Sahin et al. *Advanced Modeling and Simulation in Engineering Sciences(2024)11:11*
https://doi.org/10.1186/s40323-024-00265-3

Advanced Modeling and Simulation in Engineering Sciences

RESEARCH ARTICLE Open Access

Solving forward and inverse problems of contact mechanics using physics-informed neural networks

Tarik Sahin[1*], Max von Danwitz[2] and Alexander Popp[1,2]

*Correspondence: tarik.sahin@unibw.de
[1]Institute for Mathematics and Computer-Based Simulation (IMCS), University of the Bundeswehr Munich, Werner-Heisenberg-Weg 39, 85577 Neubiberg, Bavaria, Germany
[2]Institute for the Protection of Terrestrial Infrastructures, German Aerospace Center (DLR), Rathausallee 12, 53757 Sankt Augustin, North Rhine-Westphalia, Germany

Abstract

This paper explores the ability of physics-informed neural networks (PINNs) to solve forward and inverse problems of contact mechanics for small deformation elasticity. We deploy PINNs in a mixed-variable formulation enhanced by output transformation to enforce Dirichlet and Neumann boundary conditions as hard constraints. Inequality constraints of contact problems, namely *Karush–Kuhn–Tucker* (KKT) type conditions, are enforced as soft constraints by incorporating them into the loss function during network training. To formulate the loss function contribution of KKT constraints, existing approaches applied to elastoplasticity problems are investigated and we explore a nonlinear complementarity problem (NCP) function, namely *Fischer–Burmeister*, which possesses advantageous characteristics in terms of optimization. Based on the Hertzian contact problem, we show that PINNs can serve as pure partial differential equation (PDE) solver, as data-enhanced forward model, as inverse solver for parameter identification, and as fast-to-evaluate surrogate model. Furthermore, we demonstrate the importance of choosing proper hyperparameters, e.g. loss weights, and a combination of *Adam* and *L-BFGS-B* optimizers aiming for better results in terms of accuracy and training time.

Keywords: Physics-informed neural networks, Mixed-variable formulation, Contact mechanics, Enforcing inequalities, *Fischer–Burmeister* NCP-function

Introduction

Machine learning approaches usually require a large amount of simulation or experimental data, which might be challenging to acquire due to the complexity of simulations and the cost of experiments. Also, data scarcity can cause data-driven techniques to perform poorly in terms of accuracy. This is particularly true when using real-world observations that are noisy or datasets that are incorrectly labeled, as there is no physics-based feedback mechanism to validate the predictions. To tackle this problem, physics-informed neural networks (PINNs) have been developed. PINNs integrate boundary or initial boundary value problems and measurement data into the neural network's loss function to compensate for the lack of sufficient data and the black-box behavior of purely data-driven techniques [1]. In terms of forward problems, PINNs can serve as a partial differential

equation (PDE) solver even in cases where domains are irregular. This is because PINNs utilize automatic differentiation and therefore do not require any connectivity of the sampling points, making them a mesh-free method [2]. Moreover, PINNs can break the curse of dimensionality when approximating functions in higher dimensions [3,4]. Additionally, PINNs are a good candidate for addressing inverse problems due to the easy integration of measurement data [5].

To take advantage of these benefits, PINNs have been employed in various fields of engineering and science including geosciences [6], fluid mechanics [7,8], optics and electromagnetics [9–11], and industrial applications, e.g., fatigue prognosis of a wind turbine main bearing [12]. Based on sensor data of a physical object, PINNs can be used in hybrid digital twins of civil engineering structures [13] and for critical infrastructure protection [14]. Particularly in solid mechanics, PINNs have been developed for solving problems of linear elasticity, elastodynamics, elastoplasticity [15], and inverse problems for parameter identification [16]. Rao et al. [17] propose PINNs in a mixed-variable formulation to solve elastodynamic problems inspired by hybrid finite element analysis [18]. They introduce displacement and stress components as neural network output to enforce boundary conditions as hard constraints by deploying additional parallel networks. Also, it is claimed that a mixed-variable formulation enhances the accuracy and ease of training for the network. Samaniego et al. [19] utilize energy methods to develop PINNs for solving various examples in computational mechanics, i.e. elastodynamics, hyperelasticity and phase field modeling of fracture. Lu and colleagues develop physics-informed neural networks with hard constraints (hPINNs) to perform topology optimization [20]. The authors enhance the loss formulation with the penalty method and the augmented Lagrangian method to enforce inequality constraints as hard constraints. Moreover, they deploy output transformation to enforce equality constraints explicitly for simple domains as introduced in the study of Lagaris et al. [21]. In another study, Haghighat and colleagues utilize PINNs in the field of solid mechanics to tackle inverse problems and construct surrogate models [22]. Their approach involves parallel networks based on the mixed-variable formulation for linear elasticity, and they expand their methodology to address nonlinear elastoplasticity problems including classical *Karush–Kuhn–Tucker* (KKT) type inequality constraints. They enforce KKT constraints as soft constraints via a sign function, which has discontinuous gradients. As an extension of their previous work on elastoplasticity, Haghighat et al. [15] deploy PINNs for constitutive model characterization and discovery through calibration by macroscopic mechanical testing on materials. As an alternative to the sign function, they adopt the Sigmoid function to enforce KKT constraints, since Sigmoid has well-defined gradients, but requires an additional hyperparameter.

As far as the authors are most aware, no previous work has been conducted on PINNs to solve contact mechanics problems. Here, we focus on the novel application of PINNs for contact mechanics for small deformation elasticity including benchmark examples, e.g. contact between an elastic block and a rigid flat surface, as well as the Hertzian contact problem. To enforce displacement and traction boundary conditions, we deploy PINNs with output transformation in the mixed-variable formulation inspired by the Hellinger–Reissner principle [23] in which displacement and stress fields are defined as network outputs. Additionally, contact problems involve a well-known set of *Karush–Kuhn–Tucker* type inequality and equality constraints sometimes also referred to as Hertz–Signorini–Moreau conditions in the contact mechanics community. We enforce this given set of

equations as soft constraints via three different methods: sign-based method, Sigmoid-based method and a nonlinear complementarity problem (NCP) function, namely the *Fischer–Burmeister* function. NCP functions enable reformulating inequalities as a system of equations, and have proven particularly robust and efficient for the design of semi-smooth Newton methods in contact analysis [24–27].

To validate our PINN formulation for contact mechanics, two examples are investigated. The first example involves the contact between an elastic block and a rigid flat surface where all points in the possible contact area will actually be in contact. The second example is the famous Hertzian contact problem, where the actual contact area will be determined as part of the solution procedure. Furthermore, we illustrate four distinct PINN application cases for the Hertzian contact problem. In the first use case, we deploy the PINN as a pure forward solver to validate our approach by comparing results with a finite element simulation. PINNs can easily incorporate external data, such as measurements or simulations. In the second scenario, we therefore utilize displacement and stress fields obtained through FEM (in the sense of "virtual experiments") to enhance the accuracy of our PINN model. The third application is to deploy PINNs to solve inverse problems, particularly identifying the prescribed external load in the Hertzian contact problem based on FEM data. As a fourth and final example, the load (external pressure) is considered as another network input to construct a fast-to-evaluate surrogate model, which predicts displacement and stress fields for unseen pressure inputs. In the very active research field of physics-informed machine learning further advanced techniques, such as variational PINNs (VPINNs) [28,29] and integrated finite element neural networks (I-FENNs) [30], have been proposed recently. In particular, VPINNs as a Petrov–Galerkin scheme as compared to collocation in standard PINNs might be of interest for (non-smooth) contact problems. However, these recent developments are beyond the scope of this first study and subject to future work.

The remainder of this article is structured as follows: "Problem formulation" section summarizes the fundamental equations and constraints of contact mechanics with small deformation elasticity. Also, the basics of the so-called mixed-variable formulation based on the Hellinger–Reissner principle are given. In "Physics-informed neural networks for solid and contact mechanics" section, a generalized formulation of PINNs with output transformation is outlined in detail, which in principle allows for the solution of arbitrary partial differential equations (PDEs). We narrow the field of interest down to solid and contact mechanics problems based on a mixed-variable formulation, and therefore different methods to enforce KKT constraints are explained. Several benchmark examples are analyzed in "Numerical examples" section, including the Lamé problem of elasticity, contact between an elastic block and a rigid domain, and the Hertzian contact problem. "Conclusion" section concludes the paper by summarizing our key findings and providing an outlook on future research directions.

Problem formulation

Contact mechanics

We consider a 2D contact problem between an elastic body and a fixed rigid obstacle as illustrated in Fig. 1. In the reference configuration, the elastic body is denoted by Ω_0, and in the current configuration, it is represented by Ω_t while the rigid obstacle has the

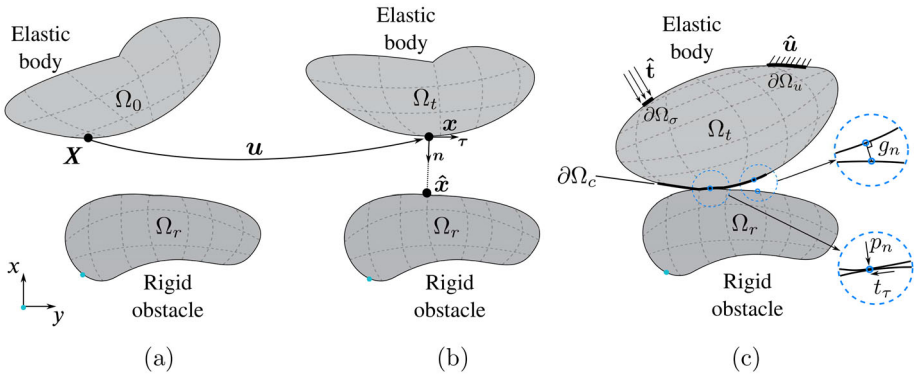

Fig. 1 Contact problem between an elastic body and a rigid obstacle. **a** Reference configuration, **b** current configuration, **c** accompanying boundary conditions, illustration of the gap g_n, tangential traction $t_{\boldsymbol{\tau}}$ and contact pressure p_n

same configuration Ω_r. Figure 1c shows a configuration for which two bodies come into contact. The surface of the elastic body can be partitioned into three sections: the Dirichlet boundary $\partial\Omega_u$, where displacements are prescribed, the Neumann boundary $\partial\Omega_\sigma$, where tractions are given, and the potential contact boundary $\partial\Omega_c$ where contact constraints are imposed. The actual contact surface is a subset of $\partial\Omega_c$ and is sought for during the solution procedure.

Let us consider the boundary value problem (BVP) of small deformation elasticity

$$\nabla\cdot\boldsymbol{\sigma} + \hat{\mathbf{b}} = \mathbf{0} \qquad \text{in } \Omega, \qquad \rightarrow \text{(BE)} \tag{1}$$

$$\boldsymbol{u} = \hat{\boldsymbol{u}} \qquad \text{on } \partial\Omega_u, \qquad \rightarrow \text{(DBC)} \tag{2}$$

$$\boldsymbol{\sigma}\cdot\boldsymbol{n} = \hat{\boldsymbol{t}} \qquad \text{on } \partial\Omega_\sigma \qquad \rightarrow \text{(NBC)} \tag{3}$$

where $\boldsymbol{\sigma}$ denotes the Chauchy stress tensor, $\boldsymbol{u}$ is the displacement vector representing the so-called primal variable, $\hat{\mathbf{b}}$ denotes the body force vector, and $\boldsymbol{n}$ is the unit outward normal vector. Prescribed displacements are represented by $\hat{\boldsymbol{u}}$ on $\partial\Omega_u$, and $\hat{\mathbf{t}}$ denotes prescribed tractions on $\partial\Omega_\sigma$. Abbreviations BE, DBC and NBC denote the balance equation, Dirichlet boundary condition and Neumann boundary condition, respectively. The kinematic equation (KE) and constitutive equation (CE) for the deformable body are expressed as

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T), \qquad \rightarrow \text{(KE)} \tag{4}$$

$$\boldsymbol{\sigma} = \mathbb{C} : \boldsymbol{\varepsilon}. \qquad \rightarrow \text{(CE)} \tag{5}$$

Here, $\boldsymbol{\varepsilon}$ is the infinitesimal strain tensor and $\mathbb{C}$ is the fourth-order elasticity tensor. In the specific case of linear isotropic elasticity, the constitutive equation can be expressed via Hooke's law as

$$\boldsymbol{\sigma} = \lambda\,\mathrm{tr}(\boldsymbol{\varepsilon})\mathbf{I} + 2\mu\boldsymbol{\varepsilon}, \tag{6}$$

where λ and μ are the Lamé parameters, $\mathrm{tr}(\cdot)$ is the *trace* operator to sum strain components on the main diagonal and $\mathbf{I}$ is the identity tensor.

The displacement vector $\boldsymbol{u}$ can be obtained for the elastic body by describing the motion from the reference configuration $\boldsymbol{X}$ to the current configuration $\boldsymbol{x}$ as follows (see Fig. 1a, b)

$$\boldsymbol{u} = \boldsymbol{x} - \boldsymbol{X}. \tag{7}$$

The gap function (GF) g_n is defined as a distance measure between elastic and rigid bodies in the current configuration as

$$g_n = -\boldsymbol{n} \cdot (\boldsymbol{x} - \hat{\boldsymbol{x}}). \qquad \rightarrow \text{(GF)} \tag{8}$$

The term $\hat{\boldsymbol{x}}$ denotes the so-called closest point projection of $\boldsymbol{x}$ onto the surface of Ω_r (see Fig. 1b). Since all contact constraints will be defined in the current configuration, p_n and t_τ can be obtained by traction vector decomposition (TVD) of the contact traction vector $\mathbf{t}_c$ as

$$\boldsymbol{t}_c = p_n \boldsymbol{n} + t_\tau \boldsymbol{\tau}, \quad p_n = \boldsymbol{t}_c \cdot \boldsymbol{n}, \quad t_\tau = \boldsymbol{t}_c \cdot \boldsymbol{\tau}, \quad \rightarrow \text{(TVD)} \tag{9}$$

where

$$\boldsymbol{t}_c = \boldsymbol{\sigma} \cdot \boldsymbol{n} \qquad \text{on } \partial\Omega_c. \quad \rightarrow \text{(CST)} \tag{10}$$

Cauchy's stress theorem (CST) states that the stress tensor $\boldsymbol{\sigma}$ maps the normal vector to the traction vector $\boldsymbol{t}_c$. Note that the boundary vectors $\boldsymbol{n}$ and $\boldsymbol{\tau}$ can be computed on the reference configuration assuming small deformation elasticity (linear elasticity) [31].

