Multi-Physics-informed deep learning approach for computational structural mechanics

Xuan Kong (kongxuan@hnu.edu.cn)
College of Civil Engineering, Hunan University
Weiwei He
College of Civil Engineering, Hunan University
Jinzhao Li
kingle@hnu.edu.cn
Lu Deng
denglu@hnu.edu.cn

Article

Keywords:

Posted Date: October 13th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3363237/v1

Additional Declarations: There is **NO** Competing Interest.

Multi-Physics-informed deep learning approach for computational structural mechanics

Weiwei He[1,a], Jinzhao Li[2,a], Xuan Kong[3,*], Lu Deng[4]

[1] PhD Candidate, College of Civil Engineering, Hunan University, Changsha 410082, China, E-mail: heweiwei@hnu.edu.cn

[2] Associate research fellow, College of Civil Engineering, Hunan University, Changsha 410082, China, E-mail: kingle@hnu.edu.cn

[3] Professor, Key Laboratory for Damage Diagnosis of Engineering Structures of Hunan Province, College of Civil Engineering, Hunan University, Changsha 410082, China, E-mail: kongxuan@hnu.edu.cn (Corresponding author)

[4] Professor, Key Laboratory for Damage Diagnosis of Engineering Structures of Hunan Province, College of Civil Engineering, Hunan University, Changsha 410082, China, E-mail: denglu@hnu.edu.cn

[a] These authors contributed equally: Weiwei He, Jinzhao Li.

* Corresponding author

Abstract

Finite element method (FEM) has been popular for decades for structure analysis by solving partial differential equations (PDEs). Recently, physics-informed deep learning has emerged as a promising approach for solving PDEs and shows great advantages over FEM in the computational efficacy. However, it is still a challenge for the structural mechanics computation since it involves solving higher-order PDEs as the governing equations are fourth-order nonlinear PDEs. Therefore, we develop a

novel multi-physics-informed neural network (m-PINN) framework via combining multiple neural networks (NNs) where each NN representing physics information such as geometrical compatibility, constitutive relation, and static equilibrium. The verification result demonstrates that m-PINN has the same accuracy as FEM for the computation of beam and shell structures. The proposed method has the potential to be a new paradigm in the structural mechanics computation as an alternative of FEM, and could serve as an intelligent computational module in digital twin system.

Main text

Structural mechanics computation plays an important role in the field of civil and structure engineering, which is fundamental for the design and analysis of structures. In particular, with the emergence of Digital Twin (DT) [1-3], it requires establishing a virtual digital model and implementing real-time simulation to accurately reflect the real physical structure. Therefore, the accurate and efficient computation of structural behavior is critical in the DT system [4, 5]. Currently, the FEM is commonly used in structural computation, and numerous commercial FEM software (e.g., ANSYS and ABAQUS) are available. However, there exit three main limitations with regards to FEM, especially as a computational module in DT system [6]: (i) The repeated establishment and computation of FEM model in terms of model updating process hardly satisfies the real-time computation in the DT system due to the cumbersome and time-consuming updating process. (ii) The encapsulated FEM software is hardly available to be embedded in the DT system due to the limitation of the integrated software. (iii) The FEM approach cannot handle the infinite domain problems in

which the boundary conditions or initial conditions may be unknown.

With the rise of artificial intelligence (AI) in recent years, especially the advancement of deep learning developed from neural network, AI technology has been broadly used in the field of engineering structures [7-10], such as damage identification [11-14], model optimization [15-16], and performance prediction [17-20]. However, the neural networks remain a black box that cannot be explicitly described in a physical sense, and this purely data-driven method heavily relies on vast amounts of high-fidelity data. To overcome this problem, the physics-informed neural network (PINN) has been proposed [21, 22], in which the underlying physical laws are considered via incorporating the physical equations into the network. PINN has the capability to learn a priori knowledge of partial differential equations (PDEs) with or without labeled data. Moreover, PINN has the advantage of solving practical problems with unknown boundary conditions or initial conditions imposing on the boundary [24]. In addition, the well-trained PINN model with the ability of generalization can be regarded as a surrogate model, which greatly improves the computational efficacy and has the potential to realize the real-time prediction. To this end, PINN has attracted increasing attention ever since it was first proposed [25] and has been applied in the community of shape optimization [26], thermochemical curing [27], quantification of cracks [28], metal additive manufacturing [29], seismic response prediction [30, 31], nonlinear modeling [32], structural design [33], material defect identification [34], subsurface flow problems [35], engineering problems in heterogeneous domains [36] and etc. In addition, there are also a number of other

extensions in computational science, including solving PDE problems with optimized framework [37-45], surrogate modeling [46], multi-physics simulation framework [47], fluid mechanics [48-52], solid mechanics [53-56], transfer learning [57], computational elastic dynamics [58]. In general, the application of PINN could be divided into two categories, i.e., forward and inverse problems. The inverse problem aims to determine unknown physical parameters of the system from the known output by optimizing the prediction effect of data-driven NNs through learning physical information. The forward problem is to implement PINN as a solver of PDEs without labeled data, which can be regarded as an unsupervised learning approach by learning physical laws instead of labeled data. The related studies on forward problem mainly focus on optimizing the PINN framework according to the properties of PDEs [40, 42].

Compared with the conventional FEM approach that needs to discretize the computational domain with meshes and solve the structural response via the matrix manipulation, the PINN method only adapts randomly distributed training points rather than generated meshes, and achieves the structural response by forward solving the PDEs of the structure based on optimization principles. Specifically, in the structural mechanics computation, the establishment of PINN for solving bending structures is still a challenging task because it involves solving higher-order PDEs. The governing equations for bending structures are fourth-order nonlinear PDEs with multi-order differential terms. Although the automatic differentiation technique implemented in PINN enables obtaining the differential terms automatically, the

solution accuracy for higher-order differentiation by the automatic differentiation could be significantly decreased. More importantly, the multi-order differential terms included in PDEs lead to hard convergence or even non-convergence due to the huge difference of weights among differential terms [39].

In this study, to resolve the above-mentioned difficulty in structural mechanics computation, we design a novel multi-level physics-informed neural network (herein termed as m-PINN) framework to realize the intelligent computation of bending structures in which the governing equations are fourth-order nonlinear PDEs. By comparison with FEM computational results, we verified the accuracy of m-PINN for the solution of bending structures including the typical beam and shell structures. To the best of our knowledge, this is the first time to apply PINN as a new paradigm in the structural mechanics computation. The proposed method provides a new perspective for structure mechanics computation and could further serve as an intelligent computational module embedded in the structural DT system.

Results

Framework of m-PINN

The classical PINN methodology for solving high-order PDEs is to directly use the automatic differentiation technique to evaluate the high-order differential terms based on a single NN (Fig. 1). Nevertheless, using a single NN to process multi-order differentiations and multiple parameters will increase the computation burden, leading to large computation time and hard convergence.

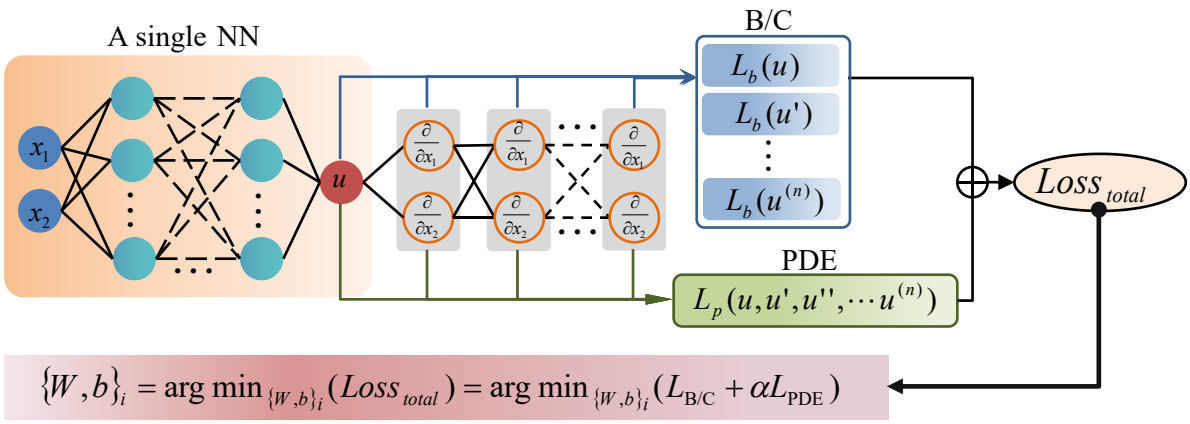


Fig. 1 | Classical PINN framework for solving high-order PDEs. $L_b(\ )$ and $L_p(\ )$ represent the loss functions from boundary constraints and loss functions from PDE, respectively. u represents differentiated physical variable and the output of the single NN.

