Ensemble of physics-informed neural networks for solving plane elasticity problems with examples

Aliki D. Mouratidou[1*], Georgios A. Drosopoulos[2,3] and Georgios E. Stavroulakis[1]

[1*]Institute of Computational Mechanics and Optimization, School of Production Engineering and Management, Technical University of Crete, Kounoupidiana, Chania, 73100, Crete, Greece.

[2]Discipline of Civil Engineering, School of Engineering and Computing, University of Central Lancashire, Preston campus, Preston, PR1 2HE, United Kingdom.

[3]Discipline of Civil Engineering, School of Engineering, University of KwaZulu-Natal, Durban campus, Durban, 4041, South Africa.

*Corresponding author(s). E-mail(s): amouratidou@tuc.gr;
Contributing authors: gdrosopoulos@uclan.ac.uk; gestavroulakis@tuc.gr;

Abstract

Two-dimensional (plane) elasticity equations in solid mechanics are solved numerically with the use of an ensemble of physics-informed neural networks (PINNs). The system of equations consists of the kinematic definitions, i.e. the straindisplacement relations, the equilibrium equations connecting a stress tensor with external loading forces and the isotropic constitutive relations for stress and strain tensors. Different boundary conditions for the strain tensor and displacements are considered. The proposed computational approach is based on principles of artificial intelligence and uses a developed open source machine learning platform, scientific software Tensorflow, written in Python and Keras library, an application programming interface, intended for a deep learning. A deep learning is performed through training the physics-informed neural network (PINN) model in order to fit the plain elasticity equations and given boundary conditions at collocation points. The numerical technique is tested on an example, where the exact solution is given. Two examples with plane stress problems are calculated with the proposed multi-PINN model. The numerical solution is compared with results obtained after using commercial finite element software. The numerical results have shown that an application of a multi-network approach is more beneficial in comparison with using a single PINN with many outputs. The derived results confirmed the efficiency of the introduced methodology. The proposed technique can be extended and applied to the structures with nonlinear material properties.

Keywords: Plane elasticity equations, Partial differential equation, Physics-informed neural network model, Computational artificial intelligence, Numerical simulation algorithm.

1 Introduction

Partial differential equations (PDEs) are used for mathematical modelling in many areas of applied mathematics and mechanics, like physical, biological, financial and economics problems. The underlying laws in those problems are expressed in the form of PDEs and the related mathematical analysis is focused on the investigation of well-posedness and uniqueness of solution. For practical applications with complicated domains, boundary or initial conditions and properties of the continuous media, it is impossible to find closed-form analytical solutions. Therefore, the question of numerical approximation of the solution arises. Numerical mathematics and computational mechanics deal with this question.

Artificial neural networks (ANNs) have been developed for the solution of machine learning problems based on examples. The well-known example is the feedforward backpropagation neural network model, which is trained based on given input-output relations (data-set) in the sense of a supervised learning. These data can be created from a model, describing, e.g. mechanical phenomena or measured in the laboratory. Training is in fact an iterative process of the optimization problem to find the parameters of the neural network in order to get an approximate solution. It's response for various inputs, in a least squares sense.

Artificial neural networks have been adopted in different areas of science and engineering, where data-set for training and testing are available. In particular, ANNs are employed in elastoplastic and contact problems in mechanics by using the minimization of energy. The Hopfield and Tank neural networks have been proposed by Kortesis and Panagiotopoulos [22]. Inverse and parameter-identification problems in mechanics have been solved by using backpropagation neural networks in Stavroulakis et al. [38], [39], Stavroulakis [37], Drosopoulos and Stavroulakis [8], [9] Waszczyszyn and Ziemianski [42]. Buckling loads in nonlinear problems for elastic plates have been calculated in Muradova and Stavroulakis [31]. A recent review of classical usage of neural networks within computational mechanics has been written in Yagawa and Oishi [44].

ANNs have also been used to solve differential equations. In particular, usage of ANNs for the solution of partial and ordinary differential equations has been proposed in the similar works of Kortesis and Panagiotopoulos [22], Theocaris and Panagiotopoulos [41] and Lagaris et al. [24]. Potential energy optimization can be integrated in the neural network training and lead to the Deep Energy methods. Alternatively, the partial differential equations and initial or/and boundary conditions are approximated on collocation points by using automatic differentiation of the neural network.

After some years, the developments and the accumulated experience in the field of deep learning and the appearance of user-friendly software that allows for automatic differentiation in neural networks, led to the development of physics-informed neural networks (PINNs) for solving direct and inverse problems, e.g. by Karniadakis et al. [19], Raissi et al. [33], Tartakovsky et al. [40], Kadeethum et al. [18] and Faroughi et al. [11]. The unterlying idea behind PINNs is the minimization of the energy functional or an error functional, that is the residual of the PDE and the required initial and boundary conditions (Guo and Haghighat [14]). Recently, PINNs have become one of the major numerical techniques in solving scientific and engineering problems involving ordinary or partial

differential equations. It is one of the most promising research directions in computational artificial intelligence which can model and solve direct and inverse problems in mechanics within a unified framework, thanks to automatic differentiation and modern open-source scientific software. In addition, PINNs can be combined with classical numerical methods, e.g., spectral elements, achieving both high accuracy of computations and flexibility.

PINNs have been applied in a wide range of scientific computing applications, e.g. in direct and inverse problems, Karniadakis [19], Raissi et al. [33], Baydin et al. [2], Meade and Fernadez [28], Shin et al. [36], in solid mechanics, Haghighat et al. [15], [16], Muradova and Stavroulakis [29], [30], Katsikis et al. [20], in computational fluid dynamics (CFD), Cai et al. [4], Haghighat et al [15] and in electromagnetics, Chen et al. [6].

In [32], a framework of finite-strain elasto-plasticity is proposed using PINNs, considering rate-independent isotropic hardening. A mixed formulation approach involving PINNs is proposed in [34], requiring up to first-order derivatives to construct the physical loss functions. A PINNs framework is suggested in [10], presenting a mixed formulation that can be used to find a solution for a non-uniform beam resting on an elastic foundation, subjected to arbitrary external loading. In [17] a mixed formulation is presented using PINNs, aiming to solve multi-physical problems, emphasizing in a stationary thermo-mechanically coupled system of equations. A PINN formulation is provided in [3], analyzing the nonlinear buckling behaviour of a three-dimensional functionally graded porous, slender beam, which rests on a Winkler-Pasternak foundation. In [5] the accuracy of the deep energy method, a type of PINN adopting the principle of minimum potential energy to predict the mechanical response, is investigated using a random Fourier feature mapping and other techniques.

Another recent direction in the field is the application of neural operators, where the papers of Lu et al. [27] adopting the DeepONet neural operator, as well as Li et al. [25] and Kovachki [23], using a Fourier Neural Operator (FNO) can be mentioned. These techniques are more general, in the sense that they learn how to approximate differential equations and then can solve any new problem, with different initial and/or boundary conditions and physical parameters. The DeepONet has a NN for encoding the discrete input function space and a NN for encoding the domain of the output functions and it is based on the universal approximation theorem. In the FNO approach the integral kernel is parameterized in the Fourier space.

In this work a computational intelligence scheme, based on an ensemble of PINNs, is applied to the problem of two-dimensional elasticity. The proposed computational approach is based on principles of artificial intelligence and uses Tensorflow, an open source machine learning scientific software, and Keras library, an application programming interface intended for a deep learning, both written in Python. A deep learning is performed through training the PINN model in order to fit the plain elasticity equations and associated boundary conditions at collocation points. The numerical technique is tested on an example, where the exact solution is given.

Here, in contrast with the scientific computing methodology using artificial neural networks [16], eight unknowns are considered in the plain elasticity equations with given boundary conditions and external loading forces. The equations from the theory of elasticity are simulated and solved with the use of ensemble of PINNs, formulating a

multi-PINN model. This model combines and connects two types of neural networks: a surrogate and a residual neural network. The proposed architecture of the multiPINN provides eight surrogate networks for the unknowns and one residual network for training.

It is noted that an approximate differentiation is an ill-posed problem in the sense that if a function is approximated with some error, then the error of calculating the approximate derivatives of this function can be quite big. For instance, the calculation of the strains and stresses using the derivative of displacements involves an error, since within traditional finite element analysis the displacement field is approximately determined and calculating its derivative may increase the error. To overcome this issue, one neural network is considered in this article for each unknown in the elasticity equations. Thus, eight surrogate networks are intended for the unknown functions, namely, the components of the strain and the stress tensors as well as the unknown displacement field, respectively. The residual network provides the residuals of the partial differential equations (PDEs) and of the boundary conditions. The proposed scheme suggests, therefore, a novel PINNs architecture aiming to overcome the mentioned accuracy issue. This summarizes the innovation of this work, since to the authors best knowledge limited relevant literature can be found.

The numerical technique is tested on an example where the exact solution, i.e. displacement components in x and y directions and external forces are known. The predicted stress and strain tensor are computed by the PINN model. An investigation is also conducted that includes the loss function, as well as the choice of the number of hidden layers, neurons, training iterations, training samples (collocation points), and batch sizes. To verify the proposed scheme, the displacements in the predicted by the PINN model are compared with the exact solution at discrete points. Two examples are also considered, where the numerical solution is compared with results, obtained after using commercial finite element software ABAQUS. The sensitivity of the PINN's performance with respect to the hyperparameters, i.e., the number of layers, neurons, training samples (collocation points) and the required time of computations has been investigated.

