the order of the differential operator, thus requiring a lower regularity of the solution [10]. Moreover, VPINNs can utilize domain decomposition and a varying number and order of test functions, increasing the flexibility of the method [11].

In this work, both the PINN and VPINN frameworks are developed for two-dimensional plane-strain solid mechanics problems. In Section 2, the strong loss function for PINNs is formulated to include the PDE, the constitutive equations, the kinematics and the Dirichlet and Neumann boundary conditions. Following that, the weak loss function is derived step-by-step, incorporating a suitable set of polynomial test functions and a numerical quadrature scheme. Section 3 evaluates PINN and VPINN on a benchmark. The results of both frameworks are presented. Section 4 closes this contribution with a conclusion.

2 | METHOD

In this section, the PINN framework is first briefly introduced for the general case of solving PDEs, before the method is applied to solving two-dimensional linear elasticity problems. The framework is then reformulated to VPINNs by introducing test functions and integration by parts.

2.1 | PINNs

The following nonhomogeneous linear PDE is considered

$$\mathcal{D}(u) = f(x), \quad x \in \Omega, \tag{1}$$

with the belonging Dirichlet and Neumann boundary conditions

$$u(x) = \mathcal{B}_D(x), \quad x \in \Gamma_D, \tag{2}$$

$$\mathbf{n} \cdot \nabla u(x) = \mathcal{B}_N(x), \quad x \in \Gamma_N. \tag{3}$$

Herein, $\mathcal{D}(\cdot)$ denotes the differential operator; u is the unknown field variable; $f(x)$ is a known function; Ω is the domain; Γ_D and Γ_N are the Dirichlet and Neumann boundaries; $\mathbf{n}$ is the outward facing unit normal vector; and $\mathcal{B}_D$ and $\mathcal{B}_N$ are the prescribed boundary values. An artificial neural network $\mathcal{N}$, parameterized by its weights $\mathbf{W}$ and biases $\mathbf{b}$, is introduced to approximate the solution of the differential equation:

$$\tilde{u}(x) = \mathcal{N}(x \mid \mathbf{W}, \mathbf{b}). \tag{4}$$

In order to train the network, that is adjust its weights and biases in a way such that $\tilde{u}(x) \approx u(x)$, a loss function has to be formulated. Therefore, a set of collocations points $S_{CP}\{x \mid x \in \Omega\}$, randomly generated over the domain, is introduced and used as input to the neural network in the training process. The output $\tilde{u}(x)$ of Equation (4) is used to form a residual, which can be minimized by utilizing gradient descent algorithms. In order to ensure a well-posed problem, the loss function additionally has to consider the boundary conditions. To that end, two additional sets of collocations points are introduced on the Dirichlet and Neumann boundaries as $S_{DCP}\{x \mid x \in \Gamma_D\}$ and $S_{NCP}\{x \mid x \in \Gamma_N\}$, respectively. The loss functions, in its general form, can then be formulated to

$$\mathcal{L} = |\mathcal{D}(\tilde{u}(x_i)) - f(x_i)|_{x_i} + \left|\tilde{u}\left(x_j\right) - \mathcal{B}_D\left(x_j\right)\right|_{x_j} + |\mathbf{n} \cdot \nabla \tilde{u}(x_k) - \mathcal{B}_N(x_k)|_{x_k}, \tag{5}$$

where $|\cdot|$ denotes the *mean squared error* (MSE) over $x_i \in S_{CP}$, $x_j \in S_{DCP}$ and $x_k \in S_{NCP}$, respectively [3].

For the application of this PINN framework to solve solid mechanics, the two-dimensional elastostatic equation alongside a constitutive equation and kinematics are introduced as

$$\nabla \boldsymbol{\sigma} + \mathbf{f} = \mathbf{0}, \tag{6}$$

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\epsilon} + \lambda tr\boldsymbol{\epsilon}\mathbf{I}, \tag{7}$$

$$\boldsymbol{\epsilon} = \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^{\top}\right), \tag{8}$$