To tackle this problem, we design a novel multi-physics-informed neural network (m-PINN). Two strategies have been proposed in the m-PINN: (i) a composite scheme of NNs is established by combining multiple single NNs, where each NN involves only first-order or second-order PDEs representing the geometrical, constitutive, and equilibrium relations of the structure, which effectively lowers down the order of the original higher-order PDEs; (ii) The loss functions corresponding to each NN with lower-order PDEs are linked and combined as one loss function with similar weights for each variables, and the loss function for boundary constraints can be directly defined in each NN rather through differential terms. These treatments enable the successful solution of bending structures. The specific process is described below.

The present m-PINN model is established to obtain the vertical displacements of bending structures, including the beam and shell structures. The beam structure can be regarded as a one-dimensional (1D) scenario, for which the computation framework of m-PINN is correspondingly presented in Fig. 2. The beam is subjected to a uniform distributed load q (Fig. 2a), and the vertical displacement ω, combining with section angle θ, curvature κ, bending moment M, and shear force F_Q, can be simultaneously obtained by the m-PINN model. The basic governing equation for the beam structure

is a fourth-order differential equation between q and ω, which can be transferred into five first-order equations in terms of the geometric, constitutive, and equilibrium relations. The five equations actually represent the three typical equations in structural mechanics, that is, geometric equation, stress-strain equation, and equilibrium equation with physical variables (i.e., θ, κ, M, and F_Q) as the transition parameters (see 'Methods' for details). Fig.1b illustrates the relationships between those physical variables as the transition from q to ω. Fig. 2c exhibits the framework of m-PINN to illustrate the computation process of beam structures. Firstly, three NNs are established to obtain the output of ω, κ, and M, which means that each parameter is trained separately within each independent NN. This strategy facilitates the convergence of training owing to reducing the complexity of NN for each parameter. For more details, see the section of 'Discussion'. Secondly, the derivatives of the NN output are obtained by utilizing the automatic differentiation technique, including $d\omega/dx$, $d\theta/dx$, $d\kappa/dx$, dF_Q/dx. Lastly, the multiple NNs are connected by the loss functions, and a general loss function is formed including both the loss functions from differential equations, L_p, and those from the boundary constraints, L_b. During the iterative computation process, the updated parameters of multiple NNs are oriented to minimize the loss function, and the training is completed when the loss function converges to stability.

For the two-dimensional (2D) scenario of shell structures, the computation framework of m-PINN is presented in Fig. 3. The approach to reduce the PDE order of shell structures is similar to that of beam structures, except that the decomposed lower-order PDEs (i.e., the three typical equations) involve second-order differential terms such as the relation between $\omega_{NN}(x, y)$ and $\kappa_{NN}(x, y)$ as well as the relation between $M_{NN}(x, y)$ and $q(x, y)$ (see Fig .3b). Moreover, the framework for shell

156 structure is more complicated (Fig. 3c) due to the increase of dimension, the existence
157 of second-order partial derivatives, and more boundary constraints. Hence, the
158 training for the framework of shell structures is more time-consuming, which needs
159 more iteration steps to achieve convergence.

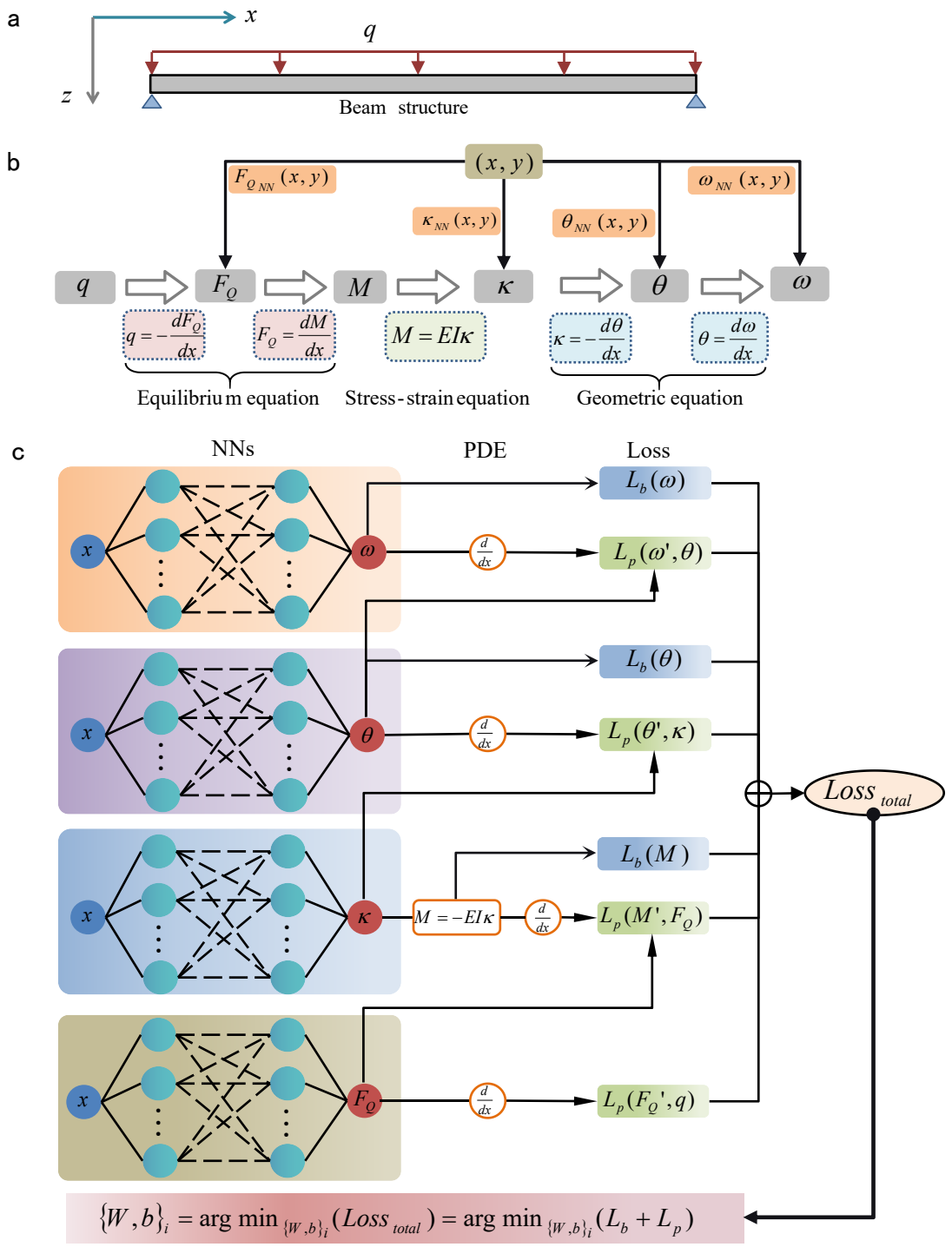


Fig. 2 | Multi-physics-informed neural network (m-PINN) for 1D beam. a Diagrammatic sketch of beam structure. M represents bending moment. **b** Physical relationship between the relevant variables that transforms the uniform load q to the vertical displacement ω by utilizing the three typical equations. The intermediate variables including F_Q, κ, θ, and ω are outputted from each neural network. **c** Schematics of m-PINN framework for beam structure. $L_p(\omega', \theta)$, $L_p(\theta', \kappa)$, $L_p(M', F_Q)$ and $L_p(F_Q', q)$ are loss functions corresponding to each differential equation. $L_b(\omega)$, $L_b(\theta)$ and $L_b(M)$ are the loss functions from boundary constraints corresponding to the vertical displacement, section turning angle and bending moment, respectively.