The article is organized as follows. In Section 2 the plain elasticity equations are described. The architecture of the proposed multi-PINN model is presented in Section 3. Here, the eight surrogate networks and the one residual network are constructed. In Section 4 an introduction of the computational algorithm is provided, including the feedforward neural network and backpropagation. The description of the algorithm in Python programming environment with using key commands and methods of the scientific software Tensorflow and Keras library is given. Section 5 presents numerical examples, where the proposed PINN is applied on structural problems. Discussions and conclusions are given in Section 6.

2 Elasticity equations

The elasticity basic equations consist of equilibrium equations of stresses, constitutive equations that relate stresses and strains (Hooke's law) and strain-displacement

relations. In two-dimensional case the partial differential equations for stresses with external body forces read,

$$\sigma_{ij,j} + f_i = 0, \tag{1}$$

where $\sigma_{11} = \sigma_{xx}$, $\sigma_{22} = \sigma_{yy}$ and $\sigma_{12} = \sigma_{xy}$ are the components of the stress tensor and the their partial derivatives $\sigma_{ij,j}$. The functions $f_1 = f_x$, $f_2 = f_y$ are external forces. The constitutive equations in isotropic linear case are

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

where $\varepsilon_{11} = \varepsilon_{xx}$, $\varepsilon_{22} = \varepsilon_{yy}$ and $\varepsilon_{12} = \varepsilon_{xy} = \varepsilon_{21} = \varepsilon_{yx}$ are the components of the strain tensor and λ and μ are the Lam´e parameters.

The strain-displacement linear relations are written as

$$\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right), \tag{3}$$

where $u_1 = u_x$ and $u_2 = u_y$ are the displacements in x and y directions, respectively. In the formulas (1)-(3), x,y are the Cartesian coordinates and $(x,y) \in \Omega$, where Ω is the domain of definition for the stresses, strains and displacements.

When a nonlinear material law is true, then equations (2) are replaced by nonlinear constitutive relations (e.g. the Ogden polynomial models).

3 Architecture of the multi-physics informed neural network model

As it is known, artificial neural networks (ANNs) have been created on the principles of biological neural networks. A neural network consists of input, hidden and output layers. Each layer provides neurons, which are connected with the neurons from the previous and next layers. The procedure which supplies the network with an information spreading from the input layer to the outputs, going through all the hidden layers with neurons, is called feedforward. For a network with two input variables, the neurons in the layers within the feedforward process, are connected with the formulas

$$\mathbf{z}_0 = (\mathbf{x}, \mathbf{y}) \tag{4}$$

$$\mathbf{z_k} = S(\mathbf{W}_k^T \cdot \mathbf{z}_{k-1} + \mathbf{b}_k), \quad k = 1, 2, ..., N_l \tag{5}$$

where the pair $(\mathbf{x},\mathbf{y})$ represents the input of the NN (neural network), $\mathbf{W}_k$ is the matrix of dimensions $(n_{(k-1)} \times n_k)$ containing the weights, $\mathbf{b_k}$ is the bias vector between $k-1$ and k layers of the neural network, and $N_l - 1$ is the number of the hidden layers.

The training of a network is perhaps the most important part of machine learning since through this process a loss function (predicting the error of the neural network) is minimized and the weights and biases are updated. For minimizing the loss function, it is necessary to calculate the derivatives of the function with respect to its variables, weights

and biases. The backpropagation algorithm calculates all the necessary derivatives using the rule of chain differentiation, starting from the last layer and moving backwards to the input layer of the neural network.

Main optimization algorithms used in ANN training, are the Gradient Decent Algorithm ([35]), the L-BFGS (Limited-Memory Broyden-Fletcher-Goldfarb-Shanno), ([26], [12], [43]) and the Adam (Adaptive Moment Estimation) algorithm ([21]). The Gradient Decent Algorithm minimizes the loss error function with respect to the weights and the biases of the network which change in the opposite direction of the feedforward process. The partial derivatives of the function with respect to each parameter are calculated. The L-BFGS algorithm belongs to the family of quasi-Newton methods developed by Broyden, Fletcher, Goldfard and Shanno. Newton's method for minimization problems requires the calculation of the Hessian matrix of the objective function, i.e. it requires the calculation of second derivatives with respect to all the parameters, which is computationally prohibitive in deep learning problems. In contrast, the quasi-Newton methods approach find zeroes or local maxima and minima of functions.

PINNs are neural networks that allow solving ordinary and partial differential equations in a specific domain area. The PINNs output is linearly dependent on the last hidden layer. The output layer does not enter the activation function as usually happens in ANNs. Therefore, instead of (4), (5) it is obtained

$$\mathbf{z}_0 = (\mathbf{x}, \mathbf{y}) \tag{6}$$

$$\mathbf{z}_k = S(\mathbf{W}_k{}^T \cdot \mathbf{z}_{k-1} + \mathbf{b}_k),\ k = 1,2,\ldots,N_l - 1, \tag{7}$$

$$\mathbf{y} = \mathbf{z}_{N_l} = \mathbf{W}_{N_l}^T \cdot \mathbf{z}_{N_l-1} + \mathbf{b}_{N_l}. \tag{8}$$

Here, for the investigated elasticity problem (1)-(3) the proposed ensemble-PINN model combines and connects two types of neural networks: eight surrogate networks and one residual network. The surrogate neural networks are intended for computing the components of the stress tensor, σ_{ij} ($\sigma_{ij} = \sigma_{ji}$), the components of the strain tensor, ε_{ij} ($\varepsilon_{ij} = \varepsilon_{ji}$) and the displacement field, u_x and u_y, respectively. These surrogate networks take as input the coordinates of collocation points, where the equilibrium equations (1), stress-strain constitutive law (2), strain-displacement relations (3) and given boundary conditions are calculated. During the training process, each surrogate neural network constructs an approximate solution, based on the activation functions, weights and biases of the feedforward algorithm (6), (7) and (8). The neural quantities (expansions) σ_{ij}^{NN}, ε_{ij}^{NN}, u_x^{NN} and u_y^{NN} of the unknown functions σ_{ij}, ε_{ij}, u_x and u_y, respectively of the system (1)-(3) which satisfy the associated boundary conditions, are the corresponding outputs of the surrogate models. The architecture of the surrogate models, depicting the output components of the stress, the strain and the displacements, are shown in Figures 1, 2 and 3, respectively.

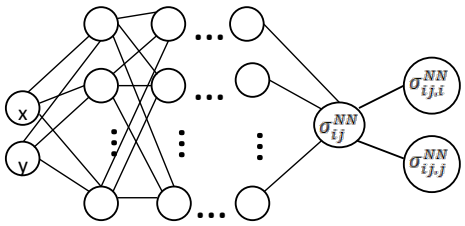


Fig. 1: Architecture of the three surrogate neural models σ_{ij}^{NN}, $i,j = 1,2$ (σ_{xx}, σ_{yy} and $\sigma_{xy} = \sigma_{yx}$) with the partial derivatives $\sigma_{ij,j}^{NN}$ computed by the automatic differentiation in the multi-PINN.

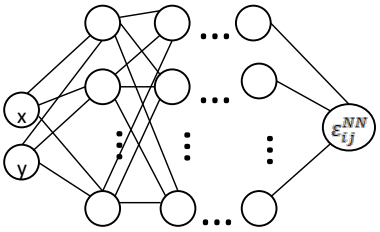


Fig. 2: Architecture of the three surrogate neural models ε^{NN}_{ij}, $i,j = 1,2$ (ε_{xx}, ε_{yy} and ε_{xy}) in the multi-PINN.

The residual network takes the input (the collocation points for the system and for the boundary conditions) and the output from each surrogate network for training the multi-PINN. The architecture of the residual model is illustrated in Figure 4. The residual expressions are given below

$$E_1 = \sigma_{xx,xNN} + \sigma_{xy,yNN} + f_x,$$

$$E_2 = \sigma_{xy,xNN} + \sigma_{yy,yNN} + f_y,$$

$$E_3 = \sigma_{xxNN} - \lambda(\varepsilon_{NNxx} + \varepsilon_{NNyy}) - 2\mu\varepsilon_{NNxx},$$

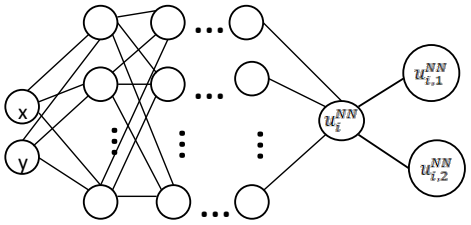


Fig. 3: Architecture of the two surrogate neural models u_1^{NN} and u_2^{NN} (u_x and u_y) with the partial derivatives $u_{i,j}^{NN}$ computed by the automatic differentiation in the multi-PINN.

$$
\begin{aligned}
E_4 &= \sigma_{yy}^{NN} - \lambda(\varepsilon_{xx}^{NN} + \varepsilon_{yy}^{NN}) - 2\mu\varepsilon_{yy}^{NN} \\
E_5 &= \sigma_{xy}^{NN} - 2\mu\varepsilon_{xy}^{NN}, \\
E_6 &= \varepsilon_{xx}^{NN} - u_{x,x}^{NN}, \\
E_7 &= \varepsilon_{yy}^{NN} - u_{y,y}^{NN}, \\
E_8 &= \varepsilon_{xy}^{NN} - \frac{1}{2}(u_{x,y}^{NN} + u_{y,x}^{NN}).
\end{aligned}
,
$$