For a frictionless contact problem, we define the classical set of Karush–Kuhn–Tucker (KKT) conditions, commonly also referred to as Hertz–Signorini–Moreau (HSM) conditions, and the frictionless sliding condition (FSC) as

$$g_n \geqslant 0, \tag{11}$$
$$p_n \leqslant 0, \quad \rightarrow \text{(KKT)} \tag{12}$$
$$p_n g_n = 0, \tag{13}$$
$$t_\tau = 0. \quad \rightarrow \text{(FSC)} \tag{14}$$

Equation 11 enforces the kinematic aspect of non-penetration as shown in Fig. 1c. If two bodies are in contact, the gap vanishes, i.e., $g_n = 0$. The term p_n denotes the normal component of the contact traction, i.e., the contact pressure. Correspondingly, Eq. 12 guarantees that no adhesive stresses are allowed in the contact zone. Furthermore, the complementarity requirement in Eq. 13 necessitates that the gap should be zero when there is a non-zero contact pressure (point in contact), and the contact pressure should be zero if there is a positive gap (point not in contact). It should be noted that the tangential component of the traction vector vanishes for frictionless contact, resulting in Eq. 14. For additional information regarding more complex contact constitutive laws including friction, we refer to [24,32–34].

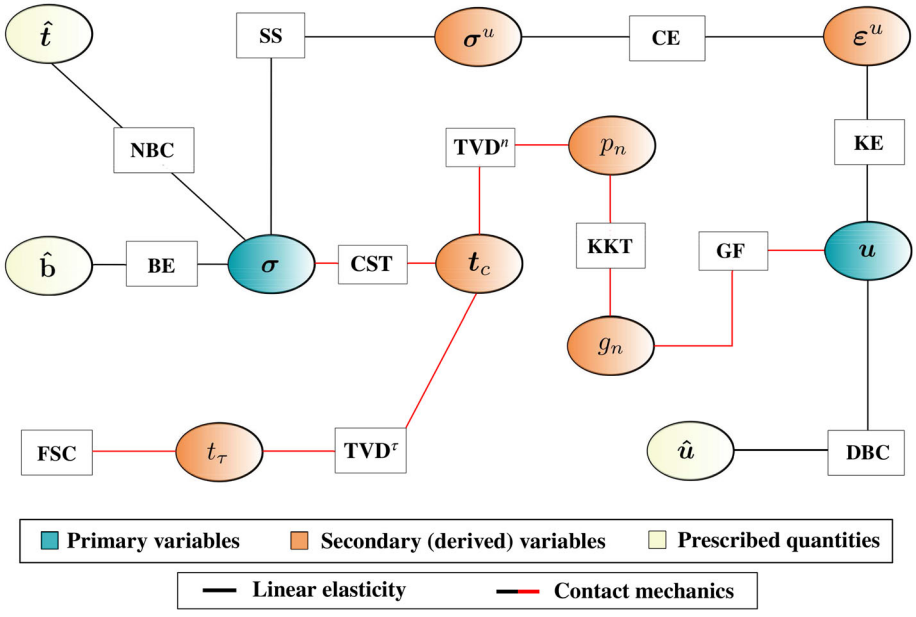

Fig. 2 Tonti's diagram of Hellinger–Reissner (HR) principle for contact problems with small strain theory and frictionless sliding condition

Mixed-variable formulation: the Hellinger–Reissner principle

Inspired by the Hellinger–Reissner principle [18,23], we construct a Tonti's diagram [35] to solve contact problems based on two primary variables: displacements $\boldsymbol{u}$ and stresses $\boldsymbol{\sigma}$ (see Fig. 2). The secondary (slave) variables are intermediate and they are derived from the primary variables, i.e., $\boldsymbol{\sigma}^u$ and $\boldsymbol{\varepsilon}^u$.

As shown in Fig. 2, the Tonti diagram summarizes the governing equations of contact problems with small strain theory and frictionless sliding condition (see Eqs. 1–14). The stress-to-stress coupling (SS) between the primary variable $\boldsymbol{\sigma}$ and the secondary variable $\boldsymbol{\sigma}^u$, defined as $\boldsymbol{\sigma} = \boldsymbol{\sigma}^u$, ensures that the two master fields remain compatible.

Physics-informed neural networks for solid and contact mechanics

Generic PINNs with output transformation

A generalized formulation for partial differential equations can be expressed in residual form with accompanying boundary conditions as

$$\begin{aligned} \mathcal{R}\big[\boldsymbol{u}(\boldsymbol{x})\big] &= \mathbf{0}, \quad \text{on } \Omega, \\ \mathcal{B}\big[\boldsymbol{u}(\boldsymbol{x})\big] - g(\boldsymbol{x}) &= \mathbf{0}, \quad \text{on } \partial\Omega. \end{aligned} \tag{15}$$

Here, $\mathcal{R}[\cdot]$ denotes a differential operator acting on a unknown solution $\boldsymbol{u}$, $\mathcal{B}[\cdot]$ is the boundary operator, $g(\boldsymbol{x})$ represents the prescribed boundary condition, $\boldsymbol{x}$ are the spatial coordinates that span the domain Ω and the boundary $\partial\Omega$.

Consider a fully-connected L-layers neural network to construct an approximated solution $\tilde{\boldsymbol{u}}$ to the BVP as follows [36,37]

$$\boldsymbol{u} \approx \tilde{\boldsymbol{u}} := \left(\mathcal{N}^L(\boldsymbol{z};\boldsymbol{\theta})\right)', \quad \mathcal{N}^L(\boldsymbol{z};\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}^n. \tag{16}$$

$\mathcal{N}^L(\boldsymbol{z};\boldsymbol{\theta})$ represents the network output in the output layer L, trainable network parameters, namely weights and biases, are denoted as $\boldsymbol{\theta}$, $(.)'$ represents a user-defined output transformation [20], and $\boldsymbol{z}$ is the network input such that $\boldsymbol{x} \subset \boldsymbol{z}$. Note that $\boldsymbol{z}$ can consist of spatial coordinates, time, and other additional input parameters. The network output is calculated using recursive $L-1$ element-wise operations between the input layer and hidden layers as

$$\begin{aligned} \text{input layer} &\rightarrow \mathcal{N}^1(\boldsymbol{z};\boldsymbol{\theta}) = \boldsymbol{z}, \\ \text{hidden layers} &\rightarrow \mathcal{N}^l(\boldsymbol{z};\boldsymbol{\theta}) = \psi(\boldsymbol{\theta}^l \cdot \mathcal{N}^{l-1}(\boldsymbol{z};\boldsymbol{\theta})), \text{ for } 2 \leqslant l \leqslant L-1, \\ \text{output layer} &\rightarrow \mathcal{N}^L(\boldsymbol{z};\boldsymbol{\theta}) = \boldsymbol{\theta}^L \cdot \mathcal{N}^{L-1}(\boldsymbol{z};\boldsymbol{\theta}), \end{aligned} \tag{17}$$

where ψ denotes the activation function that adds non-linearity to the layer output.

Output transformation enables the neural network to enforce boundary conditions explicitly. A user-defined output transformation can be obtained with suitable helper functions as

$$\tilde{u}_i(\boldsymbol{z};\boldsymbol{\theta}) = g(\boldsymbol{z}) + s \cdot h(\boldsymbol{z}) \cdot \mathcal{N}_{u_i}(\boldsymbol{z};\boldsymbol{\theta}), \tag{18}$$

where g is the prescribed boundary condition, s is a scaling parameter and h is a distance-to-boundary function fulfilling the following conditions:

$$\begin{aligned} h(\boldsymbol{z}) &= 0, \quad \text{on } \partial\Omega, \\ h(\boldsymbol{z}) &> 0, \quad \text{in } \Omega \setminus \partial\Omega. \end{aligned} \tag{19}$$

For simple boundaries $\partial\Omega$, it is relatively easy to define an appropriate distance-to-boundary function. On the other hand, it can become a quite challenging task in the case of arbitrary boundaries. One method to find a generalized distance-to-boundary function is to use NURBS parametrizations [38]. Moreover, the scaling parameter can help the optimizer avoid getting stuck in local minima by balancing target governing equations (see "Domination of p_n over g_n" section).

To ensure that $\tilde{\boldsymbol{u}}$ is a reasonable approximation of $\boldsymbol{u}$, the network parameters must be determined accordingly to the BVP. As shown in Fig. 3, the overall loss $\mathcal{L}(\boldsymbol{\theta})$ consists of PDE losses $\mathcal{L}_{\text{PDEs}}$, boundary condition losses $\mathcal{L}_{\text{BCs}}$ and experimental data losses $\mathcal{L}_{\text{EXPs}}$. Note that PDE derivatives are calculated using automatic differentiation (AD) [39]. To minimize the overall loss, the optimization process continues until a prescribed tolerance ϵ is reached so that optimal network parameters $\boldsymbol{\theta}^*$ are calculated

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \tag{20}$$

For each loss term, the inner loop sums up the *mean squared error* contributions of data points collected inside the domain or on the boundary. Specifically, $\{\boldsymbol{z}^{i_{rp}}\}_{i_{rp}=1}^{N_{rp}}$ denotes the collocation points in the domain, $\{\boldsymbol{z}^{i_{bp}}\}_{i_{bp}=1}^{N_{bp}}$ are the boundary points corresponding to the prescribed boundary conditions, and $\{\boldsymbol{z}^{i_{ep}}\}_{i_{ep}=1}^{N_{ep}}$ represents the points on which measurement data $\boldsymbol{u}^*$ is available. As the PDE residual $\mathcal{R}$ might have multiple terms and usually more than one boundary condition is defined, we use a lower index to explicitly point out that the inner summation is done for a given specific component, i.e. $\mathcal{R}_{i_n=1}$.

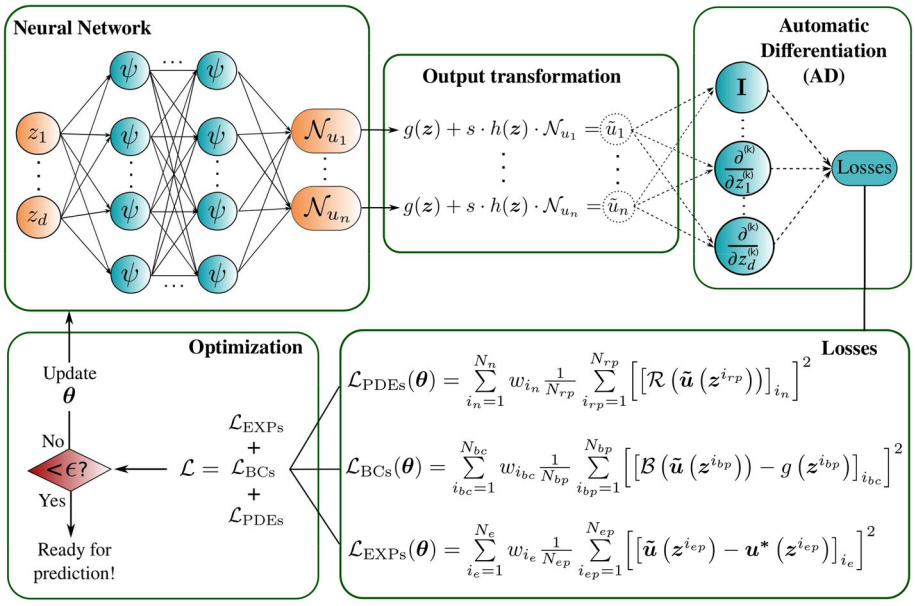

Fig. 3 The general representation of a physics-informed neural network for a BVP

Consequently, the outer loop sums up the weighted contributions coming from individual components of the loss function. The terms $\{w_{i_n}\}_{i_n=1}^{N_n}$, $\{w_{i_{bc}}\}_{i_{bc}=1}^{N_{bc}}$ and $\{w_{i_e}\}_{i_e=1}^{N_e}$ denote the loss weights for the individual components of $\mathcal{L}_{\text{PDEs}}$, $\mathcal{L}_{\text{BCs}}$, $\mathcal{L}_{\text{EXPs}}$, respectively. We observe that the weighting of loss terms can be quite crucial for the convergence of the overall loss, since it avoids the optimizer expanding greater efforts on loss components that have a larger order of magnitude compared to others. It should be noted that the identical collocation points $\{\boldsymbol{z}^{i_{rp}}\}_{i_{rp}=1}^{N_{rp}}$ are employed in the calculation of every component of $\mathcal{L}_{\text{PDEs}}$. However, the boundary and experimental points may vary in each $\mathcal{L}_{\text{BCs}}$ and $\mathcal{L}_{\text{EXPs}}$ contribution.