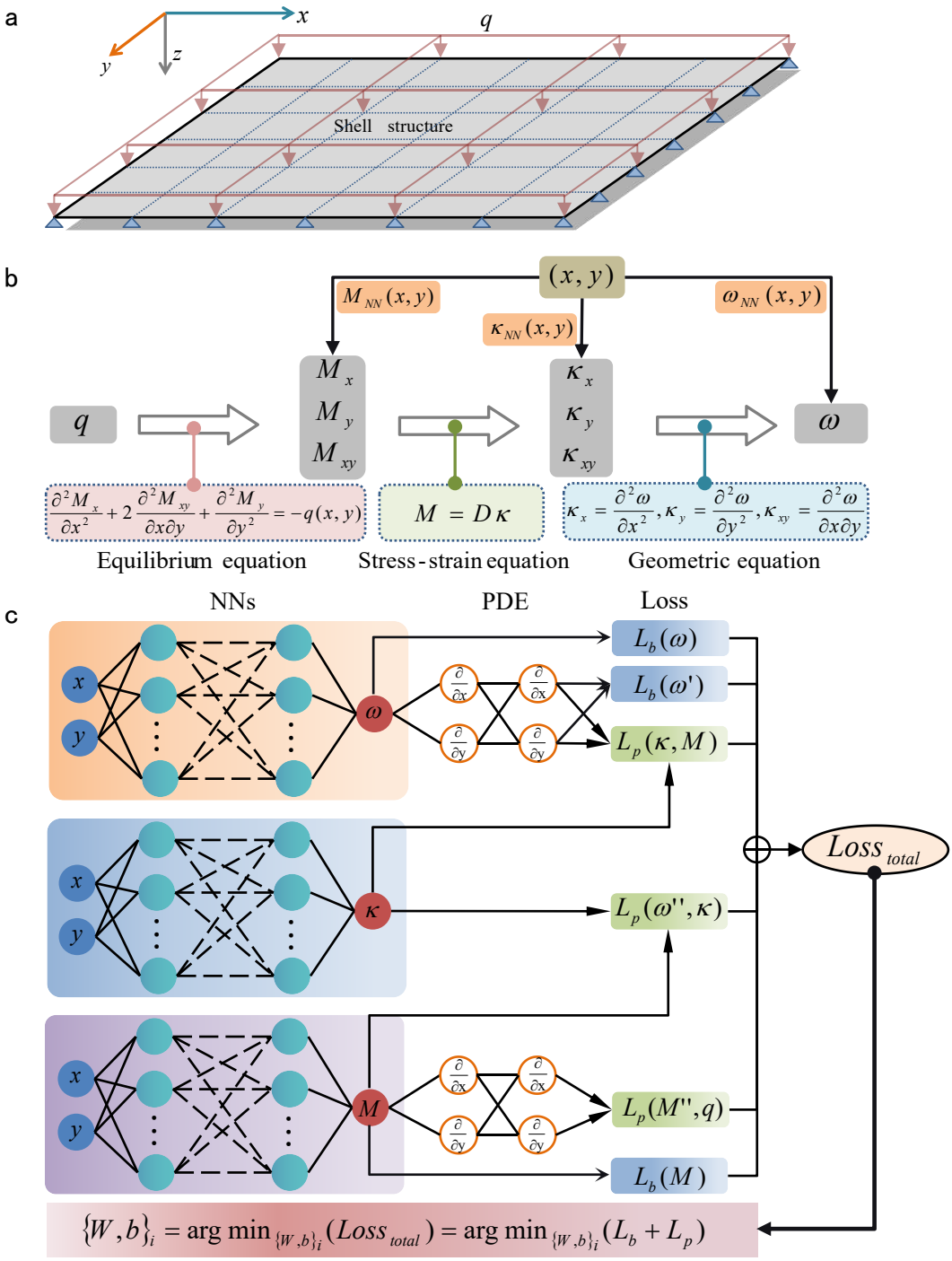


Fig. 3 | Multi-physics-informed neural network (m-PINN) for 2D shell. a Diagrammatic sketch of shell structures. M_x and M_y represent the bending moment of shell structures in x-direction and y-direction respectively. M_{xy} denotes the torque in the z-direction. The vertical displacement is in the z-direction. **b** Physical relationship between the relevant variables that transforms the uniform load q to the vertical displacement ω by utilizing the three typical equations. The intermediate variables including M, κ, and ω are outputted from each neural network. **c** Schematics of m-PINN framework for shell structures. $L_p(\omega'', \kappa)$, $L_p(\kappa, M)$ and $L_p(M'', q)$ are the loss functions corresponding to PDEs. $L_b(\omega)$, $L_b(\omega')$ and $L_b(M)$ are the loss functions of boundary constraints from the vertical displacement, section turning angle and bending moment, respectively.

Computational results of bending structures

We first consider the computation of 1D beam structures under four different loading conditions, and the m-PINN computed results of beam deformation were compared with the corresponding FEM results. As previously mentioned, the arbitrarily distributed points, termed as collocation points, are required to train the NN. A total number of 25 collocation points that are randomly distributed along the beam structure are used for training (Fig .4a). Regarding the influence of point number on the computational results, see the forthcoming sensitivity analysis.

Table.1 The summary of loading conditions for beam structure

Load and constraints	case 1		case 2		case 3		case 4	
	$x/l = 0$	$x/l = 1$	$x/l = 0$	$x/l = 1$	$x/l = 0$	$x/l = 1$	$x/l = 0$	$x/l = 1$
ω	0	0	0	0	0	1	0	0
θ	/	/	0	0	0	0	0	1
q	-1		-1		0		0	

Notes: ω and θ represent the constraints of displacement and rotational angle imposing on two ends of the beam (i.e., $x/l = 0$ and $x/l = 1$ where l represents the beam length). q represents the vertical uniform force acting on the beam. Note that the values provided in the table are nondimensionalized

The loads exerting on the beam could be divided into external force (i.e., the uniform load q) and boundary constraints (i.e., the displacement ω and rotational angle θ imposing on two ends of the beam). The detailed conditions of different loading cases are listed in Table 1. Figs. 4b~4e exhibit the comparison of deformation results between m-PINN and FEM under four different loading conditions. It shows that the m-PINN computed results of vertical displacement agree quite well with the FEM computed results. The profiles of beam deformation from both m-PINN and

FEM are almost overlapped (relative error < 0.5%), which indicates that the m-PINN shares the same accuracy with FEM in terms of bending beam computation.