The output of the residual network is the loss error function, which is calculated as the mean square error (MSE),

$$MSE = MSE_e + MSE_b, \tag{9}$$

where MSE_e is the loss error function for the system (1)-(3) and MSE_b is the loss error for the associated boundary conditions. The residual MSE_e is computed as

$$MSE_e = \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \sum_{k=1}^{8} (E_k(x_i, y_j))^2 ,$$

where (x_i, y_j) are the interior collocation points on the mesh N_1xN_2. For the boundary conditions, in case of the rectangular domain $(0,l_1)\times(0,l_2)$, where l_1, l_2 are the lengths of the sides of the rectangular area, is considered as the investigated structure, the residual MSE_b is written as

$$
\begin{aligned}
MSE_b = & \frac{1}{N_2} \sum_{i=1}^{N_2} \left[(E_b(0, y_i))^2 + E_b(l_1, y_i))^2 \right] \\
& + \frac{1}{N_1} \sum_{i=1}^{N_1} \left[(E_b(x_i, 0))^2 + E_b(x_i, l_2))^2 \right] ,
\end{aligned}
$$

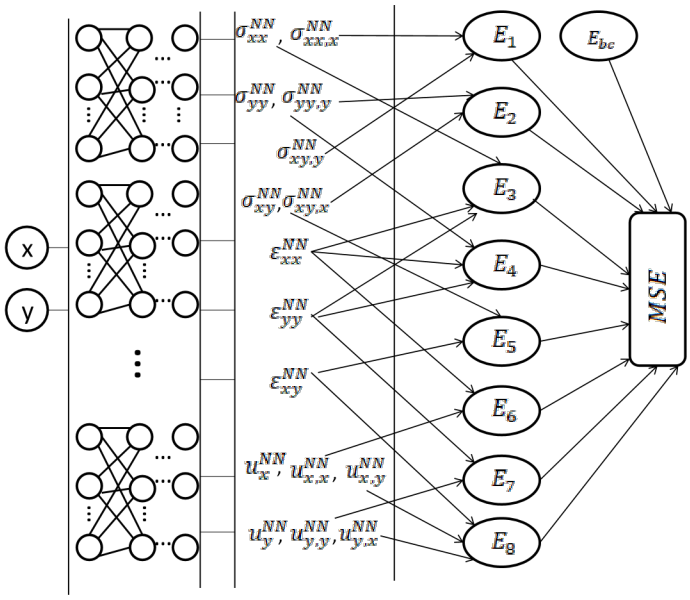


Fig. 4: Architecture of the residual model in the multi-PINN.

where $E_b(0,y_i)$, $E_b(l_1,y_i)$, $E_b(x_i,0)$ and $E_b(x_i,l_2)$ are the residuals on the boundaries and $(0,y_i)$, (l_1,y_i), $(x_i,0)$ and (x_i,l_2) are the boundary collocation points.

All the residuals for the system (1)-(3) are being computed during the training process. Therefore, the residual is the error obtained after substituting the approximate solution for σ_{ij}^{NN}, ε_{ij}^{NN}, u_x^{NN} and u_y^{NN} from the surrogate networks, in the system of equations (1)-(3).

It is noted that the calculation of the residual requires computation of the differential operators, entering in the system. This calculation is performed using automatic differentiation ([2], [13]) that allows determining the differential operators at any given point without adopting any discretization and mesh as it usually happens in difference methods. The residual network provides the surrogate networks with the loss function (mean square error). The parameters of the neural networks, namely, the weights and biases, are being updated during the training process by minimizing the error function (9), within backpropagation. Updating of the weights and biases is usually considered using a stochastic optimizer, such as the Stochastic Gradient Descent (SGD) and Adam's method [21]. Here, the residual network updates the surrogate networks' weights and biases using the residual obtained from the residual network, adopting the Adam optimization algorithm. From a computational point of view, one of the advantages of using PINN methods for scientific computing is the possibility of exploiting high-performance algorithms, such as the Tensorflow [1] or PyTorch and Keras library [7], designed specifically for deep learning and artificial intelligence.

4 Computational algorithm with feedforward and backpropagation

In this section the proposed computational procedure of solving the system (1)(3) using Python programming scientific software Tensorflow, "built in" Keras is described. In order to optimize the weights and biases the Adam's optimization algorithm is applied. The training set is divided into butches. The steps for the implementation of the technique, with some key functions and modules on Python using automatic differentiation method of Tensorflow and Keras library for deep learning, are presented below[1]

1. Import all necessary Python libraries:

   Numpy, Tensorflow, Keras, Matplotlib.pyplot

   etc.
2. Set up physical parameters of the problem.
3. Set up input, training and test samples (collocation points), output and a number of training iterations (epochs) for the neural networks.
4. Assign the input data for the neural networks. Two input vectors **x**,**y** are considered, which are the coordinates of the collocation points defined for the system (1)-(3) and for the boundary conditions.
5. Construct the surrogate net models for σ_{ij}^{NN}, ε_{ij}^{NN}, u_x^{NN} and u_y^{NN} with using

   keras.Input, keras.layers.Dense, keras.models.Model

   methods.
6. Set up the boundary conditions.
7. Compose functions in Python for the automatic differentiation using Tensorflow

   GradientTape

   module.
8. Implement the automatic differentiation by calling the Python *def* functions, defined in Step 7.
9. Formulate the residuals $E_1,E_2,\ldots,E_8$ (see Figure 4) for the system (1)-(3) and the residual-vector $\mathbf{E_b}$ of the associated boundary conditions. 10. Construct the residual model, based on the Keras method keras.models

   including the variables x,y as input and the residuals $E_1,E_2,\ldots,E_8$, and $\mathbf{E_b}$ as output.
11. Provide the input values for the residual model from Step 3 for creating training and testing input data and the values $E_i = 0$, $i = 1,2,\ldots,8$, $\mathbf{E_b} = \mathbf{0}$ for creating the training output.
12. Compile the residual neural model using the Keras module

    keras.models.Model.compile

[1] The entire software package is available upon request.

with the help of the built in Adam's optimizer and *Mean Square Error* modules.

13. Train the multi-PINN model (the residual with the surrogate models) with the training input and output data.
14. Sketch graphs for the model accuracy, model loss and approximate solution for the system using *Matplotlib* software.

5 Numerical examples

Example 1: Verification of the multi-PINN model using an analytical solution

In order to solve the equations (1)-(3) it is needed to set up boundary conditions. In order to have a unique solution for (1)-(3) eight boundary conditions are needed to set up. These boundary conditions will be derived for the displacements and the components of the stress tensor of the investigated structures.

For a verification of the multi-PINN model an example similar with the example from the article [16] has been tested. The structure has a rectangular shape, $0 \le x \le l_1, 0 \le y \le l_2$, where l_1, l_2 are the lengths of the sides, and it is fixed on the side $y = 0$ and under extension on the other parallel side $y = l_2$. The boundary conditions read

$$u_x(x,0) = 0,\ u_y(x,0) = 0,\ u_x(x,l_2) = 0,\ \sigma_{yy}(x,l_2) = P(x),$$
$$\sigma_{xx}(0,y) = 0,\ u_y(0,y) = 0,\ \sigma_{xx}(l_1,y) = 0,\ u_y(l_1,y) = 0, \tag{10}$$

where the function $P(x) = (\lambda + 2\mu)Q\sin(\pi x)$. The external extension forces are:

$$f_x = \lambda\left[4A\pi^2\cos(2\pi x)\sin(\pi y) - \pi\cos(\pi x)Qy^3\right] + \mu\left[9A\pi^2\cos(2\pi x)\sin(\pi y) - \pi\cos(\pi x)Qy^3\right], \tag{11}$$

$$f_y = \lambda\left[-3\sin(\pi x)Qy^2 + 2A\pi^2\sin(2\pi x)\cos(\pi y)\right] + \mu\left[-6\sin(\pi x)Qy^2 + 2A\pi^2\sin(2\pi x)\cos(\pi y) + \pi^2\sin(\pi x)Qy^4/4\right] \tag{12}$$

and the exact solution of the problem (1)-(3), (10)-(12), is written as

$$u_x(x,y) = A\cos(2\pi x)\sin(\pi y), \tag{13}$$
$$u_y(x,y) = Q\sin(\pi x)y^4/4. \tag{14}$$

Is is considered a case of square structure, i.e. $l_1 = l_2 = 1$m. The structure is shown on Figure 5. The Lam´e parameters take the values $\lambda = 1$, $\mu = 0.5$ (in GPa), respectively. The displacements, the strain and stress components have been computed with $A = 0.02$ and $Q = 0.5$. The samples (collocation points) for training data and the collocation points for test data have been taken 1600 ($N_1 = N_2 = 40$) and 900 respectively. The batch size is 64.

The loss error function after 15000 epochs (training iterations) of the PINN with 4 hidden layers [15,30,30,15] neurons is $1.04 \cdot 10^{-4}$. On Figures 6, 7 the neural approximate and exact (13), (14) solutions at the collocation points are presented. The error

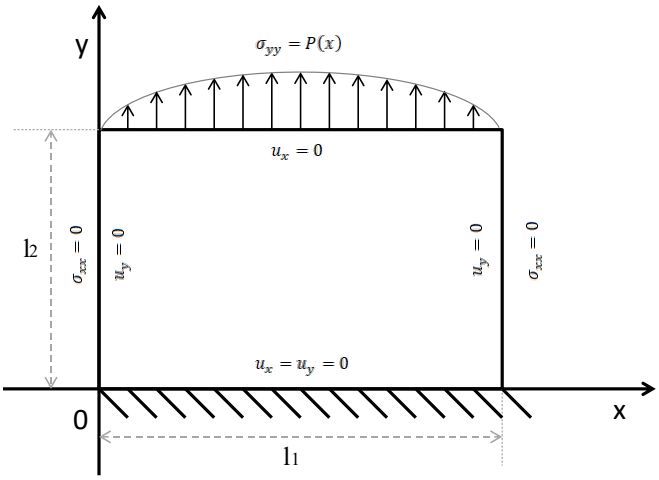


Fig. 5: Structure with the fixed and with extensional loading edges. of

approximation is

$$\delta_x = \max_{(x_i,y_j)} |u^{NN}{}_k(x_i,y_j) - u_k(x_i,y_j)| < 2.46 \cdot 10^{-3},\ k = 1,2.$$

On Figure 8 the surfaces of the neural solution are shown. The predicted components of the stress and strain tensors at the collocation points, computed by the PINN are sketched on Figures 9, 10.