Application of PINNs to solid and contact mechanics

In the context of solid and contact mechanics problems, we use PINNs with output transformation in a mixed-variable formulation. In the mixed-variable formulation for quasi-static problems without additional network input parameters (i.e. $\boldsymbol{z} = \boldsymbol{x}$), a fully-connected neural network (FNN) maps the given spatial coordinates $\boldsymbol{x}$ to the displacement vector $\boldsymbol{u}$ and stress tensor $\boldsymbol{\sigma}$. In other words, the displacement and stress fields are chosen as the quantities of interest that the FNN approximates as (see Fig. 4)

$$\tilde{\boldsymbol{u}} := (\mathcal{N}_{\boldsymbol{u}}(\boldsymbol{x};\boldsymbol{\theta}))' \quad \text{and} \quad \tilde{\boldsymbol{\sigma}} := (\mathcal{N}_{\boldsymbol{\sigma}}(\boldsymbol{x};\boldsymbol{\theta}))'. \tag{21}$$

Combining the information provided in Fig. 3 for losses of general PINNs with the governing equations of solid mechanics, we obtain the total loss $\mathcal{L}_E$ for linear elasticity (without contact) in the mixed-variable formulation with additional experimental data as

$$\mathcal{L}_E = \mathcal{L}_{\text{PDEs}} + \mathcal{L}_{\text{DBCs}} + \mathcal{L}_{\text{NBCs}} + \mathcal{L}_{\text{EXPs}}, \tag{22}$$

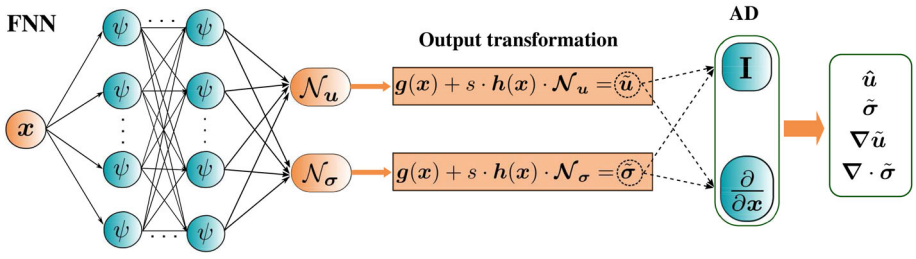

Fig. 4 Physics-informed neural networks in the mixed-variable form to solve quasi-static solid and contact mechanics problems without additional network parameters

where

$$
\begin{aligned}
\mathcal{L}_{\text{PDEs}} &= \sum_{i_n=1}^{N_m} w_{i_n} \frac{1}{N_{rp}} \sum_{i_{rp}=1}^{N_{rp}} \left[\left[\nabla \cdot \tilde{\boldsymbol{\sigma}} \left(\boldsymbol{x}^{i_{rp}} \right) + \hat{\mathbf{b}} \left(\boldsymbol{x}^{i_{rp}} \right) \right]_{i_n} \right]^2 + \\
&\quad \sum_{i_n=N_m+1}^{N_n} w_{i_n} \frac{1}{N_{rp}} \sum_{i_{rp}=1}^{N_{rp}} \left[\left[\tilde{\boldsymbol{\sigma}} \left(\boldsymbol{x}^{i_{rp}} \right) - \mathbb{C} : \tilde{\boldsymbol{\varepsilon}} \left(\boldsymbol{x}^{i_{rp}} \right) \right]_{i_n} \right]^2, \\
\mathcal{L}_{\text{DBCs}} &= \sum_{i_{bc,D}=1}^{N_{bc,D}} w_{i_{bc,D}} \frac{1}{N_{bp,D}} \sum_{i_{bp,D}=1}^{N_{bp,D}} \left[\left[\tilde{\boldsymbol{u}} \left(\boldsymbol{x}^{i_{bp,D}} \right) - \hat{\boldsymbol{u}} \left(\boldsymbol{x}^{i_{bp,D}} \right) \right]_{i_{bc,D}} \right]^2, \\
\mathcal{L}_{\text{NBCs}} &= \sum_{i_{bc,N}=1}^{N_{bc,N}} w_{i_{bc,N}} \frac{1}{N_{bp,N}} \sum_{i_{bp,N}=1}^{N_{bp,N}} \left[\left[\tilde{\boldsymbol{\sigma}} \left(\boldsymbol{x}^{i_{bp,N}} \right) \cdot \boldsymbol{n} - \hat{\mathbf{t}} \left(\boldsymbol{x}^{i_{bp,N}} \right) \right]_{i_{bc,N}} \right]^2 \\
\mathcal{L}_{\text{EXPs}} &= \sum_{i_{e,\boldsymbol{u}}=1}^{N_{e,\boldsymbol{u}}} w_{i_{e,\boldsymbol{u}}} \frac{1}{N_{ep,\boldsymbol{u}}} \sum_{i_{ep,\boldsymbol{u}}=1}^{N_{ep,\boldsymbol{u}}} \left[\left[\tilde{\boldsymbol{u}} \left(\boldsymbol{x}^{i_{ep,\boldsymbol{u}}} \right) - \boldsymbol{u}^* \left(\boldsymbol{x}^{i_{ep,\boldsymbol{u}}} \right) \right]_{i_{e,\boldsymbol{u}}} \right]^2 + \\
&\quad \sum_{i_{e,\boldsymbol{\sigma}}=1}^{N_{e,\boldsymbol{\sigma}}} w_{i_{e,\boldsymbol{\sigma}}} \frac{1}{N_{ep,\boldsymbol{\sigma}}} \sum_{i_{ep,\boldsymbol{\sigma}}=1}^{N_{ep,\boldsymbol{\sigma}}} \left[\left[\tilde{\boldsymbol{\sigma}} \left(\boldsymbol{x}^{i_{ep,\boldsymbol{\sigma}}} \right) - \boldsymbol{\sigma}^* \left(\boldsymbol{x}^{i_{ep,\boldsymbol{\sigma}}} \right) \right]_{i_{e,\boldsymbol{\sigma}}} \right]^2.
\end{aligned}
\tag{23}
$$

Here, terms $\mathcal{L}_{\text{DBCs}}$ and $\mathcal{L}_{\text{NBCs}}$ denote losses for Dirichlet and Neumann BCs, respectively, and the $\mathcal{L}_{\text{EXPs}}$ term represents losses due to additional experimental data. In the mixed-variable formulation, the $\mathcal{L}_{\text{PDEs}}$ term is constructed in a composite form to fulfill both the balance equation (BE) and the stress-to-stress coupling (SS) as depicted in Fig. 2. The index N_m is used to distinguish the loss weights related to BE and SS. Since stress components are directly defined as network outputs, traction or Neumann BCs can be imposed as hard constraints using output transformation. Moreover, it is sufficient to calculate first-order derivatives of the neural network outputs with respect to the inputs, since the governing equations in the mixed-variable formulation contain only first-order derivatives. An alternative to the mixed-variable formulation is the classical displacement-based formulation in which only $\boldsymbol{u}$ is considered as the network output. However, such an approach requires second-order derivatives for evaluating the balance equation, and traction BCs can not be enforced as hard constraints [17,40].

Next, we construct the composite (total) loss function $\mathcal{L}_{\text{C}}$ for linear elasticity including contact as follows:

$$
\mathcal{L}_{\text{C}} = \mathcal{L}_{\text{E}} + \mathcal{L}_{\text{FS}} + \mathcal{L}_{\text{KKT}}. \tag{24}
$$

The first additional loss term

$$\mathcal{L}_{\mathrm{FS}} = w^{(\mathrm{fs})} |t_\tau|_{\partial\Omega_c} \tag{25}$$

enforces the frictionless sliding condition in the contact zone $\partial\Omega_c$ (see Eqs. 9 and 10). For simplicity, we denote the *mean squared error* (MSE) as $|\cdot|$, which can be calculated as $\frac{1}{n}\sum_{i=1}^{n}(\cdot)^2$. The second additional term $\mathcal{L}_{\mathrm{KKT}}$ will be elaborated in the next section.

While the various ways of evaluating the normal gap g_n are a matter of intense discussions, especially within discretization schemes such as the finite element method [34,41], a very simple gap calculation is sufficient here due to the fact that we only consider contact problems between an elastic body and a rigid flat surface. The normal gap is consistently expressed by evaluating the orthogonal projection of the elastic body onto the rigid flat surface.

Enforcing the Karush–Kuhn–Tucker inequality constraints

There are several methods available to enforce inequality conditions in general. The direct approach is to formulate loss functions of inequalities and impose them as soft constraints with fixed loss weights [15,22]. However, setting large loss weights can cause an ill-conditioned problem [20]. On the other hand, when small loss weights are chosen, the estimated solution may violate the inequalities. To tackle this problem, authors in [20] suggest penalty and augmented Lagrangian methods, well-known from constrained optimization, which construct loss formulations with adaptive loss weights. In the following, we investigate three methods to enforce KKT conditions of normal contact problems based on soft constraints.

Sign-based method

One possible way to enforce KKT conditions is to use the sign function [22], which leads to

$$\begin{aligned}\mathcal{L}_{\mathrm{KKT}} &= \mathcal{L}_{\tilde{g}_n \geqslant 0} + \mathcal{L}_{\tilde{p}_n \leqslant 0} + \mathcal{L}_{\tilde{p}_n \tilde{g}_n = 0} \\ &= w_1^{(\mathrm{KKT})} \left| \frac{1}{2}\left(1 - \operatorname{sign}(\tilde{g}_n)\right) \tilde{g}_n \right|_{\partial\Omega_c} + w_2^{(\mathrm{KKT})} \left| \frac{1}{2}\left(1 + \operatorname{sign}(\tilde{p}_n)\right) \tilde{p}_n \right|_{\partial\Omega_c} + \\ &\quad w_3^{(\mathrm{KKT})} \left| \tilde{p}_n \, \tilde{g}_n \right|_{\partial\Omega_c}.\end{aligned} \tag{26}$$

Here, $\{w_i^{(\mathrm{KKT})}\}_{i=1}^{3}$ represent the loss weights on the corresponding KKT condition. Figure 5 illustrates that $\frac{1}{2}\left(1 - \operatorname{sign}(\tilde{g}_n)\right)\tilde{g}_n$ contributes to the loss component $\mathcal{L}_{g_n \geqslant 0}$ when the gap g_n is less than zero. On the other hand, $\frac{1}{2}\left(1 + \operatorname{sign}(\tilde{p}_n)\right)\tilde{p}_n$ contributes to $\mathcal{L}_{p_n \leqslant 0}$ if the contact pressure p_n is positive. Note that the sign function has gradient jumps, which is typically not a desired feature in the context of optimization.

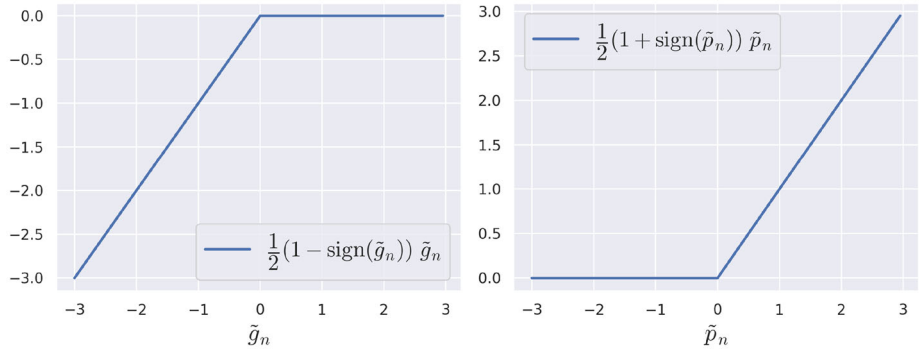

Fig. 5 An illustration of the sign-based function depending on gap g_n and contact pressure p_n

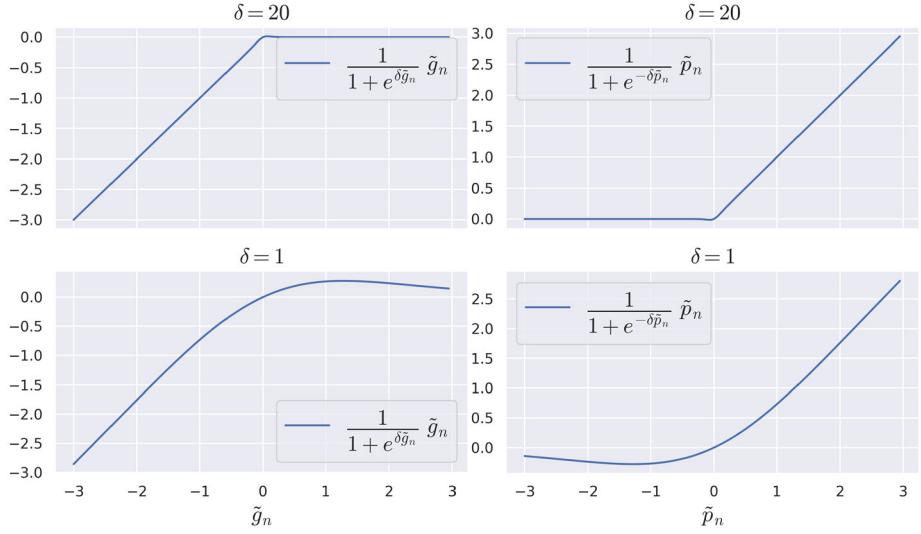

Fig. 6 An illustration of the sigmoid-based function depending on gap g_n and contact pressure p_n for different δ values

Sigmoid-based method

An alternative approach to circumvent discontinuous gradients is to use the Sigmoid function [15] to obtain $\mathcal{L}_{\text{KKT}}$ as

$$\begin{aligned}\mathcal{L}_{\text{KKT}} &= \mathcal{L}_{\tilde{g}_n \geqslant 0} + \mathcal{L}_{\tilde{p}_n \leqslant 0} + \mathcal{L}_{\tilde{p}_n \tilde{g}_n = 0} \\ &= w_1^{(\text{KKT})} \Big| \frac{1}{1+e^{\delta \tilde{g}_n}} \tilde{g}_n \Big|_{\partial\Omega_c} + w_2^{(\text{KKT})} \Big| \frac{1}{1+e^{-\delta \tilde{p}_n}} \tilde{p}_n \Big|_{\partial\Omega_c} + \\ &\quad w_3^{(\text{KKT})} \big| \tilde{p}_n\, \tilde{g}_n \big|_{\partial\Omega_c},\end{aligned} \tag{27}$$

where δ is the *steepness* parameter that controls the transition between zero and non-zero loss contributions. As depicted in Fig. 6, the Sigmoid function avoids gradient jumps through exponential regularization term. However, when δ is chosen too small, e.g. $\delta = 1$, then significant unphysical loss function values are obtained for $\tilde{g}_n > 0$ and $\tilde{p}_n < 0$. On the other hand, setting δ too large recovers the sign-based implementation. Therefore, a parameter study must be conducted to find the optimal parameter δ_{opt}.