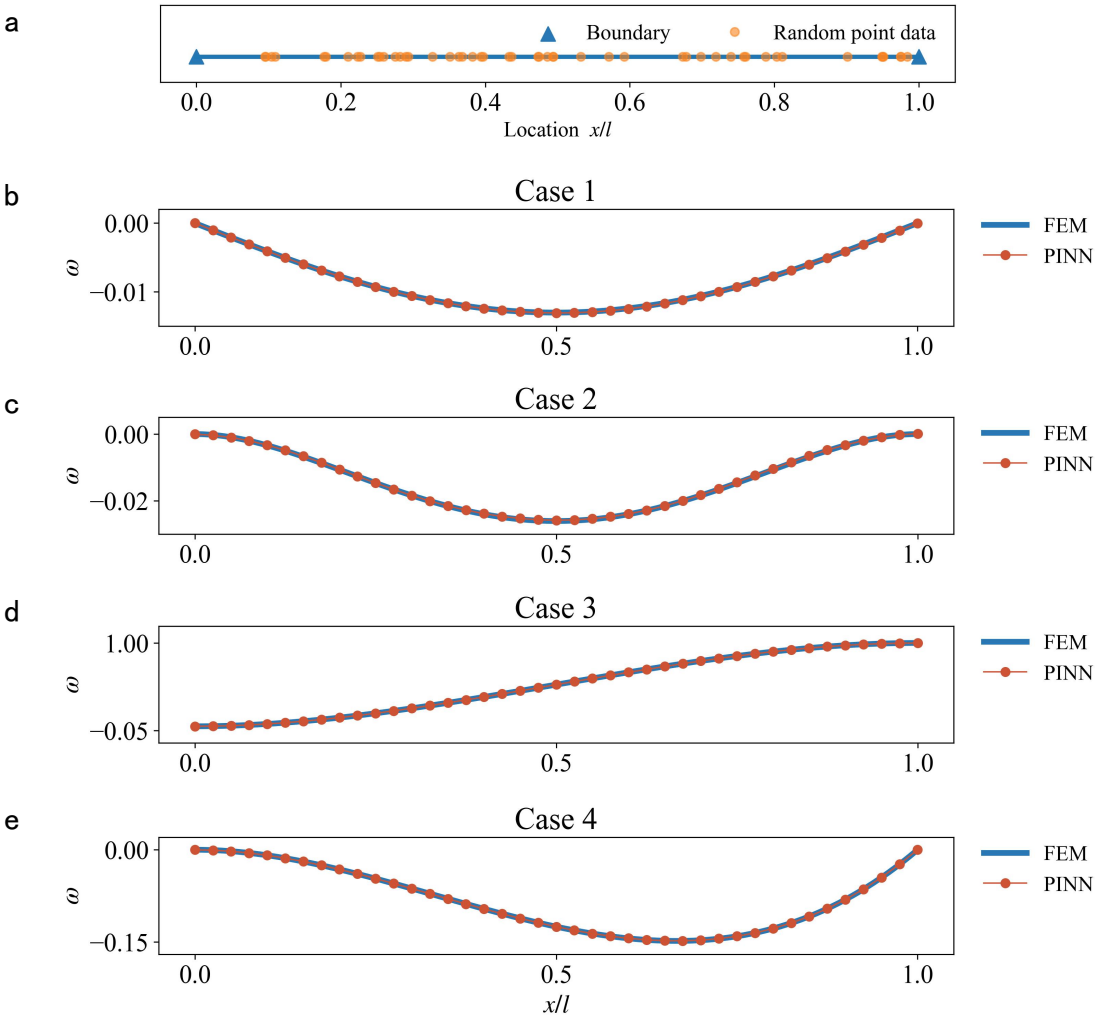


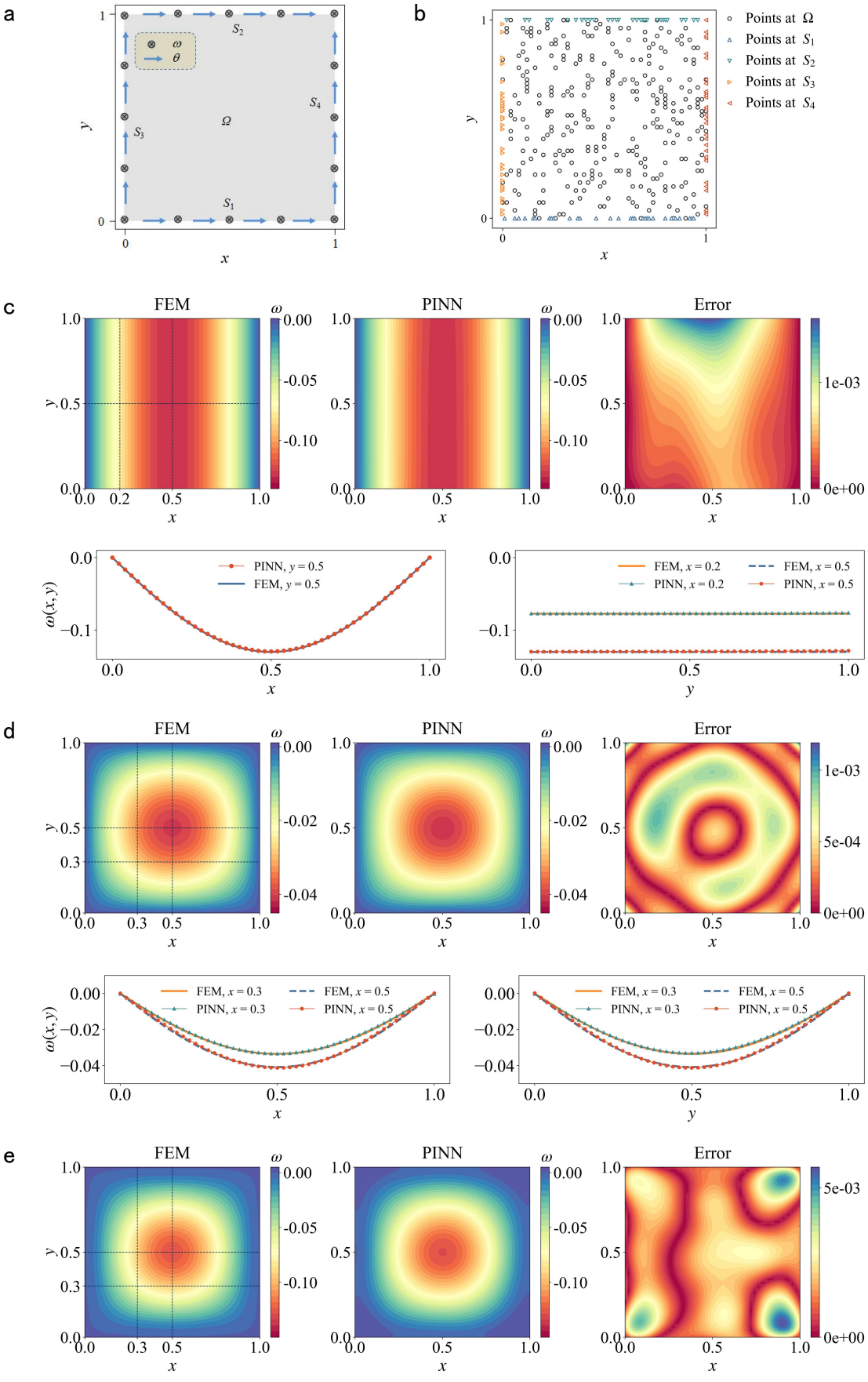
Fig. 4 | Computational results for 1D beam structure. a Randomly distributed training data for beam structures. **b** Verification for case 1 (simply supported beam). The boundary displacement constraint is given by $\omega|_{x/l=0,x/l=1} = 0$. The vertical uniform load q is set to -1. **c** Verification for case 2 (double-clamped beam). The displacement and turning angle at the ends of the beam are imposed with $\omega|_{x/l=0,x/l=1} = 0$ and $\theta|_{x/l=0,x/l=1} = 0$. The vertical uniform load q is -1. **d** Verification for case 3. The turning angles at the beam ends are constrained, namely, $\theta|_{x/l=0,x/l=1} = 0$, but the constraints of displacements are separately given by $\omega|_{x/l=0} = 0$ and $\omega|_{x/l=1} = 1$. **e** Verification for case 4. The constraints of displacements and turning angles are defined as $\omega|_{x/l=0,x/l=1} = 0$ and $\theta|_{x/l=0} = 0$ and $\theta|_{x/l=1} = 1$, respectively.

We further applied the m-PINN to solve the 2D shell. Table 2 lists the four cases corresponding to various loading conditions exerting on shell. A total number of 1000 data points are used and randomly distributed at the interior zone as well as 50 data points at each boundary (S1~S4) (see Figs. 5a and 5b). The comparison indicates that the computed results by m-PINN and FEM are very close (Figs. 5c~5f). The error contours demonstrate that the deviation between m-PINN and FEM at the corner region of the shell is more obvious than that at the interior region, especially for case 3 and case 4 (Figs. 5e and 5f). The primary reason is that more constraints imposing at the corners lead to the convergence of computation at the corner regions more difficult. Nevertheless, the error is generally low with the relative error less than 2%, which means that the proposed m-PINN model is capable of accurately solving the 2D structural mechanics problem specifically the deformation of bending shell under various loading conditions.

Table. 2 The summary of loading conditions for shell structure

Load and constraints	case 1		case 2		case 3		case 4	
	ω	θ_n	ω	θ_n	ω	θ_n	ω	θ_n
Boundary S_1	/	/	/	/	0	/	0	/
Boundary S_2	0	/	0	/	0	/	0	/
Boundary S_3	0	0	0	0	0	0	0	0
Boundary S_4	0	1	0	1	0	1	0	1

Notes: ω and θ_n represent the displacement and normal angle at the boundaries. Boundaries S_1~S_4 are defined in Fig. 4a.


a
b
c
d
e
S_1
S_2
S_3
S_4
Ω
ω
θ
Points at Ω
Points at S_1
Points at S_2
Points at S_3
Points at S_4
FEM
PINN
Error
PINN, $y = 0.5$
FEM, $y = 0.5$
FEM, $x = 0.2$
PINN, $x = 0.2$
FEM, $x = 0.5$
PINN, $x = 0.5$
FEM, $x = 0.3$
PINN, $x = 0.3$
$\omega(x, y)$