Example 2

In this example the structure from Example 1 but with a constant extension stress loading $P(x) = const = 0.3$ GPa (or 300MPa) and $f_x = f_y = 0$ is considered. The boundary conditions for the displacements and strain functions are the same, i.e., as in (10). The neural solution of the problem (1)-(3), (10) is compared with the finite element approximation from ABAQUS. The constructed PINN consists of 4 hidden layers with [15,30,30,15] neurons. For training the PINN 15000 epochs (iterations), 64 batch size have been provided. A number of the collocation points (samples) has been taken 625 (25x25) and a number of test collocation points has been taken 225, respectively.

Correspondingly for ABAQUS 25x25 nodes have been taken. The force, which is equal to 0.012GN (or 12×10^3KN), that makes a total force in 25 collocation points equal to $0.012 \times 25 = 0.3$GN (applied at a surface of 1m^2). For a simulation using ABAQUS the same boundary conditions (10) have been taken.

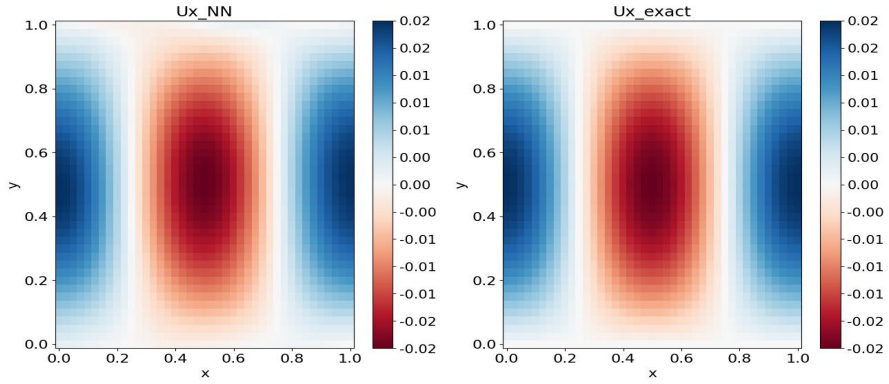


Fig. 6: Displacement u_x^{NN} (in m) after the training of the multi-PINN and the exact solution u_x.

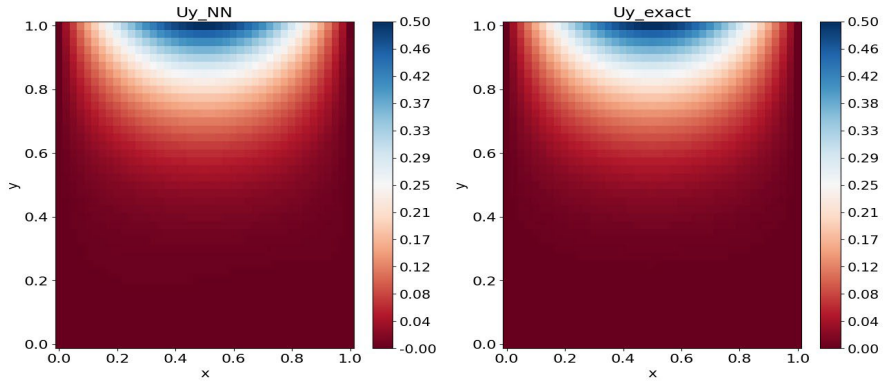


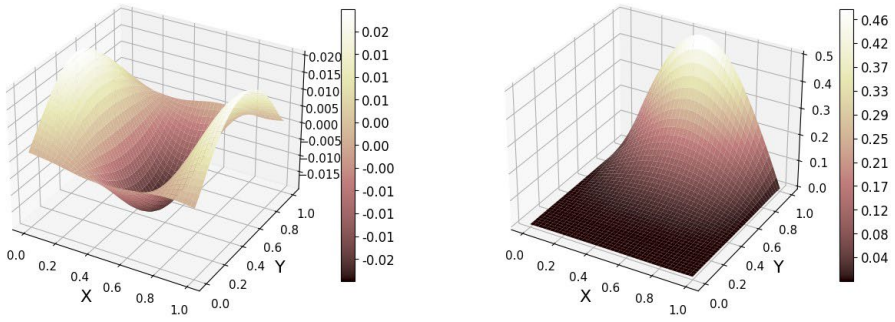
Fig. 7: Displacement u_y^{NN} (in m) after the training of the multi-PINN and the exact solution u_y.

The modulus elasticity E and ν, which are needed to input to the ABAQUS can be computed from the relations

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}.$$

The computed solutions with the application of ABAQUS at the nodes and the PINN at the collocation points are presented in Figures 11-18.

The numerous numerical experiments have shown that the accuracy of computations depends mostly on a number and a choice of collocation points, a number of epochs, layers and neurons. Table 1 shows the loss error estimates with respect to the



u_x^{NN} u_y^{NN}

Fig. 8: Surfaces for the displacements (solution) and (in m) after the training of the multi-PINN.

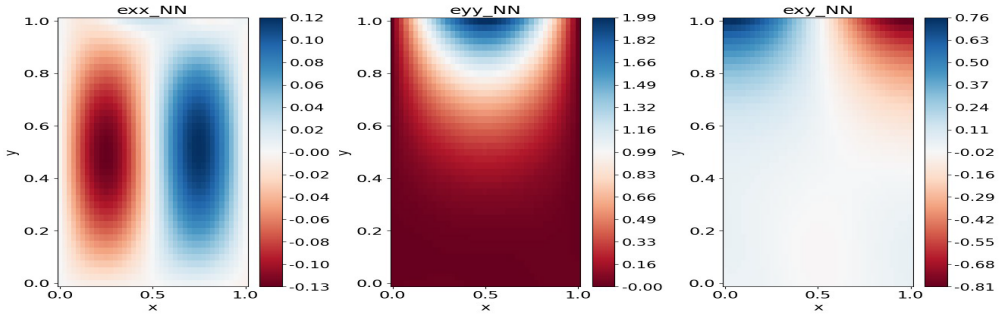
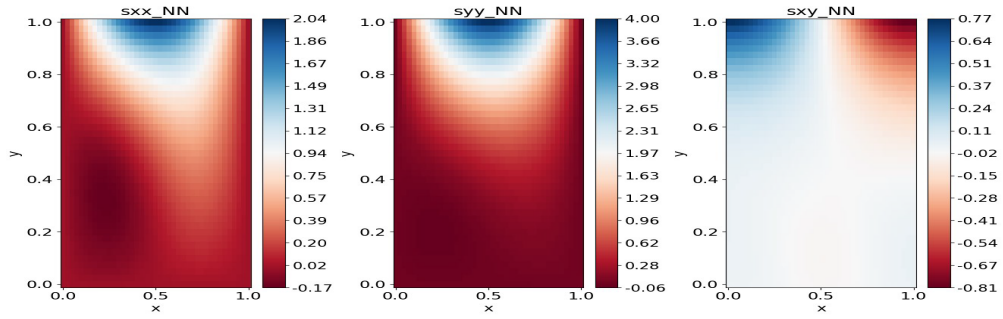


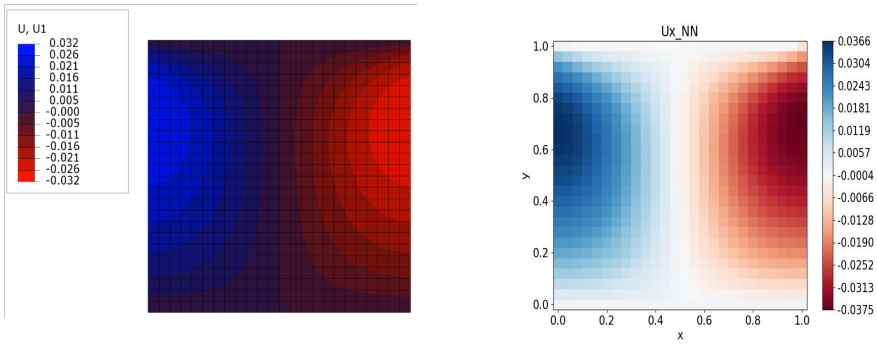
Fig. 9: Strain functions $e_{xx}^{NN}, e_{yy}^{NN}, e_{xy}^{NN}$ after the training of the multi-PINN.

number of epochs, layers and neurons. A deep learning provides a fast convergence of the approximate solution (output of the NN) to the exact solution, however during the process of minimization an accumulation of errors can be occur, which is a result of the complexity of the neural network. Therefore, the parameters of the neural network and a number of samples (collocation points) should be chosen carefully. A choice of collocation points is also important in order to predict a behaviour of the solution.