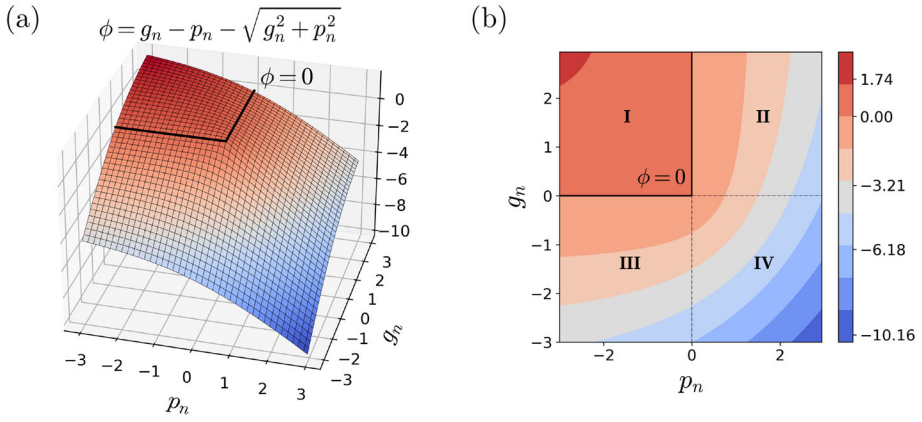

Fig. 7 The Fischer–Burmeister NCP function depending on gap g_n and contact pressure p_n as a 3D plot (**a**) and as a 2D contour plot (**b**)

A nonlinear complementarity problem function: Fischer–Burmeister

Nonlinear complementarity problem (NCP) functions are developed based on reformulating inequalities as equalities [42]. One popular choice of NCP function is the Fischer–Burmeister function [43] expressed as

$$\phi_{\mathrm{FB}}(a, b) := a + b - \sqrt{a^2 + b^2} = 0 \quad \Longleftrightarrow \quad a \geqslant 0, b \geqslant 0, ab = 0. \tag{28}$$

By setting $a = \tilde{g}_n$ and $b = -\tilde{p}_n$ in the Fischer–Burmeister function, we obtain $\mathcal{L}_{\mathrm{KKT}}$ as follows

$$\mathcal{L}_{\mathrm{KKT}} = w^{(\mathrm{KKT})} \left| \phi_{\mathrm{FB}}(\tilde{g}_n, -\tilde{p}_n) \right|_{\partial\Omega_c} = w^{(\mathrm{KKT})} \left| \tilde{g}_n - \tilde{p}_n - \sqrt{\tilde{g}_n^2 + \tilde{p}_n^2} \right|_{\partial\Omega_c}. \tag{29}$$

The Fischer–Burmeister function is a particularly suitable choice for typical loss calculations based on the mean squared error (MSE), since $(\phi_{\mathrm{FB}})^2$ is continuously differentiable also at $a = b = 0$ as reported [44]. As shown in Fig. 7, the largest loss contribution comes from section IV in which both excessive penetrations, i.e. $g_n \leqslant 0$, and large adhesive stresses, i.e. $p_n \geqslant 0$, are present. We refer to [42–45] for more details about the Fischer–Burmeister function. Note also that having fewer loss terms generally eases the optimization process as well as parameter tuning, and in contrast to the previous variants in "Sign-based method" and "Sigmoid-based method" sections only one single loss weight is required in the case of the Fischer–Burmeister NCP function.

Algorithmic challenges

Domination of p_n over g_n

The Fischer–Burmeister NCP function reduces a set of inequality constraints into a single equation. However, it might cause domination of one term over another. As explained in the previous sections, g_n is derived from the displacement field $\boldsymbol{u}$ and p_n is derived from the Cauchy stress field $\boldsymbol{\sigma}$. Depending on the chosen problem parameters, e.g., stiffness, the estimated quantities $\boldsymbol{u}$ and $\boldsymbol{\sigma}$ might have different scales, and then so do g_n and p_n. As illustrated in Fig. 8, when g_n and p_n have similar scales, there is no domination. But increasing p_n causes domination of p_n over g_n. In terms of optimization, the neural network will then expand more effort into minimizing the large-scale quantity p_n, which

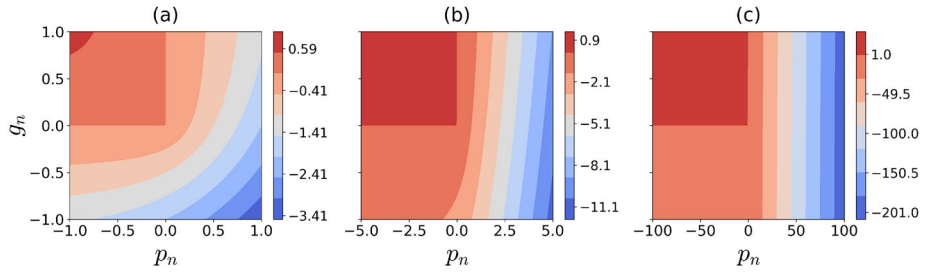

Fig. 8 Domination of p_n over g_n: **a** no domination, **b** intermediate domination and **c** strong domination

might cause unacceptable violations of constraints related to g_n. To tackle this problem, non-dimensionalization techniques can be used to ensure that p_n and g_n have similar scales [46].

Importance of output scaling

Output scaling is a functionality of output transformation that can prevent the optimizer to get stuck in local minima. In the context of solid and contact mechanics, the PINNs estimate the displacement field $\boldsymbol{u}$ and the stress field $\boldsymbol{\sigma}$ as neural network outputs. Assuming that no transfer learning is used, the first estimations of outputs are done randomly due to random initialization of the neural network parameters. Also, there are well-known and popular methods that generate initial network estimations following a normal distribution with a mean of zero and a standard deviation of one [47], e.g., *Glorot* initialization. However, using such initialization methods could cause convergence issues because of the output quantities having similar magnitudes. This issue can be explained through the example of the SS loss term (see Eq. 23)

$$\mathcal{L}_{\mathrm{SS}} = \sum_{i_n=N_m+1}^{N_n} w_{i_n} \frac{1}{N_{rp}} \sum_{i_{rp}=1}^{N_{rp}} \left[\left[\boldsymbol{\sigma}\left(\boldsymbol{x}^{i_{rp}}\right) - \mathbb{C} : \boldsymbol{\varepsilon} \right]_{i_n} \right]^2. \tag{30}$$

Minimization of $\mathcal{L}_{\mathrm{SS}}$ requires the condition $\boldsymbol{\sigma} = \mathbb{C} : \boldsymbol{\varepsilon}$ to hold. In case very large values of the material properties are chosen, e.g. Young's modulus, the term $\mathbb{C} : \boldsymbol{\varepsilon}$ will initially dominate $\boldsymbol{\sigma}$, which means a large loss contribution has to be handled by the optimizer. Therefore, to minimize the loss, either a significant increase in $\boldsymbol{\sigma}$ or a significant decrease in $\boldsymbol{u}$ is required. Such large increments in the optimization procedure are troublesome, as the gradient of the employed tanh() activation function tends to zero for large function arguments, which is also referred to as the vanishing gradient problem [48]. To ease optimization, the network output $\boldsymbol{u}$ can be scaled by the inverse of the Young's modulus, i.e. by $\frac{1}{E}$. This ensures that the initial magnitudes of both terms in the SS condition are comparable, which summarizes the benefits of output scaling in a nutshell.

Numerical examples

In the following, we investigate three numerical examples. The first example is the well-known Lamé problem of elasticity, which is considered as a preliminary test without contact to verify that our PINN framework works as expected, including in particular the hard enforcement of DBC and NBC with output transformation. Afterward, our investigation focuses on examining two contact examples: a contact problem between a simple

square block and a rigid flat surface, and the Hertzian contact problem. The main difference between the two contact examples is the fact that the actual contact area has to be identified by the PINN in the Hertzian example, while the potential and actual contact areas are the same in the case of the square block and the rigid flat surface. Note that 2D plane strain conditions are considered throughout the entire section and body forces are neglected.

All numerical examples have the following common settings. The PINN maps spatial coordinates $\boldsymbol{x} = (x, y)$ as inputs to transformed mixed-form outputs $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\sigma}}) = (\tilde{u}_x, \tilde{u}_y, \tilde{\sigma}_{xx}, \tilde{\sigma}_{yy}, \tilde{\sigma}_{xy})$. Networks are initialized using the *Glorot uniform* initializer, and the *tanh* function is chosen as activation function. Models are first trained using the stochastic gradient descent optimizer *Adam* [49] with a learning rate, $lr = 0.001$, for 2000 epochs, and then we switch to the limited memory BFGS algorithm including box constraints (*L-BFGS-B*) [50] until one of the stopping criteria is met [51,52]. Our workflow is developed based on the *DeepXDE* package [53] and we refer to the *DeepXDE* documentation for default *L-BFGS-B* options. Note that training points are generated by *GMSH*, since it has strong capabilities in mesh generation and visualization, and provides boundary normals at arbitrary query points [54].

As a common error metric, we report the vector-based relative L_2 errors for displacement $E^{\boldsymbol{u}}_{L_2}$ and stress fields $E^{\boldsymbol{\sigma}}_{L_2}$ as follows

$$\begin{aligned} E^{\boldsymbol{u}}_{L_2} &= \frac{\sqrt{\sum_i \sum_{j=1}^{N_{\text{test}}} \left(\tilde{u}_i(\boldsymbol{x}^j) - u_i(\boldsymbol{x}^j)\right)^2}}{\sqrt{\sum_i \sum_{j=1}^{N_{\text{test}}} \left(u_i(\boldsymbol{x}^j)\right)^2}} \quad \text{for } i = (x, y), \\ E^{\boldsymbol{\sigma}}_{L_2} &= \frac{\sqrt{\sum_i \sum_{j=1}^{N_{\text{test}}} \left(\tilde{\sigma}_i(\boldsymbol{x}^j) - \sigma_i(\boldsymbol{x}^j)\right)^2}}{\sqrt{\sum_i \sum_{j=1}^{N_{\text{test}}} \left(\sigma_i(\boldsymbol{x}^j)\right)^2}} \quad \text{for } i = (xx, yy, xy). \end{aligned} \tag{31}$$

Here, $\tilde{u}_i$ and $\tilde{\sigma}_i$ denote PINN solutions and u_i and σ_i denote reference solutions that are obtained analytically or numerically. Also, the index $j = (1, \ldots, N_{\text{test}})$ runs over the test points that are generated using structured meshes. We refer to Appendix 3 for an error comparison between vector-based and integral-based error measurements.

Lamé problem of elasticity

In the first example, we study a benchmark example without contact, namely the well-known Lamé problem of a cylinder, subjected to an internal pressure p (see Fig. 9a). Since the problem is geometrically axisymmetric and the internal pressure is applied to the entire inner boundary, only a quarter of the annulus is considered. The analytical solution for the stress and displacement field can be derived in polar coordinates $\{r, \alpha\}$ as [55]

$$\begin{aligned} \sigma_{rr} &= \frac{R_i^2 p}{R_o^2 - R_i^2}\left(1 - \frac{R_o^2}{r^2}\right), \\ \sigma_{\alpha\alpha} &= \frac{R_i^2 p}{R_o^2 - R_i^2}\left(1 + \frac{R_o^2}{r^2}\right), \\ \sigma_{r\alpha} &= 0, \\ u_r &= \frac{(1+\nu) p R_i^2 R_o^2}{E(R_o^2 - R_i^2)}\left(1 + \frac{r}{R_o^2}\right). \end{aligned} \tag{32}$$

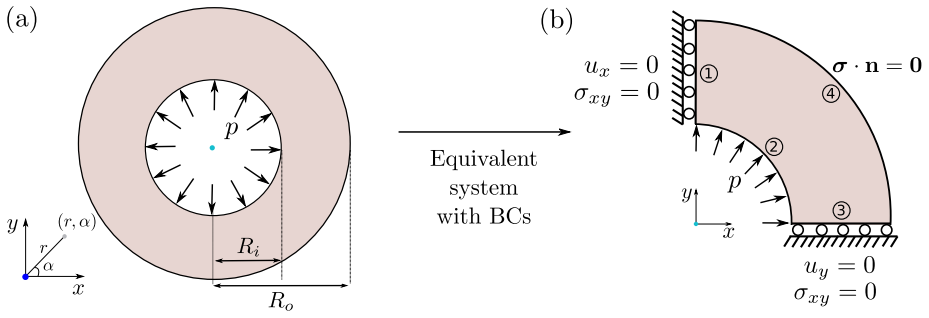

Fig. 9 The Lamé problem of elasticity. **a** The full problem under internal pressure, and **b** the equivalent problem due to the axisymmetrical nature of both geometry and loading, and accompanying boundary conditions

Table 1 The structure of hidden layers, number of training and test points, performance measurements, and errors for the Lamé problem of linear elasticity

	Hidden layers	# Training points	# Test points	Training time (s)	Prediction time (s)	$E^{\boldsymbol{u}}_{L_2}$ (%)	$E^{\boldsymbol{\sigma}}_{L_2}$ (%)
Lamé problem of elasticity	3 × 50	330	7104	27.15	0.001	0.017	0.043

Here, R_i and R_o are the inner and outer radius of the annulus, respectively, p represents the internal pressure applied on the inner radius, E is the Young's modulus and ν is the Poisson's ratio. For our specific setup, we set $R_i = 1, R_o = 2, p = 1, E = 2000$, and $\nu = 0.3$. Note that our formulation is based on Cartesian coordinates. Thus, polar transformation is performed to compare results.

To solve the Lamé problem, we deploy a PINN in the mixed variable formulation (see Eq. 39). The following output transformation is applied to enforce displacement and traction BCs as hard constraints on the edges numbered 1 and 3:

$$\tilde{u}_x = \frac{x}{E}\mathcal{N}_{u_x}, \quad \tilde{u}_y = \frac{y}{E}\mathcal{N}_{u_y}, \quad \tilde{\sigma}_{xy} = xy\mathcal{N}_{\sigma_{xy}}. \tag{33}$$

As can be seen in Fig. 9, the constructed output transformation fulfills the displacement boundary conditions at $x = 0$ and $y = 0$. Also, the displacement outputs are scaled by $1/E$ to ease the optimization process (see "Importance of output scaling" section). Traction boundary conditions are enforced as hard constraints for edges numbered 1 and 3, which lets us represent zero shear stresses there. Since traction boundary conditions on edges 2 and 4 contain a coupling of normal and shear stresses, we enforce them as soft constraints.

The employed PINN is a fully connected neural network consisting of 3 hidden layers of 50 neurons each as indicated in Table 1. We train the network using 330 training points, of which 262 are located within the domain and the remaining 68 points are on the boundary. We refer to the introductory paragraph of Numerical examples" section for further settings of both network and optimizer.