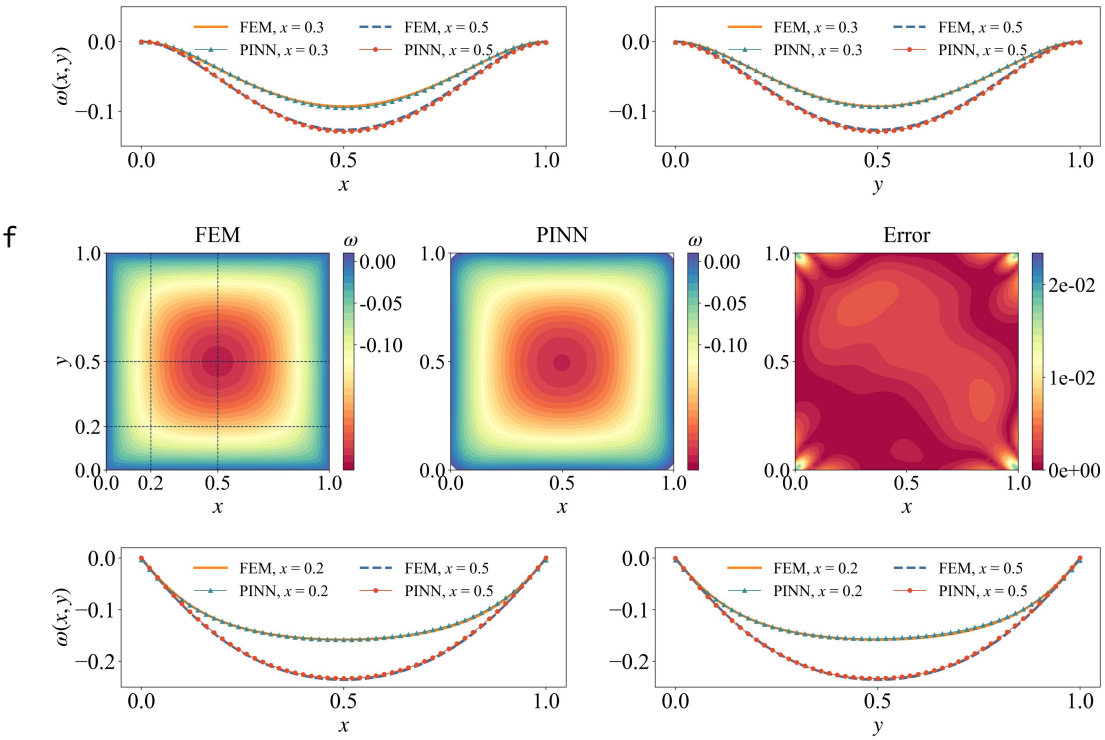
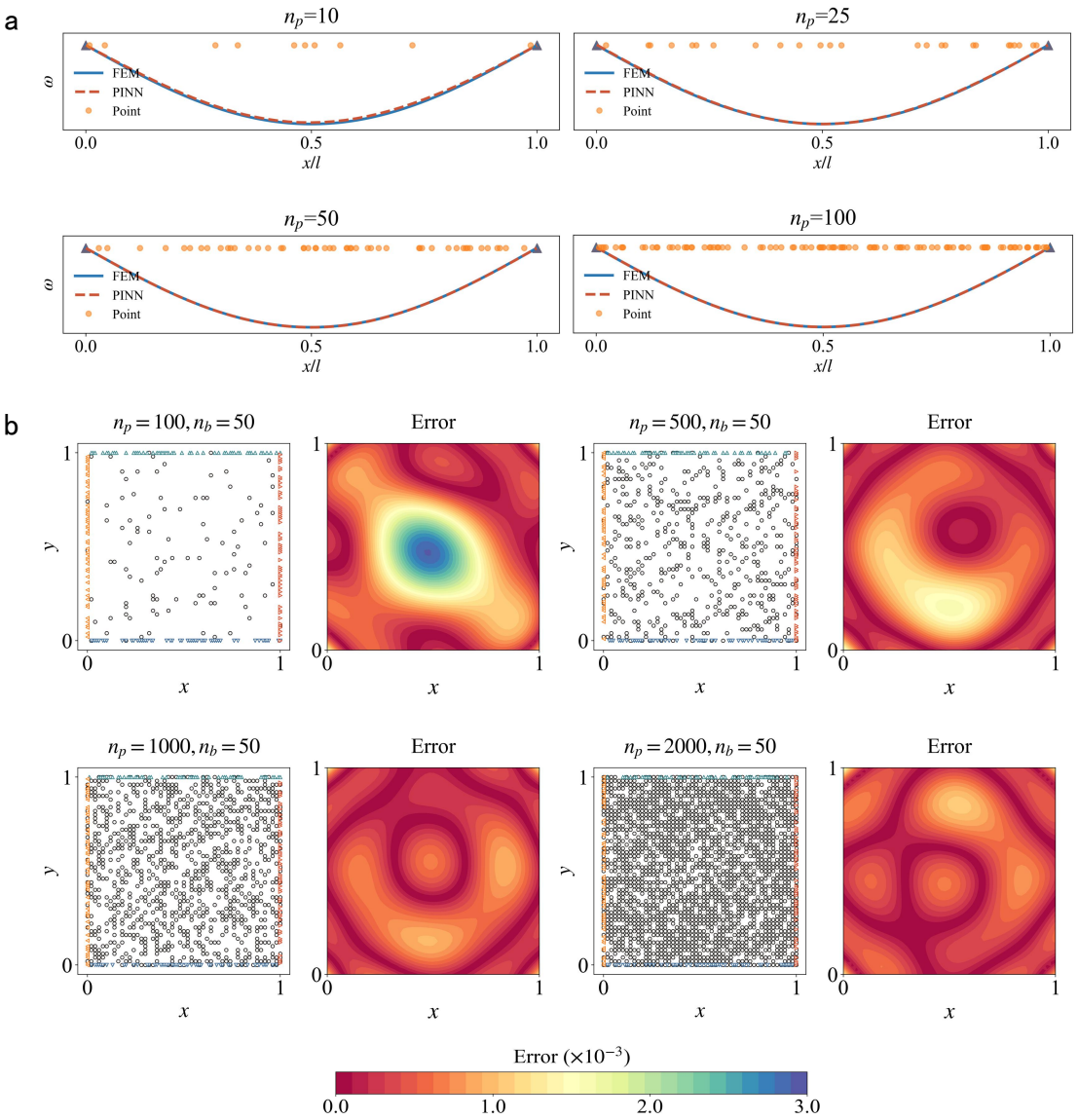


Fig. 5 | Computational results for 2D shell structure. a Diagram of boundaries. ω and θ represent the displacement and rotation angle imposing on the boundaries, respectively. Note that the direction of θ is indicated as blue arrow according to right-handed screw rule, and the direction of ω is in z-direction. **b** Distribution of collocation points. The random data points include points in the interior zone Ω and points on the boundaries S_1~S_4. **c** Verification for case 1. Only the opposite pair of boundary displacements is constrained by $\omega|_{S \in S3,\ S4} = 0$. Note that the three contour plots at the top row from left to right represent the PINN predictions, FEM predictions and the relative error, respectively. The two plots at the bottom present the extracted results of deformation in x-directions and y-directions respectively (as illustrated by the dotted line in the contour of FEM). **d** Verification for case 2. All surrounding boundary displacements are defined as $\omega|_{S \in S1,\ S2,\ S3,\ S4} = 0$. **e** Verification for case 3. The displacement $\omega|_{S \in S1,\ S2,\ S3,\ S4} = 0$ and section corner $\theta|_{S \in S1,\ S2,\ S3,\ S4} = 0$ are imposed on all surrounding boundaries. **f** Verification for case 4. The displacement $\omega|_{S \in S1,\ S2,\ S3,\ S4} = 0$ and section corner $\theta|_{S \in S1,\ S2,\ S3,\ S4} = 1$ are defined, and the uniform load is given by $q = -1$.

Sensitivity analysis of collocation points

For the m-PINN computation, the number of collocation points is critical affecting the accuracy of the computational results. Hence, a sensitivity analysis on the collocation points is required. Fig. 6 shows the comparison of m-PINN predicted results using different number of collocation points (n_p). For the beam structure case, we tested 10, 25, 50 and 100 randomly distributed points. For the shell structure case, we used 10, 100, 500 and 1000 randomly distributed points in the interior area while keeping 50 data points fixed at each boundary of the shell structure.

For the 1D beam (Fig. 6a), the computational error is relatively high (close to 3%) when rather sparse collocation points are used (n_p = 10), while the error is reduced less than 1% as the number of collocation points increases (n_p > 25). Thus, to balance the computational precision and speed, the number of points n_p = 25 has been adopted. For the 2D shell computation (Fig. 6b), Table 3 lists the quantitative comparison of loss residuals and relative errors corresponding to different number of collocation points. It seems that a number of densely distributed points are required to improve the accuracy, and the error is reduced to approximately 2% when n_p = 1000. Thus, considering both the computational accuracy and efficacy, we used 1000 collocation points to compute the shell structure, as shown in Fig. 4.



257 **Fig. 6 | Comparison of m-PINN prediction using different number of collocation points. a**
258 Beam structure case with different number of collocation points. The number of data points is
259 selected as 10, 25, 50 and 100. **b** Shell structure case with different number of collocation points.
260 n_p represents the number of data point at the interior zone, which is given by 100, 500, 1000 and
261 2000. n_b denotes the number of data point at the boundaries.

Table 3 Loss residuals and errors of prediction using different number of collocation points

Type	Collocation points	L_p / n_p	L_b / n_b	Error
Beam	n_p = 10	1.077×10^{-6}	8.426×10^{-8}	(2.50±1.5)%
	n_p = 25	1.760×10^{-7}	3.749×10^{-9}	(0.68±0.3%)
	n_p = 50	2.955×10^{-8}	9.299×10^{-10}	(0.36±0.2%)
	n_p = 100	6.123×10^{-9}	4.170×10^{-13}	(0.068±0.03%)
Shell	n_p = 100; n_b = 50	9.005×10^{-5}	9.255×10^{-6}	(9.01±5)%
	n_p = 500; n_b = 50	1.905×10^{-5}	8.008×10^{-6}	(2.94±1)%
	n_p = 1000; n_b = 50	8.760×10^{-6}	8.660×10^{-6}	(2.03±0.8)%
	n_p = 2000; n_b = 50	4.691×10^{-6}	9.521×10^{-6}	(1.59±0.6)%

Notes: n_p and n_b denote the number of collocation points at the interior zone and the boundaries, respectively. Error= $\frac{\left\|\omega_{PINN}-\omega_{FEM}\right\|_2^2}{\left\|\omega_{FEM}\right\|_2^2}$ where ω_{PINN} and ω_{FEM} represent the value of the vertical displacement computed by m-PINN and FEM respectively.