where $\boldsymbol{\sigma}$ denotes the stress tensor; $\mathbf{f}$ is the body force; $\boldsymbol{\epsilon}$ is the strain tensor; and μ and λ are the Lamé parameters. The utilized method is shown in Figure 1. The input to the network are the spatial coordinates on the domain, x and y. During training, these coordinates are the collocation points, a finite set of coordinates randomly generated using the *Latin Hypercube Sampling* (LHS). Once the network is trained, the input can be any arbitrary point on the domain, meaning the network not only learns the solution at discrete points on the domain, but a continuous function. The solution to the PDE is $\mathbf{u} = (u, v)$. The network output is extended to predict the three stress components σ_{xx}, σ_{yy} and σ_{xy} besides the displacement components u and v. This mixed-variable formulation eliminates the need of a second order derivative of $\mathbf{u}$, since the partial derivatives of $\boldsymbol{\sigma}$ can be used in the loss function instead. Otherwise, the partial derivatives of $\mathbf{u}$ would be needed in Equation (8) to calculate $\boldsymbol{\epsilon}$, which is used to calculate $\boldsymbol{\sigma}$ via Equation (7) and then derived a second time. Additionally, using the actual prediction of $\boldsymbol{\sigma}$ and a calculation via the derivative of $\mathbf{u}$, a coupling between the first two and the last three outputs of the network can be achieved. It has been shown that the mixed-variable formulation improves the prediction accuracy and trainability of the PINN [12].

The loss function of the network consists of four individual terms, where the influence of each term can be controlled using weighting factors τ. The values of τ have a great influence on the convergence behaviour of the loss and the prediction accuracy of the network. So far, there is no straightforward way of calculating them beforehand [12]. Usually, they are determined by trial-and-error for each individual problem in practice. The loss functions for the PINN reads

$$\mathcal{L}_{PINN} = \mathcal{L}_{eq} + \tau_c \cdot \mathcal{L}_c + \tau_{bc} \cdot \mathcal{L}_{bc} + \tau_{data} \cdot \mathcal{L}_{data}, \tag{9}$$

where $\mathcal{L}_{eq}$ is the loss attributed to the PDE itself; $\mathcal{L}_c$ is the loss value of the constitutive equation; $\mathcal{L}_{bc}$ is the loss value of the boundary conditions. Furthermore, supplementary training data can be considered with $\mathcal{L}_{data}$. This data is composed of input-target-pairs and improves trainability and accuracy of the solution. The four individual loss terms are defined as

$$\mathcal{L}_{eq} = \left|\sigma_{xx,x} + \sigma_{xy,y} + f_x\right|_{\Omega} + \left|\sigma_{xy,x} + \sigma_{yy,y} + f_y\right|_{\Omega}, \tag{10}$$

$$\mathcal{L}_c = \left|(\lambda + 2\mu)\epsilon_{xx} + \lambda\epsilon_{yy} - \sigma_{xx}\right|_{\Omega} + \left|(\lambda + 2\mu)\epsilon_{yy} + \lambda\epsilon_{xx} - \sigma_{yy}\right|_{\Omega} + \left|2\mu\epsilon_{xy} - \sigma_{xy}\right|_{\Omega}, \tag{11}$$

$$\mathcal{L}_{bc} = |\mathbf{u} - \mathcal{B}_D|_{\Gamma_D} + |\mathbf{n}\cdot\boldsymbol{\sigma} - \mathcal{B}_N|_{\Gamma_N}, \tag{12}$$

$$\mathcal{L}_{data} = |u - u^*| + |v - v^*| + \left|\sigma_{xx} - \sigma^*_{xx}\right| + \left|\sigma_{yy} - \sigma^*_{yy}\right| + \left|\sigma_{xy} - \sigma^*_{xy}\right|. \tag{13}$$

Herein, $|\cdot|$ denotes the MSE; $\mathbf{u}$ and $\boldsymbol{\sigma}$ are the direct outputs of the network; the partial derivatives of σ_{ij} are obtained via *automatic differentiation* [13]; the values marked with an * are given targets; and the strain components ϵ_{ij} are calculated with Equation (8). The loss function is minimized using a combination of two gradient descent optimization algorithms, namely *ADAM* and *L-BFGS-B*.