Figure 10a, b show a comparison of the normalized stress and displacement solutions in radial direction. Relative L_2 errors for displacements and stresses are calculated on test points as 0.017% and 0.043%, respectively. While the network is trained with *Adam*, the convergence rate decreases along with epochs as shown in Fig. 10c. Applying *L-BFGS-B* just after *Adam* increases the convergence rates and leads to a further significant reduction of PDE and NBC losses. The average MSE for the PDE loss reaches approximately

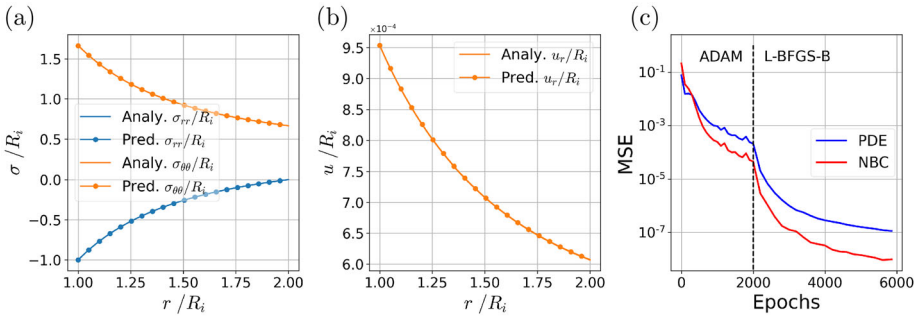

Fig. 10 Results for the Lamé problem of linear elasticity. **a** Comparison of normalized stresses obtained from the analytical solution and the predicted values, **b** comparison of normalized displacements, and **c** evolution of the MSE for the cumulative PDE loss and NBC loss

Table 2 Additional version of the Lamé problem with a challenging parameter set, selected scaling constants and computed L_2 errors

Parameter			Scaling constant		Error	
E	ν	p	$\tilde{u}_x, \tilde{u}_y$	$\tilde{\sigma}_{xy}, \tilde{\sigma}_{yy}, \tilde{\sigma}_{xy}$	$E^{\boldsymbol{u}}_{L_2}$ (%)	$E^{\boldsymbol{\sigma}}_{L_2}$ (%)
210×10^9	0.25	1.5×10^5	$\frac{p}{E}$	p	0.023	0.048

1.06e−7, while the average MSE for the NBC loss is approximately 1.48e−8 when all stopping criteria are met. We observe that deploying *Adam* and *L-BFGS-B* optimizers in a sequential order is one of the key points to obtain a good accuracy since *Adam* avoids rapid convergence to a local minimum, which has also been mentioned in [52]. Using a standard multi-core workstation as hardware, training takes 27.15 s and prediction takes 0.001 s.

An additional version of the Lamé example is generated to demonstrate that PINNs with output transformation (see Eq. 18) can effectively tackle problems with large-scale parameters, e.g. $E = 210 \times 10^9$, $\nu = 0.25$ and $p = 1.5 \times 10^5$. As provided in Table 2, relative L_2 errors for displacements and stresses are computed on test points as 0.023% and 0.048%. These results are very similar to the Lamé example with small-scale parameters since appropriate scaling constants are chosen, which dispels the common misconception that PINNs cannot handle very different magnitudes of the parameters involved. Thus, problems characterized by varying scales including those introduced by nonlinear contact terms can be effectively addressed through PINNs with proper output scaling. We refer to Appendix 2 for a comparison of the normalized stress and displacement solutions in radial direction, and the evolution of the cumulative PDE loss and NBC loss.

Contact between an elastic block and a rigid surface

The second example is a contact problem between a linear-elastic block and a rigid flat surface as depicted in Fig. 11a. The elastic block is subjected to an external pressure on its top surface and constrained in the horizontal direction on its left surface. The analytical solution [56] can easily be derived as follows

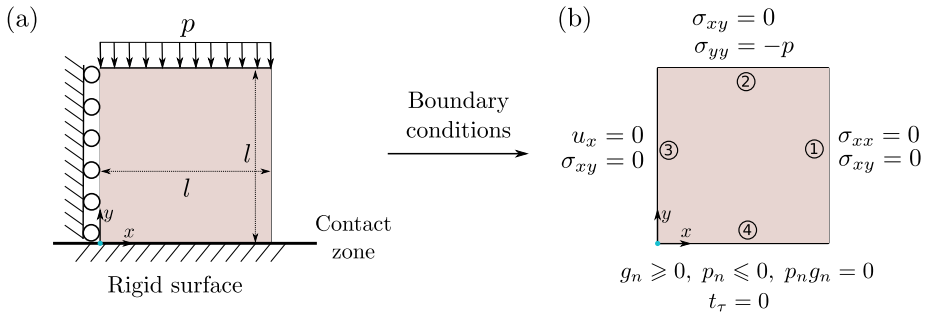

Fig. 11 Contact problem between an elastic block and a rigid flat surface, **a** geometry, supports and loading, and **b** an equivalent system including all relevant boundary conditions

$$u_x = \frac{-p}{E}\nu(1+\nu)x, \quad u_y = \frac{p}{E}(1-\nu^2)y,$$
$$\text{and}$$
$$\sigma_{yy} = -p, \quad \sigma_{xy} = 0.$$

For our specific setup, we set the material parameters as $E = 1.33$, $\nu = 0.33$, the edge length of the square block as $l = 1$ and the pressure as $p = 0.1$.

Similar as before, we apply the following output transformation to enforce displacement and traction boundary conditions on the edges numbered 1, 2 and 3 as

$$\begin{aligned} \tilde{u}_x &= x\mathcal{N}_{u_x}, \quad \tilde{\sigma}_{xx} = (l-x)\mathcal{N}_{\sigma_{xx}}, \\ \tilde{\sigma}_{yy} &= -p + (l-y)\mathcal{N}_{\sigma_{xy}}, \quad \tilde{\sigma}_{xy} = x(l-y)(l-x)\mathcal{N}_{\sigma_{xy}}. \end{aligned} \tag{34}$$

These output transformations can easily be derived based on Fig. 11b. For instance, the normal stress σ_{yy} in the loading direction is equal to $-p$ at $y = l$. Thus, we choose $g(\boldsymbol{x}) = -p$, and $h(\boldsymbol{x}) = (l - y)$ so that the requirements given in Eq. 19 are fulfilled. On the other hand, the contact constraints at the bottom edge are enforced as soft constraints using the three different methods that have been explained earlier in "Enforcing the Karush–Kuhn–Tucker inequality constraints" section: the sign-based method, the Sigmoid-based method and the *Fischer–Burmeister* NCP function. For the Sigmoid-based method, we choose $\delta_{g_n} = 10$ and $\delta_{p_n} = 100$. For training, 514 points are used (434 points lie within the domain and 80 points lie on the boundary), and 11,827 points are used for testing.

As illustrated in Fig. 12, all of the investigated methods correctly capture that u_y is linearly distributed and close to zero at the bottom due to the soft enforcement of contact constraints. The maximum absolute error for the displacement component u_y, denoted by $E^{u_y}_{abs,max}$, is larger for the sign-based formulation compared to the two other methods (see Table 3). Meanwhile, the normal stress component, σ_{yy}, is close to -0.1 and the shear stress component σ_{xy} is close to zero. Since the traction boundary condition in y-direction on the top surface and the shear stress boundary conditions are enforced as hard constraints, absolute errors in the corresponding regions are zero for all cases. The sign-based formulation also performs worst in terms of the errors $E^{\sigma_{yy}}_{abs,max}$ and $E^{\sigma_{xy}}_{abs,max}$. Moreover, L_2 relative errors show that the *Fischer–Burmeister* NCP function performs

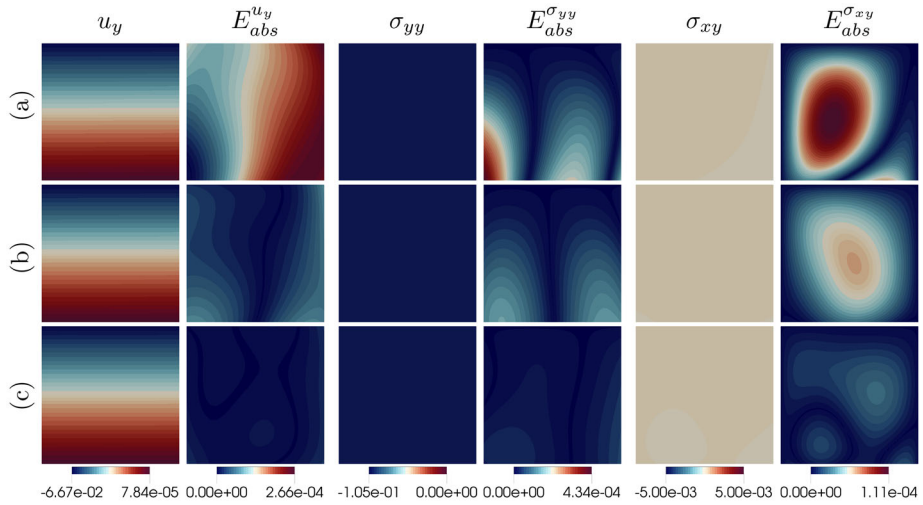

Fig. 12 Predictions of the displacement component u_y, stresses σ_{yy} and σ_{xy} with corresponding absolute errors $E^{u_y}_{abs}, E^{\sigma_{yy}}_{abs}, E^{\sigma_{xy}}_{abs}$. The contact constraints $\mathcal{L}_{\text{KKT}}$ are enforced via three different methods: **a** sign-based method, **b** Sigmoid-based method and **c** the *Fischer–Burmeister* NCP function. Absolute error is defined as $E^{*}_{abs} = \text{abs}(\dot{*} - *)$

Table 3 The structure of hidden layers, performance measurements, and errors for the contact example between an elastic block and a rigid surface

	Hidden layers	Training time (s)	Prediction time (s)	$E^{\boldsymbol{u}}_{L_2}$ (%)	$E^{\boldsymbol{\sigma}}_{L_2}$ (%)	$E^{u_y}_{abs,max}$	$E^{\sigma_{yy}}_{abs,max}$	$E^{\sigma_{xy}}_{abs,max}$
(a)	5 × 50	20.15	0.388	0.382	0.154	2.66e−4	4.34e−4	1.11e−4
(b)	5 × 50	20.14	0.389	0.090	0.094	8.23e−5	1.51e−4	6.30e−5
(c)	5 × 50	18.20	0.383	0.024	0.031	2.71e−5	6.10e−5	2.55e−5

Three different methods to enforce the KKT constraints are compared: (a) sign-based, (b) Sigmoid-based and (c) Fischer–Burmeister

best, i.e. with errors being up to one order of magnitude smaller than for the sign-based and Sigmoid-based variants. Additionally, it is observed that all investigated methods require similar computing time for training and prediction. Overall, it can be concluded that the *Fischer–Burmeister* NCP function yields the best results in terms of accuracy and computing time.

Hertzian contact problem

In this example, we consider a long linear elastic half-cylinder ($E = 200$, $\nu = 0.3$) lying on a rigid flat surface and being subjected to a uniform pressure $p = 0.5$ on its top surface as shown in Fig. 13a. The analytical solution for the contact pressure p_c is given as [57,58]

$$p_c = \frac{4Rp}{\pi b^2}\sqrt{b^2 - x^2} \quad \text{with } b = 2\sqrt{\frac{2R^2p(1-\nu^2)}{E\pi}}. \tag{35}$$

Here, b is the width of the contact zone, and R is the radius of the cylinder. For the chosen set of parameters ($p = 0.5$, $E = 200$, $\nu = 0.3$, $R = 1$), b can be calculated as 0.076. An analytical solution is available only for the contact pressure. Reference solutions for displacement and stress fields within the half-cylinder are obtained with well-established FEM algorithms. The FEM simulations are performed with our in-house multi-physics research code BACI [59].

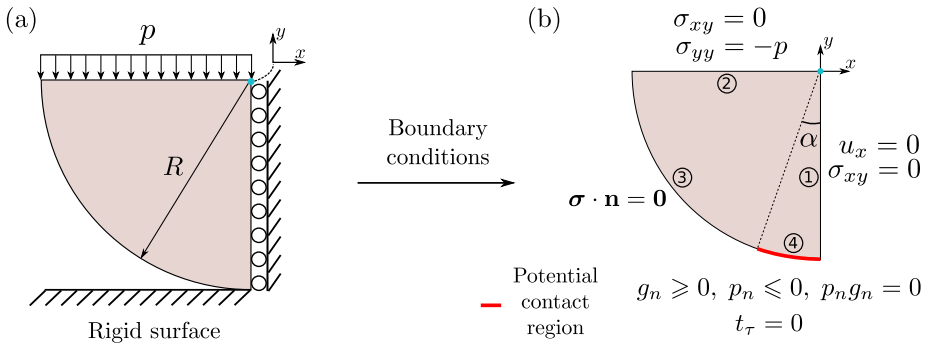

Fig. 13 The Hertzian contact problem between an elastic half-cylinder and a rigid flat surface **a** domain under uniform pressure on top and making use of symmetry, **b** accompanying boundary conditions

Table 4 The structure of hidden layers, number of input neurons, training and test points, and loss weights are given for different cases of the Hertzian contact problem

	Hidden layers	# Input neurons	# Training points	# Test points	Loss weights
Case 1	5 × 50	2	47935	26185	w^{KKT}=10^3
Case 2, 3	5 × 50	2	47935	26185	$w^{\text{KKT}} = 10^3$, $w_1^{\text{EXPs}} = 10^4$, $w_2^{\text{EXPs}} = 10^4$, $w_3^{\text{EXPs}} = 10^{-1}$, $w_4^{\text{EXPs}} = 10^{-1}$, $w_5^{\text{EXPs}} = 10^{-1}$
Case 4	8 × 75	3	1946	604	$w^{\text{KKT}} = 10^4$

The remaining loss weights are set as 1. See Appendix 1 for a detailed explanation of loss weights

The following output transformation is applied to enforce displacement and traction BCs as hard constraints on the edges numbered 1 and 2 (see Fig. 13b)

$$\tilde{u}_x = \frac{-x}{E}\mathcal{N}_{u_x}, \quad \tilde{u}_y = \frac{1}{E}\mathcal{N}_{u_y}, \quad \tilde{\sigma}_{yy} = -p + (-y)\mathcal{N}_{\sigma_{yy}}, \quad \tilde{\sigma}_{xy} = xy\mathcal{N}_{\sigma_{xy}}. \tag{36}$$

As for the Lamé problem, we scale the displacement field with the inverse of the Young's modulus $1/E$. Additionally, only one non-zero helper function $g(x) = -p$ is required for $\tilde{\sigma}_{yy}$. The traction BC on edge 3 and the contact constraints on edge 4 are enforced as soft constraints. To define the potential contact area, we set $\alpha = 15°$ (corresponding to $b = 0.259$), which extends well beyond the actual contact area. Moreover, KKT constraints are enforced using the *Fischer–Burmeister* method.

In the following, we investigate four distinct PINN application cases for the Hertzian contact problem: *Case 1*: PINNs as pure forward model/PDE solver, *Case 2*: PINNs as data-enhanced forward model, *Case 3*: PINNs as inverse solver for parameter identification, and *Case 4*: PINNs as a fast-to-evaluate surrogate model. We refer to Table 4 for information on network architecture, and training and test points. To facilitate reproduction of the results, the table also reports the weights of individual loss terms.