Discussion

In summary, this study proposes a m-PINN framework for structural mechanics computation with high-order PDEs, aiming to predict the response of bending beam and shell structures under different loading conditions. The PDEs of bending structures involve multiple physical parameters such as vertical displacement ω, section angles θ, bending moment M, and those parameters are correlated via differentiation (e.g., $\theta\propto\partial\omega$, $\kappa\propto\partial^2\omega$, and $q\propto\partial^4\omega$, see "Methods"), and these differential terms are imposed by the boundary constraints. Hence, multiple loss functions with different order of magnitudes corresponding to each differential term exist in the classical PINN framework. The disparity in the magnitudes between these loss

functions is substantial due to different orders of differentiation, which results in non-convergence or only local convergence for the NN of ω. However, the proposed m-PINN adopts multiple NNs for the output of each intermediate physical parameter such as θ, κ, and M to achieve both the local and global convergence, meanwhile reducing the order of differential terms through the transition of the intermediate parameters (e.g., $q \propto \partial F_Q$, $F_Q \propto \partial M$, $\kappa \propto \partial \theta$ and $\theta \propto \partial \omega$ in beam as well as, $q \propto \partial^2 M$, $\kappa \propto \partial \theta$ and $\theta \propto \partial \omega$ in shell). In other words, the advantage of the m-PINN to solve the high-order PDEs lies in the underlying physical relationships, i.e., the three typical equations describing the correlation between these physical variables. This strategy of reducing PDEs order using m-PINN can be also extended to other solid mechanics issues with governing PDEs having similar properties.

Currently, the major disadvantage of m-PINN framework is that the training process is time-consuming, and this is a general problem yielded in PINN for solving the forward problem. However, compared to the FEM requiring the repeated model establishment (e.g., mesh generation, parameters input and calibration), the m-PINN with the ability of generalization learning has the potential to quickly predict the results for different loading cases once the NNs are trained well. Overall, the aforementioned results and discussion indicate that the m-PINN framework achieves the same accuracy of FEM for solving bending structures. The trained m-PINN framework is similar to a complex function that can be feasibly implemented in the DT system to realize the accurate and efficient real-time prediction of the structural response.

Methods

Differential equations of beam structures

The governing equation describing the deformation of bending beam is a fourth-order differential equation, given by

$$\frac{d^4\omega}{dx^4} = \frac{q(x)}{EI} \tag{1}$$

where $q(x)$ is the uniform load on the beam, EI represents flexural stiffness, and ω is the vertical displacement. To reduce the order of Eq. (1), several intermediate variables are introduced, and Eq. (1) can be decomposed into five first-order differential equations:

$$q = -\frac{dF_Q}{dx} \tag{2-1}$$

$$F_Q = \frac{dM}{dx} \tag{2-2}$$

$$M = -EI\kappa \tag{2-3}$$

$$\kappa = -\frac{d\theta}{dx} \tag{2-4}$$

$$\theta = \frac{d\omega}{dx} \tag{2-5}$$

where θ and κ denote the section angle and curvature of the beam respectively. M and F_Q represent bending moment and shear force, respectively. In structural mechanics, the aforementioned equations are normally termed as equilibrium equation (Eqs. 2-1 and 2-2), stress-strain equation (Eq. 2-3), and geometric equation (Eqs. 2-4 and 2-5).

There are two types of boundary constraints: (i) the vertical displacement $\overline{\omega}$ and section angle $\overline{\theta}$ are defined on the boundary s_{t1} as,

$$\omega(x)=\overline{\omega}\ \cap\ \theta(x)=\overline{\theta},\quad x\in s_{t1} \tag{3}$$

Specially, $\overline{\omega}=0$ and $\overline{\theta}=0$ means the endpoint consolidation; (ii) the vertical displacement $\overline{\omega}$ and bending moment $\overline{M}$ are imposed on the boundary s_{t2} as,

$$\omega(x)=\overline{\omega}\ \cap\ M(x)=\overline{M},\quad x\in s_{t2} \tag{4}$$

For simply supported beam, the boundary condition is specified as $\overline{\omega}=0$ and $\overline{M}=0$.

Differential equations of shell structures

Similar to the governing equations of beam structure described above, the governing PDEs for the shell structures also include the geometric equation, deformation equation and equilibrium equation, which are given by,

$$\kappa_x=-\frac{\partial^2\omega}{\partial x^2},\quad \kappa_{xy}=-2\frac{\partial^2\omega}{\partial x\partial y},\quad \kappa_y=-\frac{\partial^2\omega}{\partial y^2} \tag{5}$$

$$M=\begin{bmatrix}M_x\\M_y\\M_{xy}\end{bmatrix}=D\kappa=\frac{Et^3}{12(1-\nu^2)}\begin{bmatrix}1&\nu&0\\\nu&1&0\\0&0&(1-\nu)/2\end{bmatrix}\begin{bmatrix}\kappa_x\\\kappa_y\\\kappa_{xy}\end{bmatrix} \tag{6}$$

$$\frac{\partial^2 M_x}{\partial x^2}+2\frac{\partial^2 M_{xy}}{\partial x\partial y}+\frac{\partial^2 M_y}{\partial y^2}+q(x,y)=0 \tag{7}$$

where ω is the displacement in the z-direction. κ_x, κ_y and κ_{xy} represent the curvature of the shell structures in the x-direction, y-direction and the torsion rate in the z-direction, respectively. Correspondingly, M_x, M_y and M_{xy} denote the bending moment of the shell structures in the x-direction, y-direction and the torque in the z-direction, respectively. D, E, t, v denote the elastic relationship matrix, elastic modulus, shell thickness in the z-direction, and Poisson ratio of the material, respectively. $q(x, y)$ is the uniformly distributed load exerting on the shell, which is a known quantity.

Similarly, two types of boundary constraints are defined on the two boundaries S_{t1}

and S_{t2} respectively, which are expressed as

$$\omega(x,y)=\bar{\omega} \ \cap \ \frac{\partial\omega}{\partial n}(x,y)=\bar{\theta}, \quad (x,y)\in S_{t1} \tag{8}$$

$$\omega(x,y)=\bar{\omega} \ \cap \ M_n(x,y)=\overline{M_n}, \quad (x,y)\in S_{t2} \tag{9}$$

where $\bar{\omega}$, $\bar{\theta}$ and $\overline{M}$ represent vertical displacement, section angle and bending moment, n denotes the normal direction of the boundary.

Loss function of m-PINN

The loss function of ml-PINN is defined as the residuals of PDEs and the boundary constrains. For the beam structure, the loss function of the PDE, L_p, is composed of residuals from geometric equation ($L_p(\omega', \theta)$ and $L_p(\theta', \kappa)$) as well as equilibrium equation ($L_p(M', F_Q)$ and $L_p(F_Q', q)$), as shown below. Note that the loss function from stress-strain equation is not established due to its simple linearity between κ and M.

$$L_p(\omega',\theta)=\sum_{i=1}^{n_p}\left\|\frac{d\omega_{NN}}{dx}_i-\theta_{NN}(x,y)_i\right\|_2^2, \quad x\in l \tag{10}$$

$$L_p(\theta',\kappa)=\sum_{i=1}^{n_p}\left\|\frac{d\theta_{NN}}{dx}_i-\kappa_{NN}(x)_i\right\|_2^2, \quad x\in l \tag{11}$$

$$L_p(M',F_Q)=\sum_{i=1}^{n_p}\left\|\frac{dM_{NN}}{dx}_i-F_{QNN}(x)_i\right\|_2^2, \quad x\in l \tag{12}$$

$$L_p(F_Q',q)=\sum_{i=1}^{n_p}\left\|\frac{dF_{QNN}}{dx}_i-q(x)_i\right\|_2^2, \quad x\in l \tag{13}$$

where n_p is the number of collocation points in whole regions, l represents the interior zone of the computational domain in the beam structure. $\sum\|\cdot\|_2^2$ is the quadratic norm. $\frac{d\omega_{NN}}{dx}$, $\frac{d\theta_{NN}}{dx}$, $\frac{dM_{NN}}{dx}$ and $\frac{dF_{QNN}}{dx}$ represent first-order derivative of NNs

output of ω, θ, M and F_Q, respectively.

The loss function of boundary constraints L_b includes loss function of constrained vertical displacement $L_b(\omega)$, loss function of constrained section corner $L_b(\theta)$, and loss function of constrained bending moment $L_b(M)$, namely,

$$L_b(\omega) = \sum_{i=1}^{n_b} \left\| \omega_{NN}(x)_i - \overline{\omega(x)_i} \right\|_2^2, \quad x \in l_b \tag{14}$$

$$L_b(\theta) = \sum_{i=1}^{n_b} \left\| \theta_{NN}(x)_i - \overline{\theta(x)_i} \right\|_2^2, \quad x \in l_b \tag{15}$$

$$L_b(M) = \sum_{i=1}^{n_b} \left\| M_{NN}(x)_i - \overline{M(x)_i} \right\|_2^2, \quad x \in l_b \tag{16}$$

where n_b is the number of collocation points in boundary regions. l_b represents boundary regions in the beam structure. $\overline{\omega(x)}$, $\overline{\theta(x)}$ and $\overline{M(x)}$ represent the displacement, rotation angle, and bending moment at the boundary, respectively.