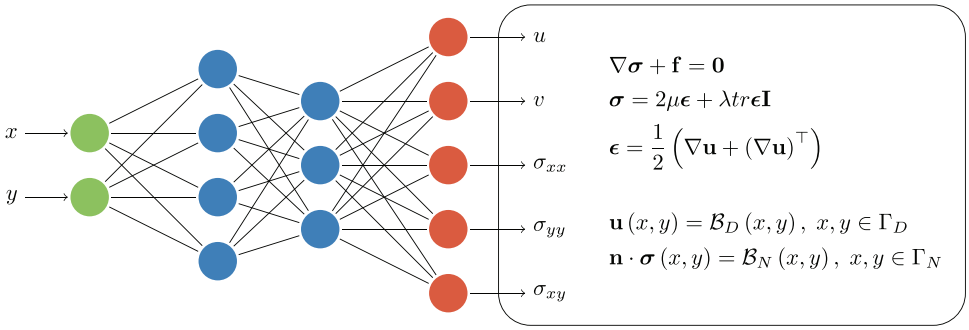


FIGURE 1 For the solution of the two-dimensional elastostatic problems adopted PINN architecture. PINN, Physics-Informed Neural Network.

2.2 | VPINNs

The basis of VPINNs is the variational or weak form of the PDE. To derive the weak form, the PDE introduced in Equation (6) is multiplied with a test function and integrated over the whole domain

$$\int_{\Omega} (-\nabla \cdot \boldsymbol{\sigma})^{\top} \cdot \boldsymbol{\nu} \, \mathrm{d}\Omega = \int_{\Omega} \mathbf{f}^{\top} \cdot \boldsymbol{\nu} \, \mathrm{d}\Omega, \tag{14}$$

where $\boldsymbol{\nu}$ denotes a test function, such that $\boldsymbol{\nu} = 0$ on Γ_D. The test function has to be a vector-valued function of the same type as the primary unknown of the PDE. Equation (14) can further be reformulated by integrating-by-parts and applying the divergence theorem to arrive at the following preliminary form

$$\int_{\Omega} \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\epsilon}(\boldsymbol{\nu}) \, \mathrm{d}\Omega = \int_{\Omega} \mathbf{f}^{\top} \cdot \boldsymbol{\nu} \, \mathrm{d}\Omega + \int_{\Gamma_N} \mathbf{t}^{\top} \cdot \boldsymbol{\nu} \, \mathrm{d}S, \tag{15}$$

where $\mathbf{t}$ denotes the traction vector. By integrating-by-parts, the order of the differential operator on the primary unknown, the displacements, is reduced by one, while the spatial derivative is placed on the test function instead. This reduction lowers the imposed stringency on the solution. Additionally, a less complex formulation is obtained since only one spatial derivative of $\mathbf{u}$ is required [11].

The domain Ω is decomposed into elements Ω^e with $e = 1, 2, \dots, E$. A suitable set of test functions has to be introduced. In this work, localized, non-overlapping test functions are employed

$$\boldsymbol{\nu}_n^e = \begin{cases} \boldsymbol{\nu}_n \neq 0 & \text{on } \Omega^e \\ 0 & \text{elsewhere.} \end{cases} \tag{16}$$

Since the vector-valued test functions have to be of the same type as the primary unknown $\mathbf{u} = (u, v)$, they are introduced as $\boldsymbol{\nu}_n = (\nu_n^u, \nu_n^v)$. The same test functions are used for u and v in this work, therefore the superscript is omitted. The element-wise test functions are defined as a combination of *Legendre polynomials* P_n as follows

$$\nu_n(x, y) = (P_{n+1}(x) - P_{n-1}(x)) \cdot (P_{n+1}(y) - P_{n-1}(y)), \quad n = 1, 2, \dots, N, \tag{17}$$

where n is the order of the Legendre polynomial and N is the total number of test functions. By combining the polynomials of order $(n + 1)$ with $(n - 1)$, the test functions always vanish at the boundaries of the integration domain, fulfilling the requirement set above, that is, $\boldsymbol{\nu} = 0$ on Γ_D. The first three test functions are depicted in Figure 2.