Case 1: PINNs as pure forward model/PDE solver

In the first use case, we deploy PINNs as a pure forward solver for contact problems to validate our approach. Training takes a total of 4049.8 s and the prediction time is 0.091 s. Note that FE simulation of the same test case requires 19.2 s, a considerably longer duration compared to the prediction time of a trained PINN since the most time-consuming aspect of PINNs is the training process. Displacement and stress components obtained

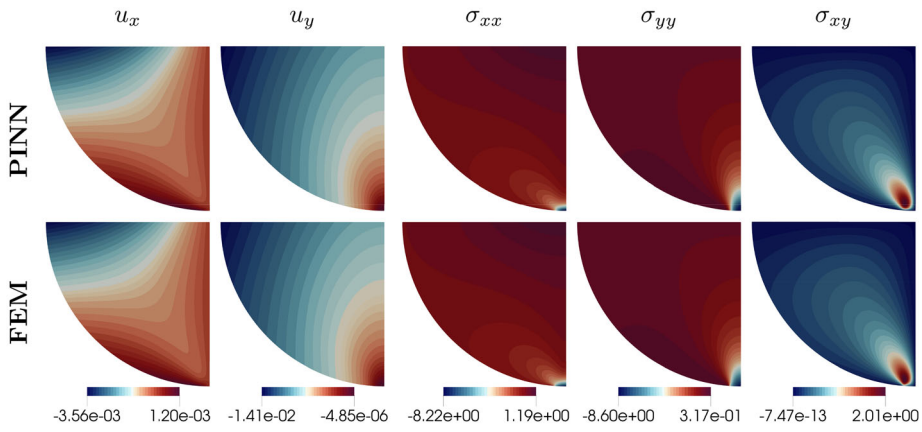

Fig. 14 PINNs as pure forward solver for the Hertzian contact problem. Comparison of stress and displacement components obtained by PINN and FEM

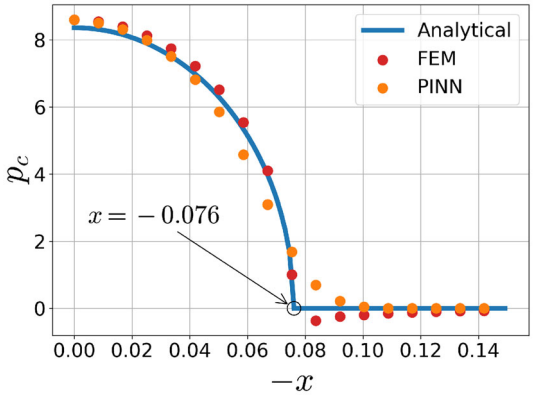

Fig. 15 Comparison of contact pressure distributions $p_c(x)$ obtained by analytical solution, PINN and FEM for case 1

through PINN and FEM are compared with contour plots in Fig. 14. Errors for displacement and stress fields are quantified using the relative L_2 norm. As summarized in Table 5, we obtain a relative error $E^{\boldsymbol{u}}_{L_2} = 2.24\%$ for the displacement field and a relative error $E^{\boldsymbol{\sigma}}_{L_2} = 3.74\%$ for the stress field. Furthermore, the contact pressure distributions obtained via analytical solution, PINN and FEM are compared in Fig. 15. Since the zero traction boundary condition and the KKT constraints on the curved surface, numbered as edge 3 in Fig. 13, are not enforced as hard but soft constraints, the PINN result can only resolve the kink at $x = \pm 0.076$ in an approximate manner depending on the chosen set of training points. Consequently, the contact pressure p_c reduces to zero smoothly and slightly violates the zero traction boundary condition in the non-contact zone. Accordingly, the rather large error values $E^{\boldsymbol{\sigma}}_{L_2}$ are mostly related to this violation of the zero traction boundary conditions and KKT constraints close to the kink. Readers familiar with FEM modeling of contact problems will notice that a quite similar phenomenon occurs for mesh-based numerical methods where the transition between contact and non-contact zones cannot be perfectly resolved either and might even cause spurious oscillations in the case of higher-order interpolation functions [24,60,61].

Table 5 Comparison of relative L_2 errors, training and prediction time for case 1 and case 2

	$E^{\boldsymbol{u}}_{L_2}$ (%)	$E^{\boldsymbol{\sigma}}_{L_2}$ (%)	Training time (s)	Prediction time (s)	$\int_{\Omega_c} \tilde{p}_c$
Case 1	2.24	3.74	4049.8	0.091	0.4993
Case 2	0.11	2.79	7126.7	0.092	0.4978

The term $\int_{\Omega_c} \tilde{p}_c$ denotes an integral of the predicted contact pressure $\tilde{p}_c$ over the potential contact boundary

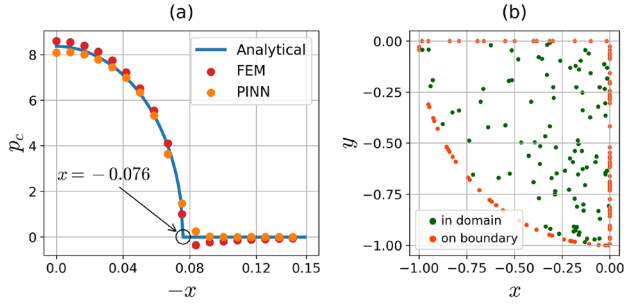

Fig. 16 Comparison of contact pressure distributions $p_c(x)$ obtained by analytical solution, PINN and FEM for case 2. **a** PINNs as a data-enhanced forward solver, **b** distribution of data points generated through FEM simulations in the domain and on the boundary

Case 2: PINNs as data-enhanced forward model

One of the key features of PINNs is the capability of easily incorporating external data, such as measurement or simulation data, into the overall loss function. In this section, we enhance our PINN model with "artificial" measurement data obtained through FEM simulations, namely, data points for displacement and stress fields, to achieve better accuracy. The incorporated FEM data points are randomly selected, with 100 being selected within the domain and 100 being chosen along the boundary as depicted in Fig. 16b.

A comparison of the contact pressure p_c in the case of data enhancement with analytical solution and FEM reference solution is given in Fig. 16a. While the PINN accuracy is significantly improved upon close to the kink at $x = \pm 0.076$, the data-enhanced model underestimates the normal contact pressure around the origin. The relative L_2 errors confirm this assessment. While $E^{\boldsymbol{\sigma}}_{L_2}$ is only slightly reduced, $E^{\boldsymbol{u}}_{L_2}$ benefits dramatically from data enhancement. As provided in Table 5, the integrated contact pressure is close to the applied load, $p = 0.5$, due to the conservation of momentum. To fulfill the momentum equation, overshooting after the kink is balanced by undershooting around the origin. Similarly, in case 1, overshooting after the kink is balanced by the undershooting from around $x = 0.04$ to the kink. Moreover, we observe that results significantly rely on selecting appropriate additional data loss weights w^{EXPs} (reported in Table 4). So far we identified the parameters through manual adjustments. In general, we recommend a hyperparameter analysis to determine optimal loss weights. Additionally, the data-enhanced model requires more training time compared to the pure forward model, which can be explained by the need to evaluate additional loss terms. However, after training is finished, the more accurate prediction takes essentially the same time as in case 1.

Case 3: PINNs as inverse solver for parameter identification

An interesting approach to solve an inverse problem is to simply add the unknown parameter to the set of network trainable parameters $\boldsymbol{\theta}$, and it can then be identified with the help

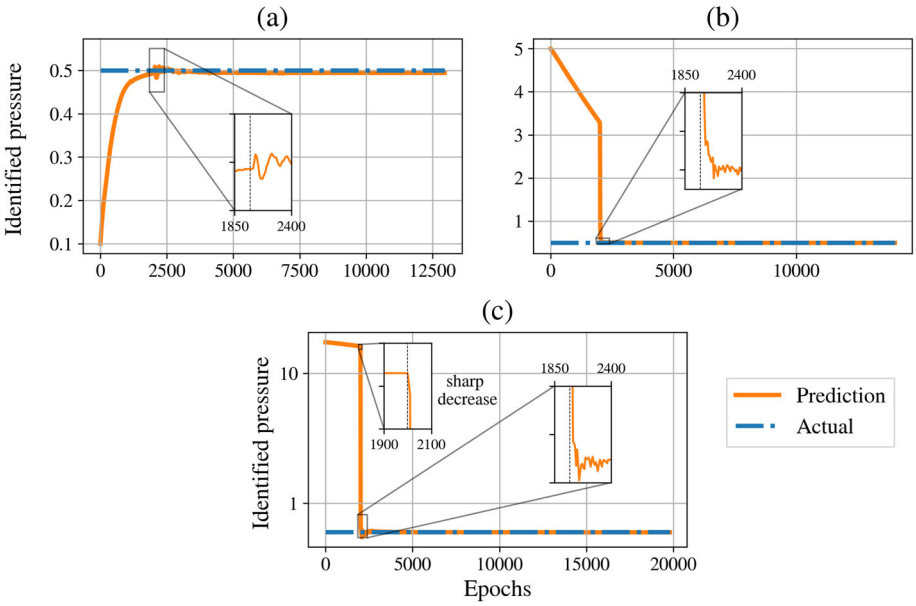

Fig. 17 Identification of the applied external pressure on the half-cylinder in case of different initial guesses **a** $p_o = 0.1$, **b** $p_o = 5$, **c** $p_o = 20$

of additional loss terms based on the difference between predictions and observations (see Eq. 37). In the following, we exemplarily identify the applied external pressure p acting on the half-cylinder using FEM results as "artificial" measurement data.

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} \mathcal{L}_C(\boldsymbol{\Theta}) \quad \text{where,} \quad \boldsymbol{\Theta} = (p, \boldsymbol{\theta}). \tag{37}$$

Figure 17 shows the convergence behavior of the identified pressure $\tilde{p}$ compared to the actual pressure p through the number of epochs for different initial guesses $\tilde{p}_o$. First, we start training with a "good" initial guess $\tilde{p}_o = 0.1$ being quite close to the actual pressure $p = 0.5$, and then we increase it to $\tilde{p}_o = 20$ to measure how sensitive the PINN is to the choice of the initial guess. As depicted in Fig. 17c, convergence can be achieved even when a relatively large and therefore unphysical initial guess is made. There is a steep increase in convergence rate after 2000 epochs, which can be explained by the fact that we switch to the *L-BFGS-B* optimizer there. As mentioned in "Lamé problem of elasticity" section, we deploy *Adam* for 2000 epochs to avoid the optimization process getting stuck in local minima. Note, however, that switching from *Adam* to *L-BFGS-B* introduces oscillations in the transition region so that transition has to be handled with care. The relative error, denoted as E, is used to measure the difference between the identified and actual pressure values, resulting in the same value of 1.2% for all three initial guesses $p_o = 0.1$, $p_o = 5$, and $p_o = 20$ as provided in Table 6. Additionally, a larger initial guess requires more computing time and epochs as expected.

Case 4: PINNs as fast-to-evaluate surrogate model

In the last use case, the load (applied external pressure) is considered as an additional network input, i.e. $\boldsymbol{z} = (\boldsymbol{x}, p)$ (compare Eq. 16), to construct a fast-to-evaluate surrogate model that is capable of predicting displacement and stress fields for different pressure values.

Table 6 Comparison of relative errors, training time and number of epochs for the inverse Hertzian problem used to identify the applied external pressure

	Initial guess $\tilde{p}_o$	Identified $\tilde{p}_f$	Relative error E(%)	Training time (s)	# Epochs
Case 3	0.1	0.494	1.2	3472.8	12958
	5	0.494	1.2	3762.2	14019
	20	0.494	1.2	5394.1	19793

The relative error is defined as $E^* = \text{abs}((* - *)/*)$

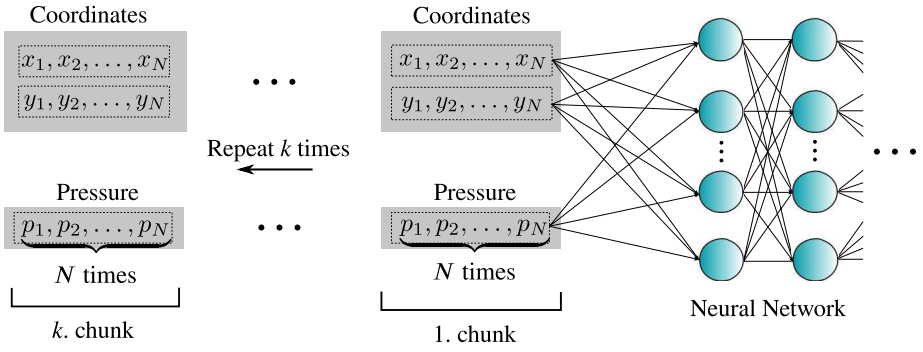

Fig. 18 An illustration of the procedure to include the applied external pressure as a neural network input

Since a single network is deployed, the length of each network input must be the same. We sample the three-dimensional input space with $N = 1946$ training points ($N = N_{rp} + N_{bp}$ (Eq. 23)). While the spatial coordinates $\boldsymbol{x}$ are selected from a two-dimensional mesh (as in "Case 1: PINNs as pure forward model/PDE solver", "Case 2: PINNs as data-enhanced forward model", and "Case 3: PINNs as inverse solver for parameter identification" sections), the third (pressure) component of the input vector is drawn randomly from a uniform distribution over the considered pressure range. To improve the accuracy of the prediction, the N sampling points are repeated k times. In the context of network training, we refer to one instance of the N distinct sampling points as one *chunk*, so that k is the number of chunks as depicted in Fig. 18. Indeed, this process increases the computing time and complexity of the model, since the input size increases from $(n, 2)$ to $(n \cdot k, 3)$. However, such a method can lead to better accuracy since the network is trained with a larger data set and also it enables batch training to reduce computational effort [47]. For our specific example, we sample pressure values from a range of $[0.2, 1.0]$ and we consider only two different numbers of chunks, namely $k = 1$ and $k = 5$.

