Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems

Minglang Yin[a,b], **Enrui Zhang**[c], **Yue Yu**[d], **George Em Karniadakis**[b,c,*,1]

[a]Center for Biomedical Engineering, Brown University, Providence, RI, United States of America

[b]School of Engineering, Brown University, Providence, RI, United States of America

[c]Division of Applied Mathematics, Brown University, Providence, RI, United States of America

[d]Department of Mathematics, Lehigh University, Bethlehem, PA, United States of America

Abstract

Multiscale modeling is an effective approach for investigating multiphysics systems with largely disparate size features, where models with different resolutions or heterogeneous descriptions are coupled together for predicting the system's response. The solver with lower fidelity (coarse) is responsible for simulating domains with homogeneous features, whereas the expensive high-fidelity (fine) model describes microscopic features with refined discretization, often making the overall cost prohibitively high, especially for time-dependent problems. In this work, we explore the idea of multiscale modeling with machine learning and employ DeepONet, a neural operator, as an efficient surrogate of the expensive solver. DeepONet is trained offline using data acquired from the fine solver for learning the underlying and possibly unknown fine-scale dynamics. It is then coupled with standard PDE solvers for predicting the multiscale systems with new boundary/initial conditions in the coupling stage. The proposed framework significantly reduces the computational cost of multiscale simulations since the DeepONet inference cost is negligible, facilitating readily the incorporation of a plurality of interface conditions and coupling schemes. We present various benchmarks to assess the accuracy and efficiency, including static and time-dependent problems. We also demonstrate the feasibility of coupling of a continuum model (finite element methods, FEM) with a neural operator, serving as a surrogate of a particle system (Smoothed Particle Hydrodynamics, SPH), for predicting mechanical responses of anisotropic and hyperelastic materials. What makes this approach unique is that a well-trained over-parametrized DeepONet can generalize well and make predictions at a negligible cost.

Keywords

Machine learning; Neural operator; DeepONet; Concurrent multiscale coupling; Finite element model; Domain decomposition

---

1. Introduction

Predicting and monitoring complex systems, where small-scale dynamics and interactions affect global behavior are ubiquitous in science and engineering [1–5]. In disciplines ranging from material fracture [6] to design problems [7], models at microscale have shown their capability in representing detailed material response. However, despite their improved accuracy, the usability of microscopic models is often compromised by several computational challenges. Specifically, microscopic models require a small spatio-temporal scale to fully resolve small-scale details and capture underlying stochastic dynamics, leading to a prohibitive computational expense. Therefore, simulating material dynamics at meso- or macroscale using microscopic models is still largely beyond reach. In addition, although bottom-up approaches such as fine-grained atomistic models have provided important insights into processes at microscale, they generally do not scale up to finite-size samples [8–10]. These challenges raise the need for efficient mathematical models and algorithms, which are capable of capturing small-scale behaviors while being computationally expedient.

In the last two decades, a variety of multiscale approaches have been proposed to address these challenges. Microscopic effects often concentrate locally, whereas a continuum model can accurately describe the system in the rest of the domain. Solving such a continuum model by well-established numerical methods would reduce the computational cost. Following this domain-decomposition strategy, several works that combine continuum and microscopic models have been proposed [11–19] to model multiscale systems. Another approach focuses on developing a fast surrogate as the fine-scale model represented by homogenization [20–25]. For example, [26] considered an approximation model of a partial differential equation (PDE) that contains small-scale oscillations in its coefficients, in essence, replacing these coefficients in the model with effective properties so that the resulting solutions can adequately approximate the solutions of the original problem. However, quantifying effective properties poses a challenging task in light of the feasibility of acquiring parameter values and the level of accuracy of the homogenized PDE model compared to the original microscopic model.