In order to implement Equation (15) into a loss function, the integration has to be addressed. In this work, the *Gauss-Lobatto quadrature* was employed as a numerical integration scheme. It utilizes the endpoints of the integration interval as quadrature points, thus increasing the accuracy for functions with possible singularities near boundaries. The number of quadrature points in x and y direction is set to be equal, $P = Q$. The belonging integration weights are denoted as w_p and w_q. Finally, the loss function for VPINNs can be formulated, equivalent to Equation (9), as

$$\mathcal{L}_{VPINN} = \mathcal{L}_{eq}^{\mathcal{V}} + \tau_c \cdot \mathcal{L}_c + \tau_{bc} \cdot \mathcal{L}_{bc} + \tau_{data} \cdot \mathcal{L}_{data}, \tag{18}$$

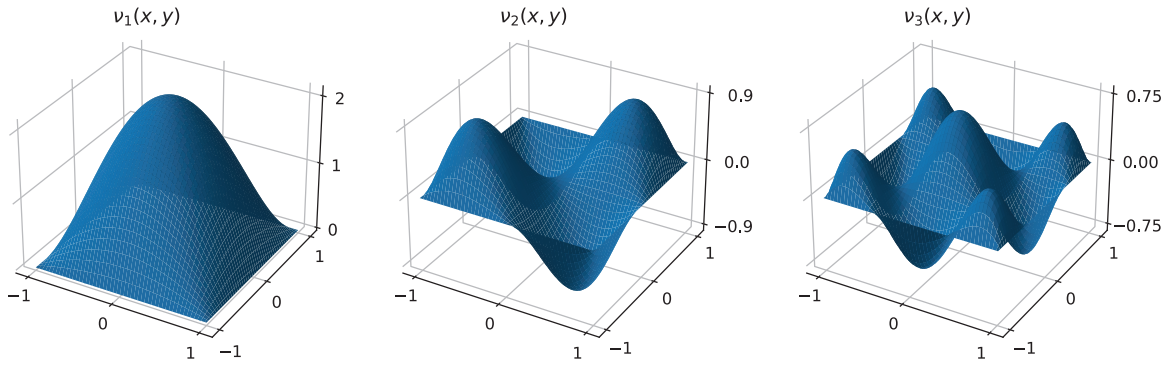


FIGURE 2 The first three test functions generated by combinations of Legendre polynomials.

where $\mathcal{L}_c$, $\mathcal{L}_{bc}$ and $\mathcal{L}_{data}$ are defined as in Equations (11)–(13); and the loss of the variational form is defined as the MSE of a residual $\mathcal{R}_n^e$, summed up over the total number of elements E and test functions N

$$\mathcal{L}_{eq}^{\mathcal{V}} = \sum_{e=1}^{E} \frac{1}{N} \sum_{n=1}^{N} \left| \mathcal{R}_n^e \right|. \tag{19}$$

The residual $\mathcal{R}_n^e$ results from Equation (15), which can be reformulated by introducing the test functions and the numerical integration scheme. Since the defined test functions are zero on all boundaries, the last term, the traction integral, vanishes. To ensure that the Neumann boundary conditions are fulfilled, they are enforced as in the PINN scheme with Equation (12).

$$\mathcal{R}_n^e = \int_{\Omega^e} \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\epsilon}(\boldsymbol{\nu}) \, \mathrm{d}\Omega^e - \int_{\Omega^e} \mathbf{f}^\top \cdot \boldsymbol{\nu} \, \mathrm{d}\Omega^e - \int_{\Gamma_N^e} \mathbf{t}^\top \cdot \boldsymbol{\nu} \, \mathrm{d}S^e \tag{20}$$

$$= \sum_{p=1}^{P} \sum_{q=1}^{Q} w_p w_q \left(\sigma_{xx} \nu_{x,x} + \sigma_{xy} \left(\nu_{x,y} + \nu_{y,x} \right) + \sigma_{yy} \nu_{y,y} - \left(f_x \nu_x + f_y \nu_y \right) \right). \tag{21}$$

Increasing the number of test functions N simultaneously increases the highest occurring order of the polynomial. This can be viewed as a *p-refinement*. Similarly, increasing the number of elements on the domain is equivalent to a *h-refinement*. The flexibility is further increased by the ability to specify the nature and number of test functions for each element [11]. Similar to the PINN scheme, the network is trained using both the ADAM and the L-BFGS-B optimizers.