As shown in Fig. 19, employing a single chunk is insufficient to accurately capture the influence of the applied external pressure p on the contact pressure distribution $p_c(\boldsymbol{x}, p)$. However, increasing the number of chunks from $k = 1$ to $k = 5$ increases the accuracy. Table 7 provides the relative L_2 errors for the contact pressure distribution $p_c(\boldsymbol{x}, p)$ between the analytical solution and the predictions of surrogate PINN models, and it can be seen that increasing the number of chunks indeed results in improved accuracy of the surrogate model. Nonetheless, the overall error level of 10–16% even in the case $k = 5$ is still too high from an engineering perspective and will be subject to further investigations. This example is only intended as a very first proof of concept, and we have already identified several algorithmic modifications that could possibly increase the accuracy of the surrogate model. For example, we use fixed loss weights even though pressure values in the input layer vary. Thus, adaptive loss weights should be implemented since the PINN

Table 7 Comparison of relative L_2 errors for the contact pressure distribution $p_c(\boldsymbol{x}, p)$ between the analytical solution and the predictions of our surrogate PINN models. Predictions are based on unseen pressure values

		Applied pressure		
		$p = 0.45$	$p = 0.98$	$p = 1.5$
Number of chunks	$k = 1$	22.81%	17.23%	16.93%
	$k = 5$	16.30%	11.47%	10.87%

Predictions are based on unseen pressure values

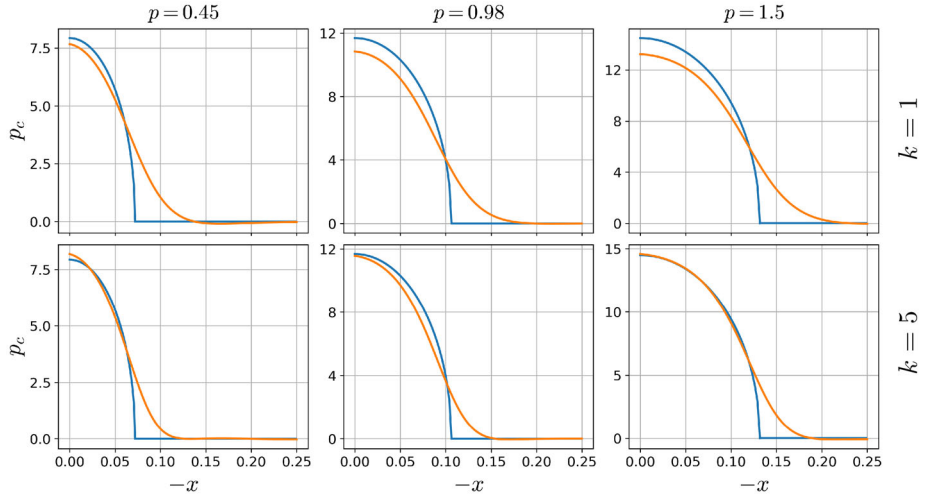

Fig. 19 Comparison of the contact pressure distribution obtained through a PINN-based surrogate model to the analytical solution. The PINNs are trained using different numbers of chunks

accuracy highly depends on choosing appropriate loss weights. Additionally, the accuracy of PINNs could be further improved by increasing the number of chunks.

Conclusion

In this study, we have presented an extension of physics-informed neural networks (PINNs) for solving forward and inverse problems of contact mechanics under the assumption of linear elasticity. The framework has been tested on several benchmark examples with different use cases, e.g. the Hertzian contact problem, and has been validated by existing analytical solutions or numerical simulations using the finite element method (FEM). As an alternative way of soft constraint enforcement as compared to existing methods, a nonlinear complementarity problem (NCP) function, namely *Fischer–Burmeister*, is explored and exploited to enforce the inequality constraints inherent to contact problems. This aspect has not been investigated in the context of PINNs so far to the best of the authors' knowledge. Besides using PINNs as pure forward PDE solver, we show that PINNs can serve as a hybrid model enhanced by experimental and/or simulation data to identify unknown parameters of contact problems, e.g. the applied external pressure. We even go one step further and deploy PINNs as fast-to-evaluate surrogate models, and could at least obtain a first proof of concept up to a certain level of accuracy.

A question that has emerged recently is whether data-driven approaches such as PINNs will replace classical numerical methods such as FEM in the near future. Within this study, we only considered benchmark examples that have been developed and solved decades

ago using the FEM. Even for these simple examples, we came to the conclusion that deploying PINNs as forward solvers for contact mechanics can not compete with FEM in terms of computational performance and accuracy. Therefore, we doubt the applicability of PINNs to complex engineering problems without data enhancement. However, PINNs can be a good candidate for solving data-enhanced forward problems and especially inverse problems due to the easy integration of additional data. Similarly, PINNs can break the curse of dimensionality of parametric models, so that more complex surrogate models can be generated. Also, it is observed that minimizing multiple loss functions simultaneously is one of the most significant challenges in training PINNs, and current optimization algorithms are not tailored to addressing this challenge. Therefore, using multi-objective optimization algorithms that are particularly designed for PINNs has the potential to be a gamechanger in improving their overall performance and accuracy. As an alternative to multi-objective optimization algorithms, it is observed that a proper scaling strategy via output transformation can be applied to ease the optimization. We believe that hybrid strategies can be a promising option to construct mixed models to benefit from the advantages of both classical and data-driven approaches.

This study reveals several possibilities for further exploration and investigation. Although the proposed PINN formulation for benchmark examples demonstrates acceptable results, further applications, particularly on complex domains including three-dimensional problems should be analyzed. As an alternative strategy to scaling network outputs, a non-dimensionalized contact formulation can be implemented. Different NCP functions other than the *Fischer–Burmeister* function can be further investigated. The inverse solver has been applied to identify the applied external pressure, but it can be extended to also predict internal material parameters. Additionally, a hyperparameter optimization study can be performed to tune loss weights, network architecture and optimizer parameters. Last but not least, related techniques such as variational PINNs might overcome the limitations of collocation inherent to PINNs and instead provide a sound variational framework.

Acknowledgements
The authors gratefully acknowledge the computing resources provided by the Data Science & Computing Lab at the University of the Bundeswehr Munich.

Author contributions
T. Sahin: methodology, software, visualization, investigation, writing—original draft, writing—review and editing. M. v. Danwitz: conceptualization, methodology, software, validation, writing— review and editing. A. Popp: conceptualization, resources, methodology, supervision, writing—review and editing.

Funding
Open Access funding enabled and organized by Projekt DEAL. This research paper is funded by dtec.bw—Digitalization and Technology Research Center of the Bundeswehr [project RISK.twin]. dtec.bw is funded by the European Union—NextGenerationEU.

Availability of data and materials
The datasets generated and analysed during the current study, as well as the source code, are publicly available at https://github.com/imcs-compsim/pinns_for_comp_mech.

Declarations

Competing interests
The authors declare that they have no conflict of interest.

Appendix 1: Loss formulation for 2D linear isotropic elasticity under plane strain conditions

The elasticity tensor $\mathbb{C}$ under plane strain conditions can be expressed in terms of Lamé constants λ and μ as

$$\mathbb{C} = \begin{bmatrix} 2\mu+\lambda & \lambda & 0 \\ \lambda & 2\mu+\lambda & 0 \\ 0 & 0 & \mu \end{bmatrix}. \tag{38}$$

Inserting Eq. 38 into Eq. 23 we can obtain the total loss for 2D linear isotropic elasticity under the plane strain condition as

$$\begin{aligned}
\mathcal{L}_{\mathrm{E}} &= \mathcal{L}_{\mathrm{PDEs}} + \mathcal{L}_{\mathrm{DBCs}} + \mathcal{L}_{\mathrm{NBCs}} + \mathcal{L}_{\mathrm{EXPs}} \\
&= w_1^{(\mathrm{PDEs})} \left|\tilde{\sigma}_{xx,x} + \tilde{\sigma}_{xy,y} + \hat{b}_x\right|_\Omega + w_2^{(\mathrm{PDEs})} \left|\tilde{\sigma}_{yx,x} + \tilde{\sigma}_{yy,y} + \hat{b}_y\right|_\Omega + \\
&\quad w_3^{(\mathrm{PDEs})} \left|\tilde{\sigma}_{xx} - (\lambda+2\mu)\tilde{\varepsilon}_{xx} - \lambda\tilde{\varepsilon}_{yy}\right|_\Omega + w_4^{(\mathrm{PDEs})} \left|\tilde{\sigma}_{yy} - \lambda\tilde{\varepsilon}_{xx} - (\lambda+2\mu)\tilde{\varepsilon}_{yy}\right|_\Omega + \\
&\quad w_5^{(\mathrm{PDEs})} \left|\tilde{\sigma}_{xy} - 2\mu\tilde{\varepsilon}_{xy}\right|_\Omega + \\
&\quad w_1^{(\mathrm{DBCs})} \left|\tilde{u}_x - \hat{u}_x\right|_{\partial\Omega_D} + w_2^{(\mathrm{DBCs})} \left|\tilde{u}_y - \hat{u}_y\right|_{\partial\Omega_D} + \\
&\quad w_1^{(\mathrm{NBCs})} \left|\tilde{\sigma}_{xx} n_x + \tilde{\sigma}_{xy} n_y - \hat{t}_x\right|_{\partial\Omega_N} + w_2^{(\mathrm{NBCs})} \left|\tilde{\sigma}_{yx} n_x + \tilde{\sigma}_{yy} n_y - \hat{t}_y\right|_{\partial\Omega_N} + \\
&\quad w_1^{(\mathrm{EXPs})} \left|\tilde{u}_x - u_x^*\right|_{\Omega_e} + w_2^{(\mathrm{EXPs})} \left|\tilde{u}_y - u_y^*\right|_{\Omega_e} + w_3^{(\mathrm{exps})} \left|\tilde{\sigma}_{xx} - \sigma_{xx}^*\right|_{\Omega_e} + \\
&\quad w_4^{(\mathrm{EXPs})} \left|\tilde{\sigma}_{yy} - \sigma_{yy}^*\right|_{\Omega_e} + w_5^{(\mathrm{EXPs})} \left|\tilde{\sigma}_{xy} - \sigma_{xy}^*\right|_{\Omega_e}
\end{aligned} \tag{39}$$

including kinematics

$$\tilde{\varepsilon}_{xx} = \tilde{u}_{x,x}, \quad \tilde{\varepsilon}_{yy} = \tilde{u}_{y,y}, \quad \tilde{\varepsilon}_{xy} = \frac{1}{2}(\tilde{u}_{x,y} + \tilde{u}_{y,x}), \tag{40}$$

where

$$\begin{aligned}
\tilde{u}_x \approx (\mathcal{N}_{u_x}(x,y))', \quad \tilde{u}_y \approx (\mathcal{N}_{u_y}(x,y))', \\
\tilde{\sigma}_{xx} \approx (\mathcal{N}_{\sigma_{xx}}(x,y))', \quad \tilde{\sigma}_{yy} \approx (\mathcal{N}_{\sigma_{yy}}(x,y))', \quad \{\tilde{\sigma}_{xy} = \tilde{\sigma}_{yx}\} \approx (\mathcal{N}_{\sigma_{xy}}(x,y))'.
\end{aligned} \tag{41}$$

The out-of-plane stress component σ_{zz} is not considered as network output since it can be calculated in the post-processing.

Appendix 2: Results for the Lamé problem of elasticity with large-scale parameters

See Fig. 20.

Appendix 3: Additional error comparisons

The vector-based L_2 error between the approximated PINN solution $\tilde{f}$ and the analytical solution f, is denoted as,

$$E_{L_2}^f := \sqrt{\sum_{j=1}^{N_{\mathrm{test}}} \left(\tilde{f}(\boldsymbol{x}^j) - f(\boldsymbol{x}^j)\right)^2}. \tag{42}$$

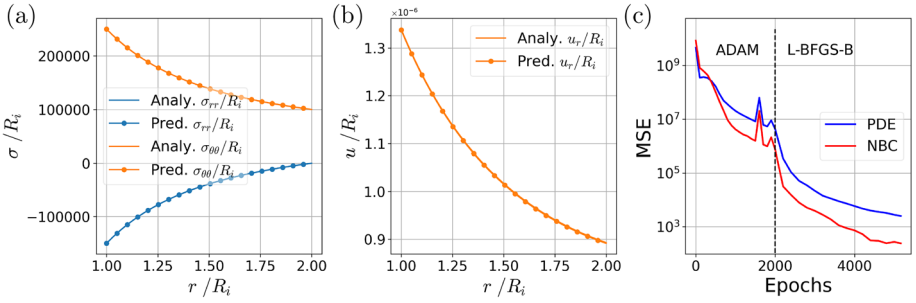

Fig. 20 Visualization of results for the Lamé problem of linear elasticity with large-scale parameters. **a** Comparison of normalized stresses obtained from the analytical solution and the predicted values, **b** comparison of normalized displacements, and **c** evolution of the MSE for the cumulative PDE loss and NBC loss

Table 8 Comparison of the vector-based and integral-based L_2 error for the Lamé problem of elasticity

Vector-based				Integral-based			
$E_{L_2}^{u_r}$	$E_{L_2}^{\sigma_{rr}}$	$E_{L_2}^{\sigma_{\theta\theta}}$	$E_{L_2}^{\sigma_{r\theta}}$	$\xi_{L_2}^{u_r}$	$\xi_{L_2}^{\sigma_{rr}}$	$\xi_{L_2}^{\sigma_{\theta\theta}}$	$\xi_{L_2}^{\sigma_{r\theta}}$
1.06e−5	0.012	0.035	0.008	8.80−e6	0.010	0.030	0.006

The evaluated quantities are displacement and stress components in polar coordinates (see "Lamé problem of elasticity" section for the example setup)

Table 9 Comparison of the vector-based and integral-based L_2 error for the contact problem between an elastic block and the rigid domain

	Vector-based					Integral-based				
	$E_{L_2}^{u_x}$	$E_{L_2}^{u_y}$	$E_{L_2}^{\sigma_{xx}}$	$E_{L_2}^{\sigma_{yy}}$	$E_{L_2}^{\sigma_{xy}}$	$\xi_{L_2}^{u_x}$	$\xi_{L_2}^{u_y}$	$\xi_{L_2}^{\sigma_{xx}}$	$\xi_{L_2}^{\sigma_{yy}}$	$\xi_{L_2}^{\sigma_{xy}}$
Sign	3.98e−3	1.753e−2	1.05e−2	1.166e−2	5.68e−3	3.92e−3	1.750e−2	1.04e−2	1.156e−2	5.74e−3
Sigmoid	1.91e−3	3.79e−3	5.89e−3	7.419e−3	3.86e−3	1.85e−3	3.75e−3	5.81e−3	7.421e−3	3.91e−3
Fischer–Burmeister	8.46e−4	8.04e−4	2.28e−3	2.10e−3	1.20e−3	8.20e−4	7.80e−4	2.26e−3	2.08e−3	1.22e−3

Evaluated quantities are displacement and stress component in cartesian coordinates. Three different methods are provided to enforce KKT constraints (see "Contact between an elastic block and a rigid surface" section for the example setup)

The corresponding integral-based L_2 error between the approximated PINN solution $\tilde{f}$ and the analytical solution f, is denoted as,

$$\xi_{L_2}^{f} := \sqrt{\frac{N_{\text{test}}}{\int_\Omega d\boldsymbol{x}} \int_\Omega \left(\tilde{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2 d\boldsymbol{x}}, \tag{43}$$

where $\int_\Omega d\boldsymbol{x}$ represents the area for Tables 8 and 9, while it represents the arc length for Table 10, since the contact pressure p_c is integrated over the potential contact boundary $\partial\Omega_c$. We refer to "Lamé problem of elasticity", "Contact between an elastic block and a rigid surface", "Hertzian contact problem" sections for the respective number of test points N_{test}. Note that in this first study, we used the vector-based error measure that is frequently used for PINNs [20,46] and easily implemented. For future studies, we suggest more expressive integral-based error estimates.