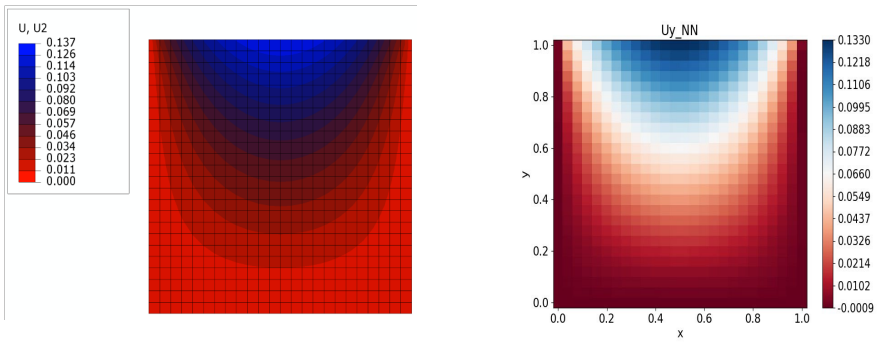
s_{xx}^{NN} ,s_{yy}^{NN} ,s_{xy}^{NN}

Fig. 10: Stress functions (in GPa) after the training of the multiPINN.



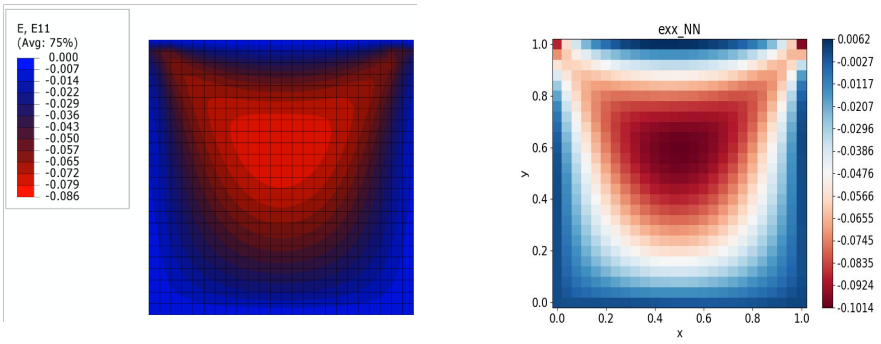
(a) (b)

Fig. 11: Displacement u_x (in m) of the structure, computed with using: a) ABAQUS, b) multi-PINN.



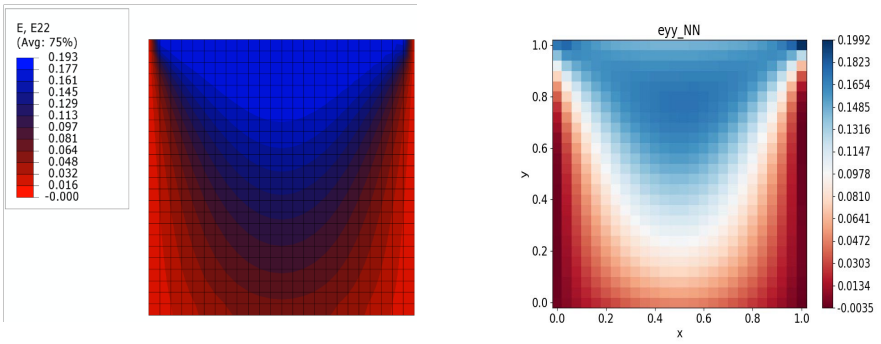
(a) (b)

Fig. 12: Displacement u_y (in m) of the structure, computed with using: a) ABAQUS, b) multi-PINN.



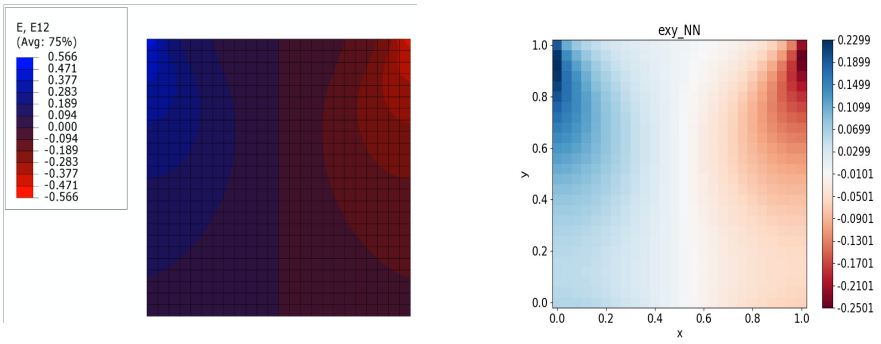
(a) (b)

Fig. 13: Strain functions ε_{xx} of the structure, computed with using: a) ABAQUS, b) multi-PINN.



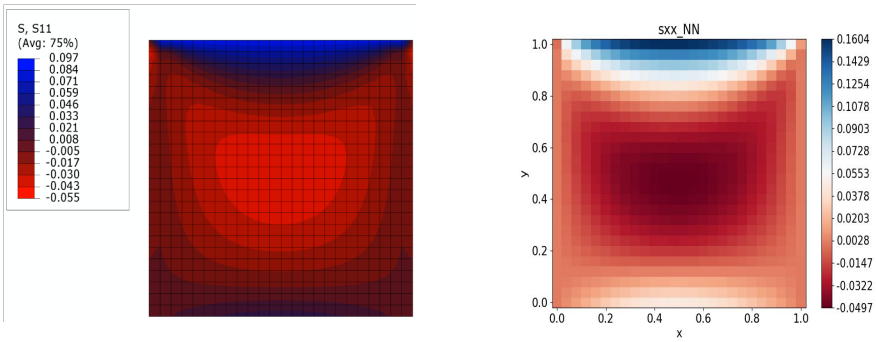
(a) (b)

Fig. 14: Strain functions ε_{yy} of the structure, computed with using: a) ABAQUS, b) multi-PINN.



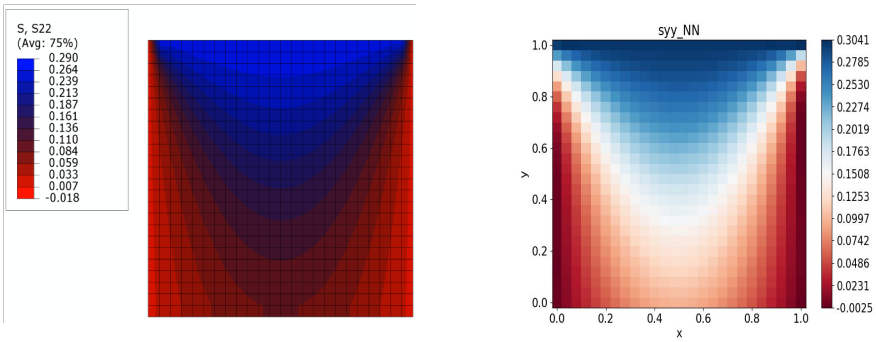
(a) (b)

Fig. 15: Strain functions ε_{xy} of the structure, computed with using: a) ABAQUS, b) multi-PINN.



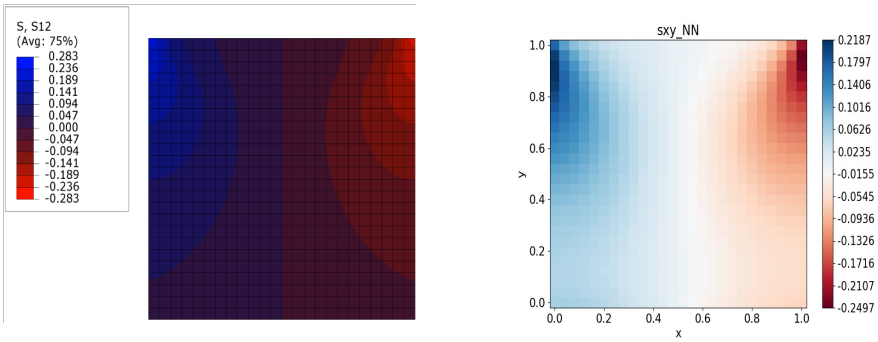
(a) (b)

Fig. 16: Stress functions σ_{xx} (in GPa) of the structure, computed with using: a) ABAQUS, b) multi-PINN.



(a) (b)

Fig. 17: Stress functions σ_{yy} (in GPa) of the structure, computed with using: a) ABAQUS, b) multi-PINN.



(a) (b)

Fig. 18: Stress functions σ_{xy} (in GPa) of the structure, computed with using: a) ABAQUS, b) multi-PINN.

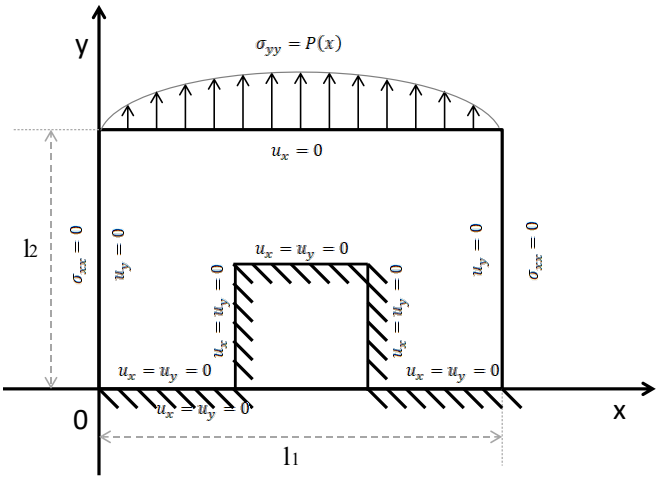


Fig. 19: Structure with the opening, the fixed and the extensional loading edges.