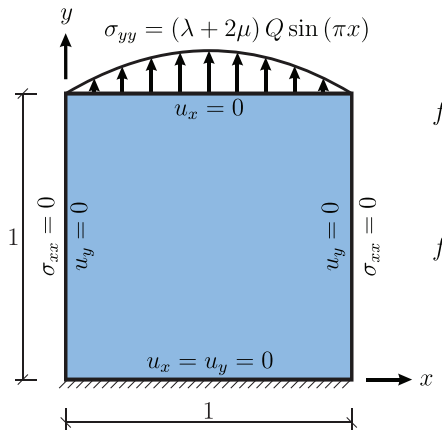
3 | RESULTS

In order to demonstrate the performance of the PINN and VPINN frameworks, a benchmark is utilized. The elastic plane-strain problem is given in Figure 3, with boundary conditions and body force in x and y direction. The belonging Lamé parameters are set to $\lambda = 1$ and $\mu = 0.5$ while the load factor is set to $Q = 4$. The analytical reference solution for the displacements u and v is given in [8] as

$$u_{ref}(x, y) = \cos(2\pi x) \sin(\pi y), \quad v_{ref}(x, y) = \sin(\pi x) Q y^4 / 4. \tag{22}$$

The neural networks itself are set up equally for both frameworks, with an architecture of 3 hidden layers with 20 neurons each, and the input and output layers with 2 and 5 neurons, respectively. The weighting factors for the individual loss terms are set to $\tau_c = 1$ and $\tau_{bc} = 10$, which has been found to produce adequate approximations. Since no input-target pairs are used in training, τ_{data} is set to 0. An overview of the benchmark parameter can be found in Table 1.

The two frameworks both take a set of spatial coordinates as input in training. In the case of PINNs, the inputs are the randomly sampled collocation points, whereas in VPINNs, quadrature points are utilized instead. In both cases, the



$$f_x = \lambda \left[4\pi^2 \cos(2\pi x) \sin(\pi y) - \pi \cos(\pi x) Q y^3 \right] + \mu \left[9\pi^2 \cos(2\pi x) \sin(\pi y) - \pi \cos(\pi x) Q y^3 \right]$$

$$f_y = \lambda \left[-3 \sin(\pi x) Q y^2 + 2\pi^2 \sin(2\pi x) \cos(\pi y) \right] + \mu \left[-6 \sin(\pi x) Q y^2 + 2\pi^2 \sin(2\pi x) \cos(\pi y) + \pi^2 \sin(\pi x) Q y^4 / 4 \right]$$

FIGURE 3 Benchmark with its physical domain and the boundary conditions (left) as well as the body forces (right) [8].

TABLE 1 Parameters of PINN and VPINN frameworks.

General	
Architecture	$[2-20-20-20-5]$
Activation functions	$tanh$
Iterations with ADAM	500
Iterations with L-BFGS-B	4500
Weighting factors	$\tau_c = 1, \tau_{bc} = 10, \tau_{data} = 0$
Boundary collocation points	4×50
PINN	
Domain collocation points	1600
VPINN	
Elements	$E = 10 \times 10$
Quadrature points	$Q = P = 4$
Test functions	$N = 3$

Abbreviations: PINNs, Physics-Informed Neural Networks; VPINNs, Variational Physics-Informed Neural Networks.

same collocation points on the boundaries are used to enforce the Dirichlet and Neumann boundary conditions, that is, 50 collocation points on each boundary. The number of domain collocation points in PINNs is set to 1600. The total number of quadrature points that pose as input to the VPINN is calculated by multiplying the number of elements with the number of quadrature points per element. With the in Table 1 given parameters, a total number of 1600 quadrature points results.

The following results are averaged over 10 random network initializations, where the best and worst predictions are ignored. Figure 4 shows the point-wise absolute error of the predictions of the two frameworks for u and v, compared to the