The total loss function L_{total} of the beam structure is added by L_p and L_b, namely

$$L_{total} = \left[L_b(\omega) + L_b(\theta) + L_b(M) \right] + \left[L_p(\omega', \theta) + L_p(\theta', \kappa) + L_p(M', F_Q) \right] + L_p(F_Q', q) \tag{17}$$

Similarly, for the shell structures, the loss function of the PDE, L_p, is composed by residuals from geometric equation $L_p(\omega'', \kappa)$, stress-strain equation $L_p(\kappa, M)$ and equilibrium equation $L_p(M'', q)$, which are written as,

$$L_p(\omega'', \kappa) = \sum_{i=1}^{n_p} \left\| [\frac{\partial^2 \omega_{NN}}{\partial x^2}, \frac{\partial^2 \omega_{NN}}{\partial y^2}, 2\frac{\partial^2 \omega_{NN}}{\partial x \partial y}]^T{}_i - \kappa_{NN}(x,y)_i \right\|_2^2, \quad (x,y) \in \Omega \tag{18}$$

$$L_p(\kappa, M) = \sum_{i=1}^{n_p} \left\| D\kappa_{NN}(x,y)_i - M_{NN}(x,y)_i \right\|_2^2, \quad (x,y) \in \Omega \tag{19}$$

$$L_p(M'', q) = \sum_{i=1}^{n_p} \left\| (\frac{\partial^2 M_{NNx}}{\partial x^2} + 2\frac{\partial^2 M_{NNxy}}{\partial x \partial y} + \frac{\partial^2 M_{NNy}}{\partial y^2})_i + q(x,y)_i \right\|_2^2, \quad (x,y) \in \Omega \tag{20}$$

where Ω represents the interior zone of the computational domain in the shell structure. $\frac{\partial^2 \omega_{NN}}{\partial x^2}$, $\frac{\partial^2 \omega_{NN}}{\partial y^2}$ and $\frac{\partial^2 \omega_{NN}}{\partial x \partial y}$ represent the second-order partial derivative of the NN output of displacement $\omega_{NN}(x, y)$, $\kappa_{NN}(x, y)$ is the NN output of curvature κ, $\frac{\partial^2 M_{NNx}}{\partial x^2}$, $\frac{\partial^2 M_{NNy}}{\partial y^2}$ and $\frac{\partial^2 M_{NNxy}}{\partial x \partial y}$ denote the second-order partial derivative of the NN output of bending moment $M_{NN}(x, y)$.

The loss functions of boundary constraints mainly involve loss function of constrained vertical displacement $L_b(\omega)$, loss function of constrained section corner $L_b(\omega')$, and loss function of constrained bending moment $L_b(M_n)$, namely,

$$L_b(\omega) = \sum_{i=1}^{n_b} \left\| \omega(x,y)_i - \overline{\omega(x,y)_i} \right\|_2^2, \quad (x,y) \in S_1, S_2, S_3, S_4 \tag{21}$$

$$L_b(\omega') = \sum_{i=1}^{n_b} \left\| \frac{\partial \omega(x,y)_i}{\partial n} - \overline{\theta(x,y)_i} \right\|_2^2, \quad (x,y) \in S_1, S_2, S_3, S_4 \tag{22}$$

$$L_b(M_n) = \sum_{i=1}^{n_b} \left\| M_n(x,y)_i - \overline{M_n(x,y)_i} \right\|_2^2, \quad (x,y) \in S_1, S_2, S_3, S_4 \tag{23}$$

where S_1, S_2, S_3 and S_4 represent boundary regions in the shell structure (Fig. 4b). $\overline{\omega(x,y)}$, $\overline{\theta(x,y)}$ and $\overline{M(x,y)}$ represent the displacement, rotation angle, and bending moment at the boundary, respectively.

The total loss function L_{total} of the shell structure is written as,

$$L_{total} = \left[L_b(\omega) + L_b(\omega') + L_b(M) \right] + \left[L_p(\omega'', \kappa) + L_p(\kappa, M) + L_p(M'', q) \right] \tag{24}$$

The NNs are automatically optimized to update the weights and biases of the network by minimizing the loss function, namely,

$$\{W, b\}^*_i = \arg\min_{\{W,b\}_i} \left[Loss_{total}(\{W, b\}) \right] \tag{25}$$

Data availability

The simulated data is provided by software ABAQUS, and is used for verification of computational results predicted by the m-PINN framework. The data is available and is bundled with code. Source data are provided with this paper.

Code availability

Source programs of m-PINN computation are available in Supplementary code. The m-PINN framework runs on the open-source Python package, Tensorflow 1.15.

Competing interests

The authors declare no competing interests.

References

1. Jiang, Y., Yin, S., Li, K., Luo, H., & Kaynak, O. (2021). Industrial applications of digital twins. Philosophical Transactions of the Royal Society A, 379(2207), 20200360.
2. Austin, M., Delgoshaei, P., Coelho, M., & Heidarinejad, M. (2020). Architecting smart city digital twins: Combined semantic model and machine learning approach. Journal of Management in Engineering, 36(4), 04020026.
3. Li, X., Liu, H., Wang, W., Zheng, Y., Lv, H., & Lv, Z. (2022). Big data analysis of the internet of things in the digital twins of smart city based on deep learning. Future Generation Computer Systems, 128, 167-177.
4. Jiang, F., Ma, L., Broyd, T., & Chen, K. (2021). Digital twin and its implementations in the civil engineering sector. Automation in Construction, 130, 103838.
5. Bado, M. F., Tonelli, D., Poli, F., Zonta, D., & Casas, J. R. (2022). Digital twin for civil

engineering systems: an exploratory review for distributed sensing updating. Sensors, 22(9), 3168.

6. Wagg, D. J., Worden, K., Barthorpe, R. J., & Gardner, P. (2020). Digital twins: state-of-the-art and future directions for modeling and simulation in engineering dynamics applications. ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg, 6(3).

7. Dimiduk, D. M., Holm, E. A., & Niezgoda, S. R. (2018). Perspectives on the impact of machine learning, deep learning, and artificial intelligence on materials, processes, and structures engineering. Integrating Materials and Manufacturing Innovation, 7, 157-172.

8. Salehi, H., & Burgueño, R. (2018). Emerging artificial intelligence methods in structural engineering. Engineering Structures, 171, 170-189.

9. Sun, L., Shang, Z., Xia, Y., Bhowmick, S., & Nagarajaiah, S. (2020). Review of bridge structural health monitoring aided by big data and artificial intelligence: From condition assessment to damage detection. Journal of Structural Engineering, 146(5), 04020073.

10. Tapeh, A. T. G., & Naser, M. Z. (2022). Artificial Intelligence, Machine Learning, and Deep Learning in Structural Engineering: A Scientometrics Review of Trends and Best Practices. Archives of Computational Methods in Engineering, 1-45.

11. Baduge, S. K., Thilakarathna, S., Perera, J. S., Arashpour, M., Sharafi, P., Teodosio, B., ... & Mendis, P. (2022). Artificial intelligence and smart vision for building and construction 4.0: Machine and deep learning methods and applications. Automation in Construction, 141, 104440.

12. Cha, Y. J., Choi, W., & Büyüköztürk, O. (2017). Deep learning-based crack damage

detection using convolutional neural networks. Computer-Aided Civil and Infrastructure Engineering, 32(5), 361-378.

13. Dorafshan, S., Thomas, R. J., & Maguire, M. (2018). Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. Construction and Building Materials, 186, 1031-1045.

14. Bao, Y., Tang, Z., Li, H., & Zhang, Y. (2019). Computer vision and deep learning‐based data anomaly detection method for structural health monitoring. Structural Health Monitoring, 18(2), 401-421.

15. Zhu, Z., & Brilakis, I. (2010). Parameter optimization for automated concrete detection in image data. Automation in Construction, 19(7), 944-953.

16. Sharafi, P., Teh, L. H., & Hadi, M. N. (2014). Shape optimization of thin-walled steel sections using graph theory and ACO algorithm. Journal of Constructional Steel Research, 101, 331-341.

17. Rahman, J., Ahmed, K. S., Khan, N. I., Islam, K., & Mangalathu, S. (2021). Data-driven shear strength prediction of steel fiber reinforced concrete beams using machine learning approach. Engineering Structures, 233, 111743.