Table 1: The model loss for different values of the training parameters

Neurons in Layers	Epochs	Training Samples	Loss Function	Time(min.)
[15,15]	2000	625	$3.64 \cdot 10^{-4}$	6
[15,15]	4000	625	$1.78 \cdot 10^{-4}$	12
[15,15]	10000	625	$1.08 \cdot 10^{-4}$	28
[15,15]	15000	625	$8.82 \cdot 10^{-5}$	33
[15,20,15]	2000	900	$9.46 \cdot 10^{-5}$	6
[15,20,15]	4000	900	$8.68 \cdot 10^{-5}$	12
[15,30,30,15]	2000	900	$8.12 \cdot 10^{-5}$	6.5
[15,30,30,15]	4000	900	$2.37 \cdot 10^{-5}$	13
[15,30,30,15]	4000	1600	$4.90 \cdot 10^{-5}$	15
[15,30,30,15]	15000	1600	$1.22 \cdot 10^{-5}$	56
[15,30,30,15]	40000	1600	$8.35 \cdot 10^{-6}$	125

Example 3: The plane stress structure with an opening

A plane stress structure with an opening and with the constant extension stress loading $P(x) = const = 0.3$GPa in the y-direction and $f_x = f_y = 0$ is considered (see Figure 19). The boundary conditions are shown on Figure 20. Here $l_1 = 1.5$m, $l_2 = 1$m. The square opening with the width and the height equal to 0.5m is considered, and an equal distance from the opening till the right and left edges of the structure is taken.

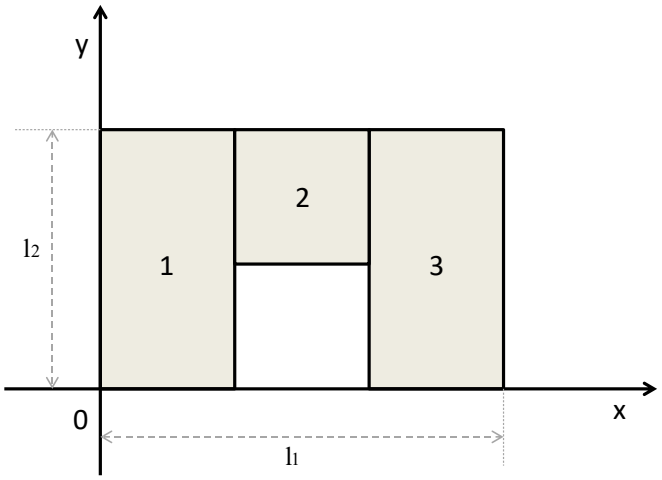


Fig. 20: The elements of the structure with the opening.

For the training of the PINN the domain of structure was divided into 3 parts (rectangular elements). For the first element and the third elements 13x41, for the second element 15x20 collocation points (training samples) are used.

The constructed PINN consists of 4 hidden layers with [15,30,30,15] neurons. For training the PINN 15000 epochs (iterations), 64 batch size have been provided. The loss error is $5.4703 \cdot 10^{-5}$. The neural solution of the problem (1)-(3) with opening is compared with the finite element approximation from ABAQUS. Correspondingly an equivalent mesh has been adopted in the commercial software. The force, which is equal to 0.018GN (or 18×10^3 KN) has been applied to each node on the edge $y = l_2$. This provides a total force 0.018GN x 25 (collocation points) applied to the area of 1.5m^2 resulting in the stress 0.3GPa.

The complexity of the geometry influences the accuracy of computations. The sensitivity of the neural models with respect to layers and neurons with the same number of collocation points can be seen in Figure 21. From Figure 21 we can conclude that deep learning, i.e., many layers and neurons can provide fast decreasing of the loss function (MSE) with increasing number of training iterations (epochs) and convergence to an exact solution of the problem. The time of computations for Example 3 is similar with the time of computations for Example 2.

The predicted solution by the multi-PINN and by using ABAQUS are illustrated in Figures 22-29.

It can be noted that a choice of collocation points can be also based on random distribution in each part of the structure domain. A deep learning provides a fast

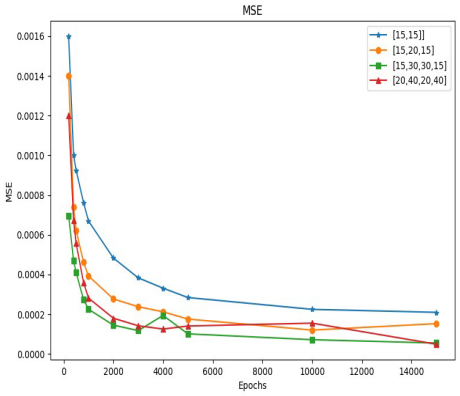
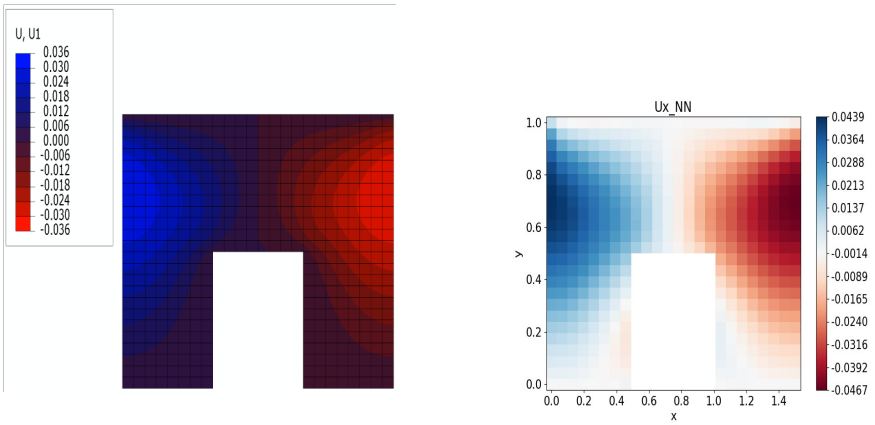


Fig. 21: The computed MSE (loss error) with different number of layers and neurons.



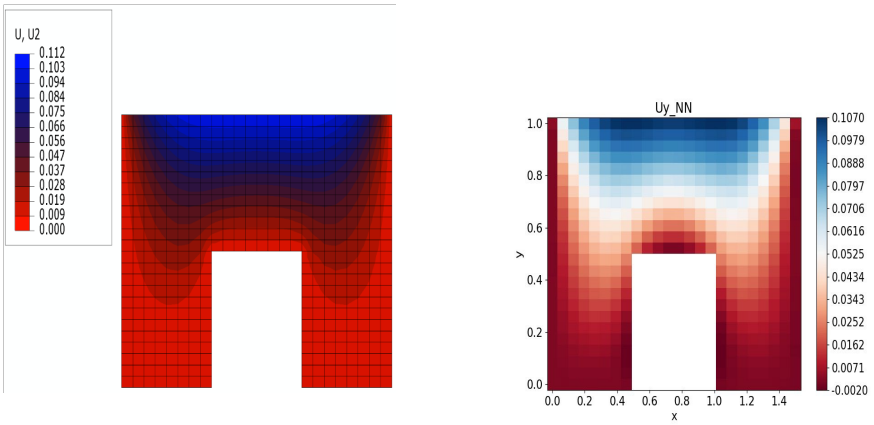
(a) (b)

Fig. 22: Displacement u_x (in m) of the structure, computed with using: a) ABAQUS, b) multi-PINN.

convergence of the numerical solution of the PINN to the exact solution. A choice of collocation points is important in order to have a good convergence and predict a behaviour of the solution. Note that the number of training iterations (epochs) for reaching desirable accuracy increases in more complex problems.

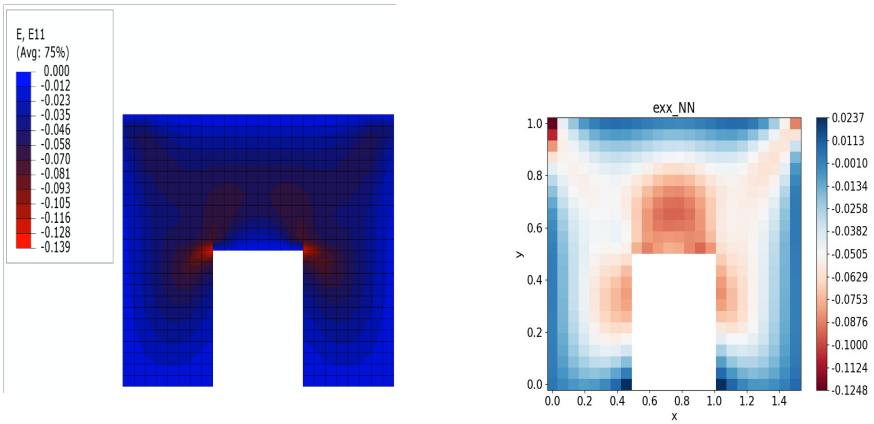
6 Conclusions

An ensemble of PINNs has been proposed for solving two-dimensional elasticity equations. The developed multi-PINN consists of eight surrogate models for unknowns σ_{xx}^{NN}, σ_{yy}^{NN}, σ_{xy}^{NN}, ε_{xx}^{NN}, ε_{yy}^{NN}, ε_{xy}^{NN}, u_{xx}^{NN} and u_{yy}^{NN} and one residual model for a construction of loss error function, training the PINN and minimizing the error. As result



(a) (b)

Fig. 23: Displacement u_y (in m) of the structure, computed with using: a) ABAQUS, b) multi-PINN.