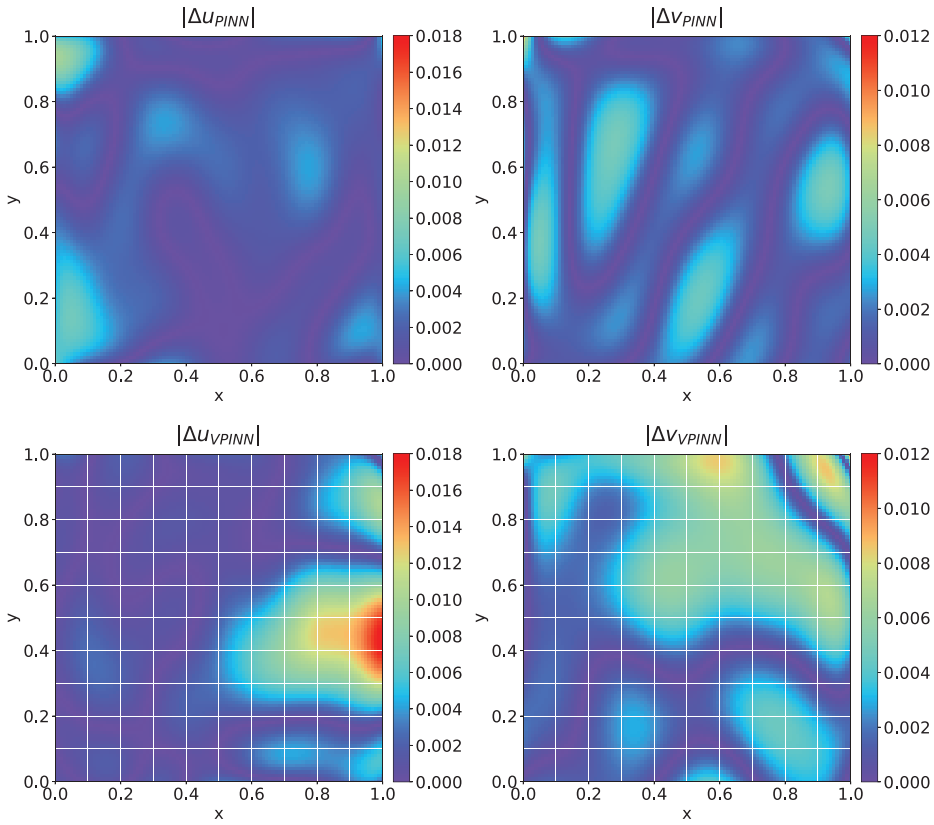


FIGURE 4 Pointwise absolute error of the predictions of PINN (top) and VPINN (bottom) for displacements u and v w.r.t. the analytical solution. Both neural network approaches are in the same accuracy range for the studied benchmark problem. PINNs, Physics-Informed Neural Networks; VPINNs, Variational Physics-Informed Neural Networks.

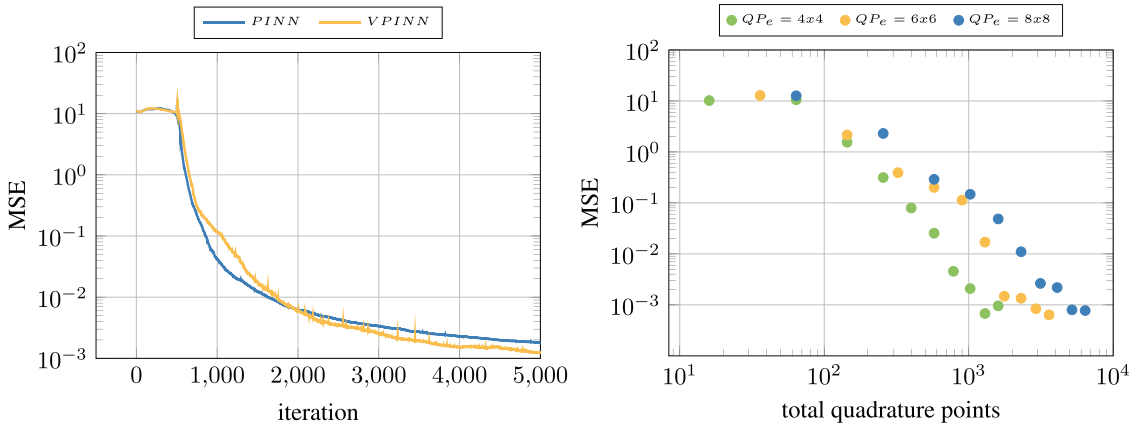


FIGURE 5 MSE of PINN and VPINN over 5000 training iterations (left). PINN and VPINN show similar convergence behaviour. Influence of the total number of quadrature points and element-wise quadrature points (QP_e) on the MSE after 5000 iterations (right). Prediction accuracy increases with increasing number of quadrature points and it is more efficient to increase the number of elements rather than the quadrature points. MSE, mean squared error; PINNs, Physics-Informed Neural Networks; VPINNs, Variational Physics-Informed Neural Networks.