Recently, deep learning algorithms have been proposed for simulating physical problems [27–31]. In particular, neural networks have been employed in conjunction with standard numerical models to address the aforementioned challenges in multiscale modeling [2,32–37]. In [33], Arbabi et al. proposed a data-driven method that trains deep neural networks to learn coarse-scale partial differential operators based on fine-scale data. In [38], Bhatia et al. presented a novel paradigm of multiscale modeling that couples models at different scales using a dynamic-important sampling approach. A machine learning model is employed to dynamically sample in the phase space, hence enabling an automatic feedback from micro to macro scale. In [39], Masi and his collaborators developed a thermodynamics-based artificial neural network (TANN) and applied it in multiscale modeling of materials with microstructure. Their results demonstrated that TANN is capable of implicitly learning the constitutive model from data and predicting the corresponding stress fields based on state variables. The authors also demonstrated that TANN is capable of solving boundary value problem in a multiscale system with complex microstructure

using double-scale homogenization scheme. Other applications of machine learning in multiscale modeling include parameter inference [40–43], uncertainty quantification [44,45], data-driven modeling [24,36,44,46–48], etc.

In the last few years, a new family of machine learning model, deep neural operators, have been proposed to learn the solution operator of a PDE system implicitly [49–52]. Unlike another type of scientific machine learning, physics-informed neural networks (PINNs) [53], these neural operators can solve a PDE system given a new instance of interface conditions or model parameters without retraining [54–60]. Hence, such computational advantage enables neural operators to serve as an efficient surrogate model in multiscale coupling tasks, and especially for time-dependent multiscale problems, which even today have remained prohibitively expensive. Among the state-of-the-art neural operators, the Deep Operator Network (DeepONet) serves as a unique model with exceptional generalization capability and flexibility, which can learn the solution operator in irregular domains even in the presence of noise [61]. Another possibility is to use the Fourier neural operator (FNO), which is fast but is limited to complex geometries and structured data [49,50]. Herein, we choose DeepONet as the surrogate model. For a more thorough comparison between DeepONet and FNO, we refer the reader to [61,62].

The present work aims to address the aforementioned challenges by developing a new multiscale coupling framework, as demonstrated in Fig. 1. Specifically, the new framework couples a surrogate of the microscopic system (DeepONet) with a standard finite element method (FEM) that represents the macroscopic model. The coupling framework is flexible in choosing the domain decomposition algorithms, the types of boundary condition, or the nature of multiphysics problems (static or time-dependent). In addition, the surrogate model can learn from a plurality of fine-resolution microscopic systems or from experimental data (Fig. 1(b)). Our approach is the first attempt to couple neural operators with standard numerical solvers using a concurrent coupling method.

The paper is organized as follows: In Section 2.1, we briefly introduce DeepONet and the adopted coupling algorithms. In Section 3, we report the coupling performance for a series of benchmarks, including a 2D Poisson equation, a 1D heat problem, and a uniaxial tension problem with elastoplastic materials and hyperelastic materials. We conclude by a brief discussion on the implications of the presented model in Section 4. In the appendices, we give an overview of the smoothed particle hydrodynamics (SPH) method and present additional results for FEM and SPH simulations. Finally, we provide further details on the network training and data generation.

2. Methodology

In this section, we introduce the general architecture of DeepONet (Section 2.1) and the domain decomposition methods for the coupling framework (Section 2.2).

2.1. Deep Operator Network (DeepONet)

We briefly summarize the general architecture of DeepONet employed in this work. Let $\Omega \subset \mathbb{R}^p$ be a bounded open set, which is the domain of our input and output functions;