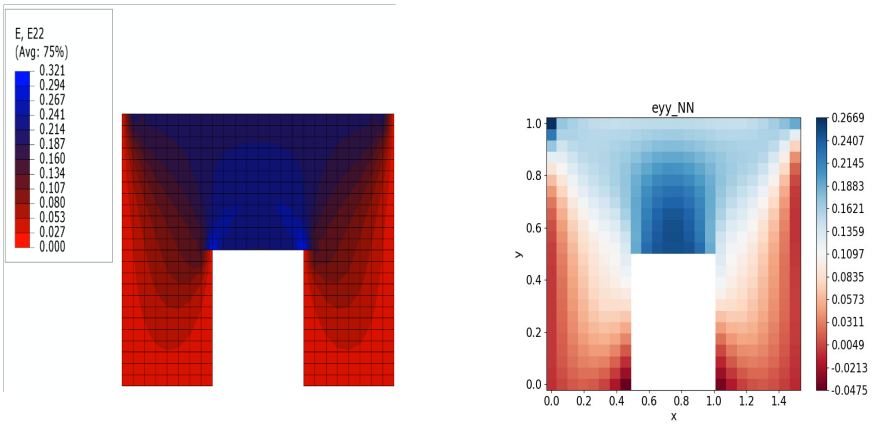
(a) (b)

Fig. 24: Displacement e_{xx} of the structure, computed with using: a) ABAQUS, b) multi-PINN.

of training optimal values for weights and biases of the multi-PINNs for different elastic problems have been obtained.

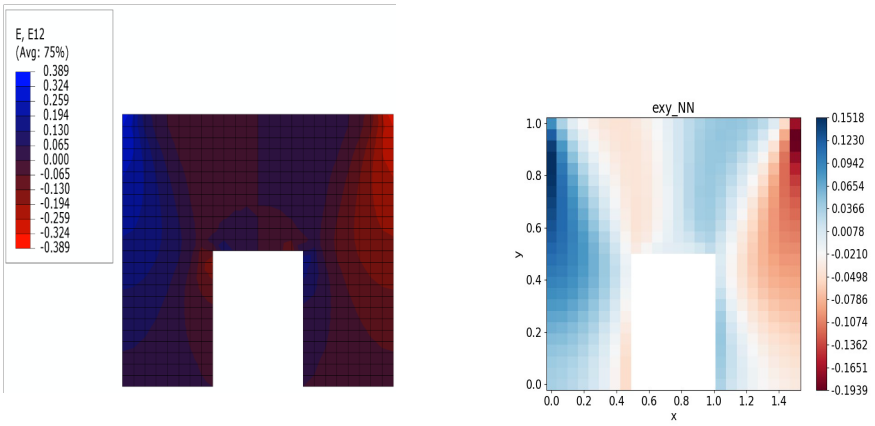
The proposed machine learning method utilizes equilibrium conditions and linear elasticity equations in a novel architecture of the ensemble of surrogate models considering unknown stresses, strains and displacements. The model is then able to predict the mechanical response of two-dimensional structures without pre-existing, input and output data. The proposed architecture of the multi-PINN model in comparison with classical finite element analysis and other numerical techniques (e.g., difference

methods) aims to reduce the error of calculating the strains, stresses and displacements because of introducing unknown strains, stresses and displacements directly in the



(a) (b)

Fig. 25: Displacement e_{yy} of the structure, computed with using: a) ABAQUS, b) multi-PINN.

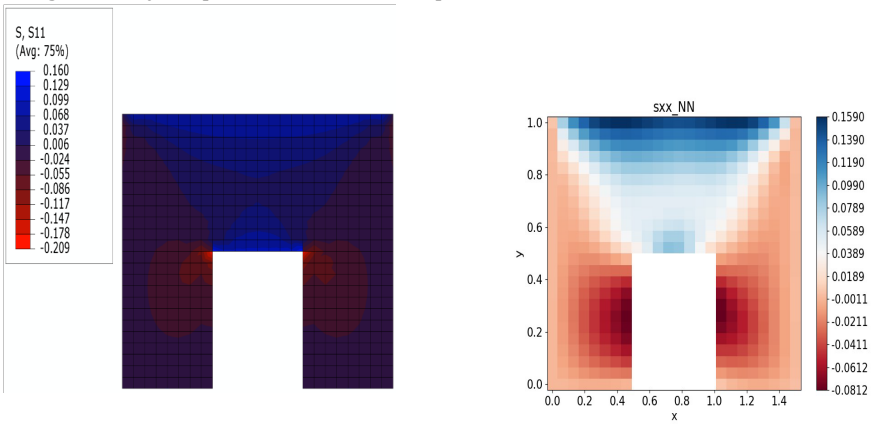


(a) (b)

Fig. 26: Displacement e_{xy} of the structure, computed with using: a) ABAQUS, b) multi-PINN.

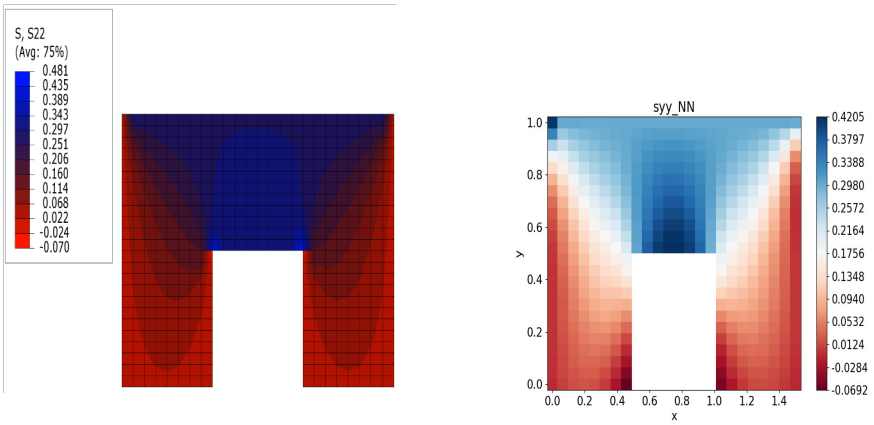
multi-PINN model and using automatic differentiation, based on the chain rule for computing the derivatives. Here it should be noticed that usage of automatic differentiation, based on the chain rule in PINN leads to lower numerical error in comparison to numerical differentiation within finite difference methods. Therefore, by using PINNs we can obtain more accurate estimates for the derivatives. However, in problems with phenomena of multi-solutions the use of the PINN techniques may be limited. In this case a combination with classical approaches, e.g., spectral finite elements

etc. can be performed. Before solving some problem by applying PINN it must be guaranteed an existence and uniqueness of the solution of the considered model, involving ordinary or partial differential equations.



(a) (b)

Fig. 27: Displacement s_{xx} (in GPa) of the structure, computed with using: a) ABAQUS, b) multi-PINN.

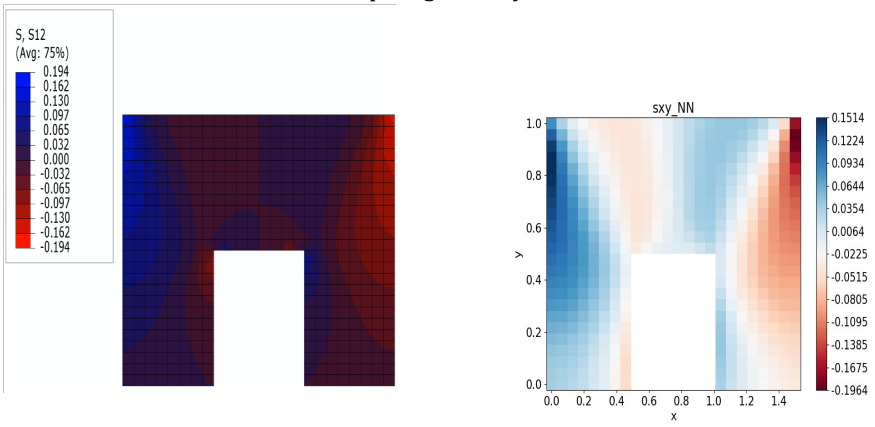


(a) (b)

Fig. 28: Displacement s_{yy} (in GPa) of the structure, computed with using: a) ABAQUS, b) multi-PINN.

The proposed technique has been tested on an example when the exact solution is known. The approach with eight surrogate models and one residual model provides a high rate of convergence and good accuracy. The results have been compared with the results, obtained after using FEMs of ABAQUS in two examples of structure (a rectangular plane stress and a rectangular plane stress with an opening). Here the nodes have been considered as collocation points. In the example of structure with an opening the domain

of the definition of the solution is divided into parts (elements) and training of the multi-PINN at collocation points (samples) is performed in each part in order to improve the simulation process of computing the unknowns of the problem. This approach can be applied to structures with more complex geometry.



(a) (b)

Fig. 29: Displacement s_{xy} (in GPa) of the structure, computed with using: a) ABAQUS, b) multi-PINN.

Comparison with the results obtained from the commercial software indicates that the contour plot distributions of displacements, strains and stresses derived from the proposed multi-PINN model are pretty similar to the ones received from the commercial software. Some differences near the boundaries (higher-lower) values of the multi-PINN model and commercial finite element analysis plots are obtained, attributed to the approximate nature of the proposed approach and the traditional finite element analysis as well as to parameters like the mesh density and the number of collocation points.

The accuracy and the computational effort required for the implementation of the model depend on the number of collocation points, adopted for the simulation. A significant part in the implementation of the method is the consideration of appropriate stress and displacement boundary conditions. The combination and type of these conditions influence the applicability and generalizability of the method.