analytical solution, that is, $|\Delta u_{PINN}| = |u_{PINN} - u_{ref}|$. Both PINN and VPINN demonstrate a satisfactory approximation of the solution. The point-wise error is of the same order for both frameworks. This is thought to be because of the smooth nature of the solution. The similar prediction behaviour can be observed in the progress of the MSE over the 5000 training iterations as depicted on the left in Figure 5. The plot on the right demonstrates the influence of the total number of quadrature points QP_{total} on the domain on the MSE. The overall trend indicates that a higher prediction accuracy can be achieved the higher QP_{total} is. It is furthermore differentiated between three different numbers of quadrature points per element QP_e. The results demonstrate that it is more efficient to increase the number of elements E rather than the number of quadrature points QP_e.

4 | CONCLUSION

In this work, we introduced a PINN framework which incorporates the strong form of a PDE into its loss function in order to learn the underlying physical behaviour of solid mechanics problems. By converting the PDE into its weak form, VPINN framework was formulated. Both frameworks were evaluated on a benchmark system to demonstrate the performance. The results showed high accuracy predictions with both networks. Since the solution is smooth, both frameworks show similar performance. The extension to employ the weak form effectively reduced the order of the differential operator on the displacement by one. Additionally, the flexibility of the method was increased in terms of an hp-refinement by increasing the number of elements or test functions.

ACKNOWLEDGMENTS

Open access funding enabled and organized by Projekt DEAL.

ORCID

Nils Radin https://orcid.org/0009-0007-7094-601X
Sven Klinkel https://orcid.org/0000-0003-1535-0823
Okyay Altay https://orcid.org/0000-0001-8518-8011

REFERENCES

1. Karniadakis, G. E., Kevredikis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, *3*, 422–440.
2. Lenzen, N., & Altay, O. (2022). Machine learning enhanced dynamic response modelling of superelastic shape memory alloy wires. *Materials*, *15*, 304.

3. Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). 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.
4. Rao, C., Sun, H., & Liu, Y. (2020). Physics-informed deep learning for incompressible laminar flows. *Theoretical and Applied Mechanics Letters*, *10*, 207–212.
5. Cai, S., Mao, Z., Wang, Z., Yin, M., & Karniadakis, G. E. (2021). Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, *37*, 1727–1738.
6. Karumuri, S., Tripathy, R., Bilionis, I., & Panchal, J. (2020). Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks. *Journal of Computational Physics*, *404*, 109120.
7. Chen, Y., Lu, L., Karniadakis, G. E., & Dal Negro, L. (2020). Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics Express*, *28*, 11618–11633.
8. Haghighat, E., Raissi, M., Moure, A., Gomez, H., & Juanes, R. (2021). A physics-informed deep learning framework for inversion and surrogate modleing in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, *379*, 113741.
9. Kharazmi, E., Zhang, Z., & Karniadakis, G. E. (2021). hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, *374*, 113547.
10. Berrone, S., Canuto, C., & Pintore, M. (2022). Variational physics informed neural networks: The role of quadratures and test functions. *Journal of Scientific Computing*, *92*, 100.
11. Kharazmi, E., Zhang, Z., & Karniadakis, G. E. (2019). Variational physics-informed neural networks for solving partial differential equations. arXiv preprint arXiv:1912.00873.
12. Rao, C., Sun, H., & Liu, Y. (2021). Physics-informed deep learning for computational elastodynamics without labeled data. *Journal of Engineering Mechanics*, *147*, 04021043.
13. Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: A Survey. *Journal of Machine Learning Research*, *18*, 1–43.

How to cite this article: Radin, N., Klinkel, S., & Altay, O. (2023). Effects of variational formulations on physics-informed neural network performance in solid mechanics. *Proceedings in Applied Mathematics and Mechanics*, *23,* e202300222. https://doi.org/10.1002/pamm.202300222