Table 10 Comparison of the vector-based and integral-based L_2 error for the contact pressure p_c of the Hertzian contact problem for different cases (see "Case 1: PINNs as pure forward model/PDE solver" and "Case 2: PINNs as data-enhanced forward model" sections for the example setup)

	Vector-based	Integral-based
	$E_{L_2}^{p_c}$	$\xi_{L_2}^{p_c}$
Case 1	2.22	2.48
Case 2	0.89	1.27

Received: 25 January 2024 Accepted: 8 April 2024
Published online: 03 May 2024

References

1. Raissi M, Perdikaris P, Karniadakis G. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys. 2019;378:686–707. https://doi.org/10.1016/j.jcp.2018.10.045.
2. Arend Torres F, Massimo Negri M, Nagy-Huber M, Samarin M, Roth V. Mesh-free Eulerian physics-informed neural networks. arXiv preprint. 2022. arXiv:2206.01545.
3. Grohs P, Hornung F, Jentzen A, von Wurstemberger P. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations, vol. 284. Providence: American Mathematical Society; 2023.
4. Poggio T, Mhaskar H, Rosasco L, Miranda B, Liao Q. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. Int J Autom Comput. 2017;14(5):503–19. https://doi.org/10.1007/s11633-017-1054-2.
5. Depina I, Jain S, Mar Valsson S, Gotovac H. Application of physics-informed neural networks to inverse problems in unsaturated groundwater flow. Georisk Assess Manag Risk Eng Syst Geohazards. 2022;16(1):21–36. https://doi.org/10.1080/17499518.2021.1971251.
6. Smith JD, Ross ZE, Azizzadenesheli K, Muir JB. HypoSVI: hypocentre inversion with stein variational inference and physics informed neural networks. Geophys J Int. 2022;228(1):698–710. https://doi.org/10.1093/gji/ggab309.
7. Rao C, Sun H, Liu Y. Physics-informed deep learning for incompressible laminar flows. Theor Appl Mech Lett. 2020;10(3):207–12. https://doi.org/10.1016/j.taml.2020.01.039.
8. Eivazi H, Vinuesa R. Physics-informed deep-learning applications to experimental fluid mechanics. arXiv preprint. 2022. arXiv:2203.15402.
9. Chen Y, Lu L, Karniadakis GE, Dal Negro L. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. Opt Express. 2020;28(8):11618–33. https://doi.org/10.1364/OE.384875.
10. Beltrán-Pulido A, Bilionis I, Aliprantis D. Physics-informed neural networks for solving parametric magnetostatic problems. IEEE Trans Energy Convers. 2022;37(4):2678–89. https://doi.org/10.1109/TEC.2022.3180295.
11. Khan A, Lowther DA. Physics informed neural networks for electromagnetic analysis. IEEE Trans Magn. 2022;58(9):1–4. https://doi.org/10.1109/TMAG.2022.3161814.
12. Yucesan YA, Viana FA. A hybrid physics-informed neural network for main bearing fatigue prognosis under grease quality variation. Mech Syst Signal Process. 2022;171: 108875. https://doi.org/10.1016/j.ymssp.2022.108875.
13. von Danwitz M, Kochmann TT, Sahin T, Wimmer J, Braml T, Popp A. Hybrid digital twins: a proof of concept for reinforced concrete beams. Proc Appl Math Mech. 2023;22(1): e202200146. https://doi.org/10.1002/pamm.202200146.
14. Brucherseifer E, Winter H, Mentges A, Mühlhäuser M, Hellmann M. Digital twin conceptual framework for improving critical infrastructure resilience. Automatisierungstechnik. 2021;69(12):1062–80. https://doi.org/10.1515/auto-2021-0104.
15. Haghighat E, Abouali S, Vaziri R. Constitutive model characterization and discovery using physics-informed deep learning. Eng Appl Artif Intell. 2023;120: 105828. https://doi.org/10.1016/j.engappai.2023.105828.
16. Bharadwaja B, Nabian MA, Sharma B, Choudhry S, Alankar A. Physics-informed machine learning and uncertainty quantification for mechanics of heterogeneous materials. Integr Mater Manuf Innov. 2022;11:1–21. https://doi.org/10.1007/s40192-022-00283-2.
17. Rao C, Sun H, Liu Y. Physics-informed deep learning for computational elastodynamics without labeled data. J Eng Mech. 2021;147(8):04021043. https://doi.org/10.1061/(ASCE)EM.1943-7889.0001947.
18. Zienkiewicz O. Displacement and equilibrium models in the finite element method by B. Fraeijs de Veubeke, chapter 9, pages 145–197 of stress analysis, edited by OC Zienkiewicz and GS Holister, published by John Wiley & Sons, 1965. Int J Numer Methods Eng. 2001;52(3):287–342. https://doi.org/10.1002/nme.339.
19. Samaniego E, Anitescu C, Goswami S, Nguyen-Thanh VM, Guo H, Hamdia K, Zhuang X, Rabczuk T. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: concepts, implementation and applications. Comput Methods Appl Mech Eng. 2020;362: 112790. https://doi.org/10.1016/j.cma.2019.112790.
20. Lu L, Pestourie R, Yao W, Wang Z, Verdugo F, Johnson SG. Physics-informed neural networks with hard constraints for inverse design. SIAM J Sci Comput. 2021;43(6):B1105–32. https://doi.org/10.1137/21M1397908.
21. Lagaris IE, Likas A, Fotiadis DI. Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Netw. 1998;9(5):987–1000. https://doi.org/10.1109/72.712178.
22. Haghighat E, Raissi M, Moure A, Gomez H, Juanes R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. Comput Methods Appl Mech Eng. 2021;379: 113741. https://doi.org/10.1016/j.cma.2021.113741.

23. Reissner E. On a variational theorem in elasticity. J Math Phys. 1950;29(1–4):90–5. https://doi.org/10.1002/sapm195029190.
24. Popp A, Wriggers P, editors. Contact modeling for solids and particles, vol. 585. CISM International Centre for Mechanical Sciences. Berlin: Springer International Publishing; 2018. https://doi.org/10.1007/978-3-319-90155-8.
25. Seitz A, Popp A, Wall WA. A semi-smooth Newton method for orthotropic plasticity and frictional contact at finite strains. Comput Methods Appl Mech Eng. 2015;285:228–54. https://doi.org/10.1016/j.cma.2014.11.003.
26. Deng Q, Li C, Tang H. A smooth system of equations approach to complementarity problems for frictionless contacts. Math Probl Eng. 2015;2015: 623293. https://doi.org/10.1155/2015/623293.
27. Li YM, Wang XT. Properties of a class of NCP-functions and a related semismooth newton method for complementarity problems. Int J Innov Comput Inf Control. 2012;8(2):1237–49.
28. Berrone S, Canuto C, Pintore M. Variational physics informed neural networks: the role of quadratures and test functions. J Sci Comput. 2022;92(3):100. https://doi.org/10.1007/s10915-022-01950-4.
29. Kharazmi E, Zhang Z, Karniadakis GE. hp-VPINNs: variational physics-informed neural networks with domain decomposition. Comput Methods Appl Mech Eng. 2021;374: 113547. https://doi.org/10.1016/j.cma.2020.113547.
30. Pantidis P, Mobasher ME. Integrated finite element neural network (I-FENN) for non-local continuum damage mechanics. Comput Methods Appl Mech Eng. 2023;404: 115766. https://doi.org/10.1016/j.cma.2022.115766.
31. Santapuri S, Lowe RL, Bechtel SE. Chapter 9. Modeling of thermo-electro-magneto-mechanical behavior, with application to smart materials. In: Bechtel SE, Lowe RL, editors. Fundamentals of continuum mechanics. London: Academic Press; 2015. p. 249–303. https://doi.org/10.1016/B978-0-12-394600-3.00009-5.
32. Popp A, Gitterle M, Gee MW, Wall WA. A dual mortar approach for 3D finite deformation contact with consistent linearization. Int J Numer Methods Eng. 2010;83(11):1428–65. https://doi.org/10.1002/nme.2866.
33. Popp A, Wall WA. Dual mortar methods for computational contact mechanics—overview and recent developments. GAMM Mitteilungen: Gesellschaft für Angewandte Mathematik und Mechanik. 2014;37(1):66–84. https://doi.org/10.1002/gamm.201410004.
34. Wriggers P, Laursen TA. Computational contact mechanics, vol. 2. Wien: Springer; 2006. https://doi.org/10.1007/978-3-540-32609-0.
35. Tonti E. The reason for analogies between physical theories. Appl Math Model. 1976;1(1):37–50. https://doi.org/10.1016/0307-904X(76)90023-8.
36. Lawrence J. Introduction to neural networks. Nevada City: California Scientific Software; 1993.
37. Kollmannsberger S, D'Angella D, Jokeit M, Herrmann L. Deep learning in computational mechanics: an introductory course, vol. 977. Cham: Springer International Publishing; 2021. p. 1–3. https://doi.org/10.1007/978-3-030-76587-3_1.
38. Saidaoui H, Espath L, Tempone R. Deep nurbs–admissible neural networks. arXiv preprint. 2022. arXiv:2210.13900.
39. Baydin AG, Pearlmutter BA, Radul AA, Siskind JM. Automatic differentiation in machine learning: a survey. J Mach Learn Res. 2018;18:1–43.
40. Sun L, Gao H, Pan S, Wang JX. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. Comput Methods Appl Mech Eng. 2020;361: 112732. https://doi.org/10.1016/j.cma.2019.112732.
41. Yastrebov VA. Numerical methods in contact mechanics. London: Wiley; 2013. https://doi.org/10.1002/9781118647974.
42. Li M, Dankowicz H. Optimization with equality and inequality constraints using parameter continuation. Appl Math Comput. 2020;375: 125058. https://doi.org/10.1016/j.amc.2020.125058.
43. Fischer A. A special Newton-type optimization method. Optimization. 1992;24(3–4):269–84. https://doi.org/10.1080/02331939208843795.
44. Sun D, Qi L. On NCP-functions. Comput Optim Appl. 1999;13:201–20. https://doi.org/10.1023/A:1008669226453.
45. Bartel T, Schulte R, Menzel A, Kiefer B, Svendsen B. Investigations on enhanced Fischer–Burmeister NCP functions: application to a rate-dependent model for ferroelectrics. Arch Appl Mech. 2019;89:995–1010. https://doi.org/10.1007/s00419-018-1466-7.
46. Xu C, Cao BT, Yuan Y, Meschke G. Transfer learning based physics-informed neural networks for solving inverse problems in engineering structures under different loading scenarios. Comput Methods Appl Mech Eng. 2023;405: 115852. https://doi.org/10.1016/j.cma.2022.115852.
47. Géron A. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. Sebastapol: O'Reilly Media, Inc.; 2022.
48. Hu Z, Zhang J, Ge Y. Handling vanishing gradient problem using artificial derivative. IEEE Access. 2021;9:22371–7. https://doi.org/10.1109/ACCESS.2021.3054915.
49. Kingma DP, Ba J. Adam: a method for stochastic optimization. Computing Research Repository. 2014. arXiv:1412.6980. https://api.semanticscholar.org/CorpusID:6628106.
50. Zhu C, Byrd RH, Lu P, Nocedal J. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Trans Math Softw. 1997;23(4):550–60. https://doi.org/10.1145/279232.279236.
51. Byrd RH, Lu P, Nocedal J, Zhu C. A limited memory algorithm for bound constrained optimization. SIAM J Sci Comput. 1995;16(5):1190–208. https://doi.org/10.1137/0916069.
52. Markidis S. The old and the new: can physics-informed deep-learning replace traditional linear solvers? Front Big Data. 2021;4: 669097. https://doi.org/10.3389/fdata.2021.669097.
53. Lu L, Meng X, Mao Z, Karniadakis GE. DeepXDE: a deep learning library for solving differential equations. SIAM Rev. 2021;63(1):208–28. https://doi.org/10.1137/19M1274067.
54. Geuzaine C, Remacle JF. Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities. Int J Numer Meth Eng. 2009;79(11):1309–31. https://doi.org/10.1002/nme.2579.
55. Timoshenko S, Goodier JN. Theory of elasticity. New York: McGraw-Hill; 1951.
56. Atluri ZDHSN, Liu HT. Meshless local Petrov–Galerkin (MLPG) mixed collocation method for elasticity problems. Comput Model Eng Sci. 2006;14(3):141–52. https://doi.org/10.3970/cmes.2006.014.141.
57. Kikuchi N, Oden JT. Contact problems in elasticity. Philadelphia: Society for Industrial and Applied Mathematics; 1988. https://doi.org/10.1137/1.9781611970845.
58. Popp A, Gee MW, Wall WA. A finite deformation mortar contact formulation using a primal-dual active set strategy. Int J Numer Meth Eng. 2009;79(11):1354–91. https://doi.org/10.1002/nme.2614.

59. BACI: a comprehensive multi-physics simulation framework. https://baci.pages.gitlab.lrz.de/website.
60. Seitz A, Farah P, Kremheller J, Wohlmuth BI, Wall WA, Popp A. Isogeometric dual mortar methods for computational contact mechanics. Comput Methods Appl Mech Eng. 2016;301:259–80. https://doi.org/10.1016/j.cma.2015.12.018.
61. Popp A, Wohlmuth BI, Gee MW, Wall WA. Dual quadratic mortar finite element methods for 3D finite deformation contact. SIAM J Sci Comput. 2012;34(4):B421–46. https://doi.org/10.1137/110848190.

Publisher's Note