The described technique can be extended to structures with nonlinear material properties, where the stress and strain tensors have nonlinear relations as well as to geometric nonlinearities. In addition, the technique can easily be applied to threedimensional plane elasticity problems. Furthermore, since PINNs are able to tackle complicated mixed boundary conditions, application to homogenization and multiscale methods is possible. Those steps are left for future investigation.

References

[1] Abadi, M.P., Barham, J., Chen, Z., Chen, A., Davis, J., Dean, M., Devin,

S., Ghemawat, G., Irving, M., Isard, M., Kudlur, J., Levenberg, R., Monga, S., Moore, D., G., Murray, B., Steiner, P., Tucker, V., Vasudevan, P., Warden, M. Wicke, Y., Yu, X., Zheng: Tensorflow: A system for largescale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), USENIX Association, Savannah, GA, pp. 265–283, URL https://www.usenix.org/conference/osdi16/technicalsessions/presentation/abadi. (2016)

[2] Baydin, A., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic Differentiation in Machine Learning: a Survey, https://arxiv.org/pdf/1502.05767.pdf (2018)

[3] Bazmara, M., Mianroodi, M., Silani, M.: Application of physics-informed neural networks for nonlinear buckling analysis of beams. Acta Mech. Sin. **39**, 422438 (2023)

[4] Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E.: Physics-informed neural networks (pinns) for fluid mechanics: A review. Acta Mech. Sin., 1–12 (2022)

[5] Chadha, C., He, J., Abueidda, D., Koric, S., Guleryuz, E., Jasiuk, I.: Improving the accuracy of the deep energy method. Acta Mech. **234**, 5975–5998 (2023)

[6] Chen, Y., Lu, L., Karniadakis, G. E., and Dal Negro, L.: Physics-informed neural networks for inverse problems in nano-optics and metamaterials. Optics express **28**, 8, 11618–11633 (2020)

[7] Chollet, F.: Deep Learning with Python, Manning Publications Company, 2017, URL https://books.google.ca/books?id= Yo3CAQAACAAJ.

[8] Drosopoulos, G. A., Stavroulakis, G. E.: Non-linear Mechanics for Composite, Heterogeneous Structures, CRC Press, Taylor and Francis (2022)

[9] Drosopoulos G.A., Stavroulakis G.E.: Data-driven Computational Homogenization Using Neural Networks: FE2-NN Application on Damaged Masonry, ACM Journal on Computing and Cultural Heritage, **14**, 1-19 (2020)

[10] Faroughi, S., Darvishi, A., Rezaei, S.: On the order of derivation in the training of physics-informed neural networks: case studies for non-uniform beam structures. Acta Mech. **234**, 5673–5695 (2023)

[11] Faroughi, S.A., Pawar, N., Fernandes, C., Raissi, M., Das, S., Kalantari, N.K., Mahjour, S.K.: Physics-Guided, Physics-Informed, and Physics-Encoded Neural Networks in Scientific Computing arXiv epring 2211.07377 (2023)

[12] Fletcher, R.: Practical Methods of Optimization (2nd ed.). John Wiley & Sons, New York (1987)

[13] Gu¨ne, A., Baydin, G., Pearlmutter, B.A., Siskind, J.M.: Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. **18**, 1–43, (2018) URL http://www.jmlr.org/papers/volume18/17-468/17-468.pdf

[14] Guo, M., Haghighat, E.: An energy-based error bound of physics-informed neural network solutions in elasticity. arXiv preprint arXiv:2010.09088 (2020)

[15] Haghighat, E., and Juanes, R. Sciann: A keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. Comp. Meth. in Appl. Mech. and Eng. 373, 113552 (2021)

[16] Haghighat, E., Raissi, M., Moure, A., Gomez, H., and Juanes, R.: A deep learning framework for solution and discovery in solid mechanics. https://arxiv.org/abs/2003.02751 (2020)

[17] Harandi, A., Moeineddin, A., Kaliske, M., Reese, S., Rezaei, S.: Mixed formulation of physics-informed neural networks for thermo-mechanically coupled systems and heterogeneous domains. Int J Numer. Methods Eng. **125**, e7388 (2024)

[18] Kadeethum, T., Jorgensen, T., Nick, H.: Physics-informed neural networks for solving nonlinear diffusivity and Biot's equations. PLoS ONE, **15**:e0232683 (2020)

[19] Karniadakis, G.E., Kevrekidis, I.G., Lu, L. et al. Physics-informed machine learning. Nat Rev Phys. **3**, 422–440 (2021)

[20] Katsikis, D., Muradova, A.D. and Stavroulakis, G.S.: A Gentle Introduction to Physics-Informed Neural Networks, with Applications in Static Rod and Beam Problems, Jr. Adv. Appl. & Comput. Math. **9**, 103-128 (2022)

[21] Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, Computer Science, Mathematics, The International Conference on Learning Representations (ICLR) (2015)

[22] Kortesis, S. and Panagiotopoulos, P.D.: Neural networks for computing in structural analysis: Methods and prospects of applications. Int. Jr. Numeric. Meth. Eng. **36**, 2305-2318 (1993)

[23] Kovachki, N. Lanthaler, S, Mishra, S.: On Universal Approximation and Error Bounds for Fourier Neural Operator, Journal of Machine Learning Research **22**, 1-76 (2021)

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

[25] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations, arXiv preprint arXiv:2010.08895 (2020)

[26] Liu, Dong, C., Nocedal, J.: On the limited memory BFGS method for large scale optimization, Mathematical Programming volume, **45**, 503–528 (1989)

[27] Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G.E.: Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat Mach Intell **3**, 218–229 (2021) https://doi.org/10.1038/s42256-021-00302-5.

[28] Meade, A.J., Fernandez, A.A.: The numerical solution of linear ordinary differential equations by feed-forward neural networks, Mathematical and Computer Modelling **19** 1–25 (1994) URL: https://www.sciencedirect.com/science/article/pii/0895717794900957. doi:10.1016/0895-7177(94)90095-7.

[29] Muradova, A.D., Stavroulakis, G.E.: Physics-informed neural networks for elastic plate problems with bending and Winkler-type contact effects. Jr. of the Serb. Soc. for Comput. Mech. **15**, 45-54 (2021)

[30] Muradova, A.D. and Stavroulakis, G.E.: Physics-informed Neural Networks for the Solution of Unilateral Contact Problems, Book of Proceedings, 13th International Congress on Mechanics HSTAM, 451-459 (2022). https://hstam2022.eap.gr/book-of-proceedings/

[31] Muradova, A.D., Stavroulakis, G.E.: The projective-iterative method and neural network estimation for buckling of elastic plates in nonlinear theory. Commun. in Nonlin. Sci. and Num. Sim. **12**, 1068-1088 (2007)

[32] Niu, S., Zhang, E., Bazilevs, Y., Srivastava, V.: Modeling finite-strain plasticity using physics-informed neural network and assessment of the network performance. Jr. of the Mech. and Phys. of Solids **172**, 105177 (2023)

[33] Raissi, M., Perdikaris, P., Karniadakis, G. E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Jr. of Comput. Phys. **378**, 686–707 (2019)

[34] Rezaei, S., Harandi, A., Moeineddin, A., Xu, B-X., Reese, S.: A mixed formulation for physics-informed neural networks as a potential solver for engineering problems in heterogeneous domains: Comparison with finite element method. Comput. Meth. in Appl. Mech. and Eng. **401**, Part B: 115616 (2022)

[35] Ruder, S.: An overview of gradient descent optimization algorithms (2017) https://arxiv.org/abs/1609.04747

[36] Shin, Y., Darbon, J., Karniadakis, G. E.: On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. Commun. Comput. Phys. **28**, 2042–2074 (2020)

[37] Stavroulakis, G.E.: Inverse and Crack Identification Problems in Engineering Mechanics, Springer (2000)

[38] Stavroulakis, G.E., Avdelas, A., Abdalla, K.M., Panagiotopoulos P.D.: A neural network approach to the modelling, calculation and identification of semi-rigid connections in steel structures, Jr. of Construct. Steel Research **44**, 91-105 (1997)

[39] Stavroulakis, G., Bolzon, G., Waszczyszyn, Z. and Ziemianski, L.: Inverse analysis. In: Karihaloo B, Ritchie RO, Milne I (eds in chief) Comprehensive structural integrity, Numerical and computational methods, Vol 3, Chap 13. Elsevier, Amsterdam, 685–718 (2003)

[40] Tartakovsky, A. M., Marrero, C. O., Perdikaris, P., Tartakovsky, G. D., and Barajas-Solano D.: Learning parameters and constitutive relationships with physics informed deep neural networks. arXiv preprint arXiv:1808.03398 (2018)

[41] Theocaris, P. S., and Panagiotopoulos, P. D.: Plasticity Including the Bauschinger Effect, Studied by a Neural Network Approach. Acta Mech. **113**, 63–75 (1995) doi:10.1007/BF01212634

[42] Waszczyszyn Z. and Ziemian´ski L.: Neural Networks in the Identification Analysis of Structural Mechanics Problems. In: Mr´oz Z., Stavroulakis G.E. (eds) Parameter Identification of Materials and Structures, CISM International Centre for Mechanical Sciences (Courses and Lectures), **469**, Springer, Vienna (2005)

[43] Xiao, Y., Zengxin, Wei, Z., Zhiguo, Wang, Zh.: A limited memory BFGS-type method for large-scale unconstrained optimization, Comput. Math. with Applic. **56**, 1001-1009 (2008)

[44] Yagawa, G., Oishi, A.: Computational mechanics with neural networks. Springer (2021)