DeepONet can learn a general continuous operator between two Banach spaces of functions taking values in $\mathbb{R}^{d_f}$ and $\mathbb{R}^{d_u}$, respectively. We denote the input and output function spaces as $\mathscr{A} = \mathscr{A}\left(\Omega; \mathbb{R}^{d_f}\right)$ and $\mathscr{U} = \mathscr{U}\left(\Omega; \mathbb{R}^{d_u}\right)$. The network aims at approximating a mapping $G: \mathscr{A} \to \mathscr{U}$ between an input function $g \in \mathscr{A}$ and its corresponding output function $G(g) \in \mathscr{U}$. Although DeepONet can learn mappings between vector-valued functions, for simplicity of exposition we focus on learning scalar-valued functions ($d_u = 1$) in the following. For any $\mathbf{y} \in \Omega \subset \mathbb{R}^p$, $G(g)(\mathbf{y}) \in \mathbb{R}$ is the evaluation of function $G(g)$ at $\mathbf{y}$. As shown in Fig. 2, the branch network takes a function g in its discrete representation $[g(\mathbf{x}_1), g(\mathbf{x}_2), ..., g(\mathbf{x}_m)]$ at locations $\{\mathbf{x}_j\}_{j=1}^m$ as input and yields an array of operator features, $\{b_i\}_{i=1}^n$, as its output. The trunk network takes $\mathbf{y}$ as input and yields another array of operator features, $\{t_i\}_{i=1}^n$. Finally, $G(g)(\mathbf{y})$ can be approximated by [51,63]:

$$G(g)(\mathbf{y}) \approx \sum_{i=1}^{n} b_i t_i. \tag{1}$$

In this work, we adopt fully-connected neural networks as the architecture of both sub-networks. We refer the readers to [51,64] for theoretical analysis and error estimations of DeepONet.

Regarding the loss function, we adopt the mean squared error (MSE) which measures the square of L_2 norm between model predictions and training data. Given M sample input functions $g^{(i)}$, $i = 1, ..., M$, and the ground-truth of their corresponding output functions, $G_{\text{data}}\left(g^{(i)}\right)(\,\cdot\,)$ on a set of N points $\{\mathbf{y}_j\}_{j=1}^N$, the loss is expressed as:

$$\mathscr{L} = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(G_{\text{model}}\left(g^{(i)}\right)(\mathbf{y}_j) - G_{\text{data}}\left(g^{(i)}\right)(\mathbf{y}_j)\right)^2 + \widehat{R}, \tag{2}$$

where G_{model} is the learned operator of DeepONet for approximation and $G_{\text{data}}\left(g^{(i)}\right)(\mathbf{y}_j)$ denotes the data measurements for the i th output function at j th evaluation point. $\widehat{R}$ is an additional loss term for the purpose of regularization (see Section 3.1 and Eq. (11)).

2.2. Coupling methods

Here we introduce the procedure of coupling DeepONet with a numerical model. Consider a system defined on a computational domain Ω; we decompose the domain into Ω_I and Ω_{II} according to the features in each domain. These two domains are described by a macroscopic model (model I) and a microscopic model (model II), respectively. In this paper, we choose model I as FEM. Nonetheless, the proposed procedure can be generalized to couple other mesh-based models [65–68] or particle (meshfree) models [69–72] as shown in Fig. 1(b). Model II represents a DeepONet, which serves as a surrogate for the microscopic, fine-scale model.

Notably, the domain decomposition can be either overlapping or non-overlapping. Fig. 3(a) shows a one-dimensional illustration of the domain decomposition method. For the

overlapping setting, $\Gamma_1(=\{x_1\}$ for this 1D case) represents the internal boundary of model I and $\Gamma_2(=\{x_2\})$ is the boundary of model II. The overlapping region is $\overline{\Omega_I} \cap \overline{\Omega_{II}} = [x_2, x_1]$. The two models communicate with each other by exchanging interface conditions on Γ_1 and Γ_2. For the non-overlapping setting, we have $\Gamma_1 = \Gamma_2 := \Gamma(x_1 = x_2$ for this 1D case) and the two model exchanges interface information on Γ. Fig. 3(b) and Algorithm 1 summarize the iterative procedure of the coupling framework with a Robin-type boundary condition [73]. For the n th iteration, we utilize quantities $(\cdot)^n$ to calculate $(\cdot)^{n+1}$. To initiate the coupling procedure ($n = 0$), we start with an initial guess of the interface solution value $u^0(x_1)$ and derivative-related information $T^0(x_1)$ on Γ_1. Then, we take a linear combination of these quantities and forms a Robin-type boundary condition on Γ_1, namely, $\tilde{h}^0(x_1) = R_1 u^n(x_1) + R_2 T^0(x_1)$, which is applied on Model I. The method proceeds by solving for $u_I^{n+1}(x)$, $x \in \Omega_I$, from Model I, and interpolating the computed solution or its derivatives at Γ_2. The interpolated information will then be transmitted to Model II as the boundary condition. Then, we solve for $u_{II}^{n+1}(x)$, $x \in \Omega_{II}$, from Model II with the transmitted interface condition on Γ_2 and interpolate its solution on Γ_1. If the stopping criterion