18. Kwon, S. J., & Song, H. W. (2010). Analysis of carbonation behavior in concrete using neural network algorithm and carbonation modeling. Cement and Concrete Research, 40(1), 119-127.

19. Hossain, M., Gopisetti, L. S. P., & Miah, M. S. (2020). Artificial neural network modelling to predict international roughness index of rigid pavements. International journal of pavement research and technology, 13, 229-239.

20. Nilsen, V., Pham, L. T., Hibbard, M., Klager, A., Cramer, S. M., & Morgan, D. (2019). Prediction of concrete coefficient of thermal expansion and other properties using machine learning. Construction and Building Materials, 220, 587-595.
21. Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. (2017). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. arXiv preprint arXiv:1711.10561.
22. 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.
23. Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. Journal of Marchine Learning Research, 18, 1-43.
24. Karniadakis,Meng, Chuizheng, et al. (2022). When physics meets machine learning: A survey of physics-informed machine learning. arXiv preprint arXiv:2203.16797 .
25. G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. Nature Reviews Physics, 3(6), 422-440.
26. Liu, S., Kappes, B. B., Amin-ahmadi, B., Benafan, O., Zhang, X., & Stebner, A. P. (2021). Physics-informed machine learning for composition-process-property design: Shape memory alloy demonstration. Applied Materials Today, 22, 100898.
27. S.A. Niaki, E. Haghighat, T. Campbell, A. Poursartip, R. Vaziri. (2020). Physics-informed neural network for modelling the thermochemical curing process of

composite-tool systems during manufacture, Computer Methods in Applied Mechanics and Engineering. 384, 113959.

28. Shukla, K., Di Leoni, P. C., Blackshire, J., Sparkman, D., & Karniadakis, G. E. (2020). Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks. Journal of Nondestructive Evaluation, 39, 1-20.

29. Zhu, Q., Liu, Z., & Yan, J. (2021). Machine learning for metal additive manufacturing: predicting temperature and melt pool fluid dynamics using physics-informed neural networks. Computational Mechanics, 67, 619-635.

30. Zhang, R., et al. (2020). Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling. Engineering Structures, 215.

31. Ren, P., Rao, C., Sun, H., & Liu, Y. (2022). SeismicNet: Physics-informed neural networks for seismic wave modeling in semi-infinite domain. arXiv preprint arXiv:2210.14044.

32. R. Zhang, Y. Liu, H. Sun. (2020). Physics-informed multi-LSTM networks for metamodeling of nonlinear structures, Computer Methods in Applied Mechanics and Engineering. 369: 113226.

33. Lu, Xinzheng, et al. (2022). Intelligent structural design of shear wall residence using physics-enhanced generative adversarial networks. Earthquake Engineering & Structural Dynamics 51,7: 1657-1676.

34. Y. Chen, L. Lu, G.E. Karniadakis, L.D. Negro. (2020). Physics-informed neural networks for inverse problems in nano-optics and metamaterials, Opt. Express 28 (8):11618-11633.

505 35. Tartakovsky, A. M., Marrero, C. O., Perdikaris, P., Tartakovsky, G. D., & Barajas-Solano,
506 D. (2020). Physics-informed deep neural networks for learning parameters and
507 constitutive relationships in subsurface flow problems. Water Resources Research, 56(5),
508 e2019WR026731.
509 36. Rezaei, S., Harandi, A., Moeineddin, A., Xu, B. X., & Reese, S. (2022). A mixed
510 formulation for physics-informed neural networks as a potential solver for engineering
511 problems in heterogeneous domains: comparison with finite element method. Computer
512 Methods in Applied Mechanics and Engineering, 401, 115616.
513 37. Yang, L., Zhang, D., & Karniadakis, G. E. (2020). Physics-informed generative
514 adversarial networks for stochastic differential equations. SIAM Journal on Scientific
515 Computing, 42(1), A292-A317.
516 38. A.D. Jagtap, E. Kharazmi, G.E. Karniadakis. (2020). Conservative physics-informed
517 neural networks on discrete domains for conservation laws: Applications to forward and
518 inverse problems, Computer Methods in Applied Mechanics and Engineering. 365,
519 113028.
520 39. J. Yu, L. Lu, X. Meng, G.E. Karniadakis. (2022). Gradient-enhanced physics-informed
521 neural networks for forward and inverse PDE problems, Computer Methods in Applied
522 Mechanics and Engineering. 393, 114823.
523 40. H. Gao, M.J. Zahr, J.X. Wang. (2022). Physics-informed graph neural Galerkin networks:
524 A unified framework for solving PDE-governed forward and inverse problems,
525 Computer Methods in Applied Mechanics and Engineering, 390.
526 41. Zhang, D., Lu, L., Guo, L., & Karniadakis, G. E. (2019). Quantifying total uncertainty in

physics-informed neural networks for solving forward and inverse stochastic problems. Journal of Computational Physics, 397, 108850.

42. H. Gao, M.J. Zahr, J.X. Wang. (2022). Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems, Computer Methods in Applied Mechanics and Engineering, 390.

43. J. Yu, L. Lu, X. Meng, G.E. Karniadakis. (2022). Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Computer Methods in Applied Mechanics and Engineering. 393, 114823.

44. Haghighat, E., Bekar, A. C., Madenci, E., & Juanes, R. (2021). A nonlocal physics-informed deep learning framework using the peridynamic differential operator. Computer Methods in Applied Mechanics and Engineering, 385, 114012.

45. Meng, X., Li, Z., Zhang, D., & Karniadakis, G. E. (2020). PPINN: Parareal physics-informed neural network for time-dependent PDEs. Computer Methods in Applied Mechanics and Engineering, 370, 113250.

46. Zhu, Y., Zabaras, N., Koutsourelakis, P. S., & Perdikaris, P. (2019). Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. Journal of Computational Physics, 394, 56-81.

47. Hennigh, O., Narasimhan, S., Nabian, M. A., Subramaniam, A., Tangsali, K., Fang, Z., ... & Choudhry, S. (2021, June). NVIDIA SimNet™: An AI-accelerated multi-physics simulation framework. In Computational Science–ICCS 2021: 21st International Conference, Krakow, Poland, June 16-18, 2021, Proceedings, Part V (pp. 447-461). Cham: Springer International Publishing.

549 48. Sun, L., & Wang, J. X. (2020). Physics-constrained bayesian neural network for fluid
550 flow reconstruction with sparse and noisy data. Theoretical and Applied Mechanics
551 Letters, 10(3), 161-169.
552 49. Wang, J. X., Wu, J. L., & Xiao, H. (2017). Physics-informed machine learning approach
553 for reconstructing Reynolds stress modeling discrepancies based on DNS data. Physical
554 Review Fluids, 2(3), 034603.
555 50. Eivazi, H., Tahani, M., Schlatter, P., & Vinuesa, R. (2022). Physics-informed neural
556 networks for solving Reynolds-averaged Navier-Stokes equations. Physics of Fluids,
557 34(7), 075117.
558 51. L. Sun, H. Gao, S. Pan, J.X. Wang. (2020). Surrogate modeling for fluid flows based on
559 physics-constrained deep learning without simulation data, Computer Methods in
560 Applied Mechanics and Engineering. 361, 112732.
561 52. Erichson, N. B., Muehlebach, M., & Mahoney, M. W. (2019). Physics-informed
562 autoencoders for Lyapunov-stable fluid flow prediction. arXiv preprint
563 arXiv:1905.10866.
564 53. Li, W., Bazant, M. Z., & Zhu, J. (2021). A physics-guided neural network framework for
565 elastic plates: Comparison of governing equations-based and energy-based approaches.
566 Computer Methods in Applied Mechanics and Engineering, 383, 113933.
567 54. Nguyen-Thanh, V. M., Zhuang, X., & Rabczuk, T. (2020). A deep energy method for
568 finite deformation hyperelasticity. European Journal of Mechanics-A/Solids, 80, 103874.
569 55. Haghighat, Ehsan, Danial Amini, and Ruben Juanes. (2022). Physics-Informed Neural
570 Network Simulation of Multiphase Poroelasticity Using Stress-Split Sequential Training.

Computer Methods in Applied Mechanics and Engineering, 397.

56. Haghighat, Ehsan, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. (2021). A Physics-Informed Deep Learning Framework for Inversion and Surrogate Modeling in Solid Mechanics. Computer Methods in Applied Mechanics and Engineering, 379.

57. Xu, Chen, Ba Trung Cao, Yong Yuan, and Günther Meschke. (2023). Transfer Learning Based Physics-Informed Neural Networks for Solving Inverse Problems in Engineering Structures under Different Loading Scenarios. Computer Methods in Applied Mechanics and Engineering, 405.

58. Rao, Chengping, Hao Sun, and Yang Liu. (2021). Physics-Informed Deep Learning for Computational Elastodynamics without Labeled Data. Journal of Engineering Mechanics 147, 8.