$$\left\| u_I^{n+1} - u_I^n \right\|_{L^2(\Omega_I)}^2 + \left\| u_{II}^{n+1} - u_{II}^n \right\|_{L^2(\Omega_{II})}^2 < \epsilon, \tag{3}$$

is satisfied, the coupling result is considered to be converged. The solution for the two domains is:

$$u_I(x) := u_I^{n+1}(x), \; x \in \Omega_I, \tag{4}$$

$$u_{II}(x) := u_{II}^{n+1}(x), \; x \in \Omega_{II}. \tag{5}$$

If the solution is not converged, we proceed to update the interface information on Γ_1 with a relaxation formulation: $\tilde{h}^{n+1}(x_1) = (1 - \theta)h_I^{n+1}(x_1) + \theta h_{II}^{n+1}(x_1)$ where $h_{I,II}^{n+1} = R_1 u_{I,II}^n(x_1) + R_2 T_{I,II}^0(x_1)$ are the Robin boundary condition from Model I and II. Here, the relaxation parameter $\theta \in [0, 1]$ can be either fixed or updated according to the Aitken's rule [73,74]. Then, we proceed to a new iteration by transmitting the updated boundary condition $\tilde{h}^{n+1}(x_1)$ to Model I and repeating the procedure stated above with $n \leftarrow n + 1$. This procedure is repeated until the stopping criterion is satisfied. These coupling algorithms have their origin to the classical Schwarz coupling methods [75–77] and an iterative patching algorithms [77].

3. Results

In this section, we present the simulation results of four benchmark problems: 2D Poisson equation, 1D heat equation, 2D elastoplasticity, and 2D hyperelasticity, to demonstrate the applicability of our method. The detailed settings of these four examples, including the choices of model I and model II, interface conditions, and whether domains overlap, are provided in Table 1. For interface conditions, we studied four types of conditions, namely, Dirichlet-Dirichlet (D-D), Robin-Dirichlet (R-D), Neumann-Dirichlet (N-D), and Robin-Neumann(R-N). The first letter represents the boundary condition for Model I and the

second letter for Model II. For each problem, our framework consists of two stages. In the first stage, we train a DeepONet offline to obtain a surrogate of model II. Then, we couple Model I with DeepONet in the online stage using our proposed coupling method. Training data of elastoplastic FEM in Section 3.3 and hyperelasticity in Section 3.4 are generated by Abaqus [78] and an SPH solver [79]. All the other usages of FEM (including FEM solver for model I and generation of training data for model II) are based on the FEniCS package [80] using second-order Lagrange polynomials. For all experiments, the networks are trained on a NVIDIA GeForce RTX 2060 GPU. The FEM simulations are performed on two 6-core Intel Core i7-8700 CPUs.

Algorithm 1 Coupling Method. $\Gamma_1 := \partial\Omega_I$, $\Gamma_2 := \partial\Omega_{II}$. If non-overlapping, $\Gamma_1 = \Gamma_2$, else $\Gamma_1 \neq \Gamma_2$.

Initialization: Set model I with $u(x_1) = 0$ and model II with $u(x_1) = 0$

Main Loop:

for $n = 0 : n_{max} - 1$ **do**

    Model I (FEM):

    • Receive the interface information $h^n(x_1)$ from Model II $(x_1 \in \Gamma_1)$.

    • Solve for u_I^{n+1} from Model I.

    • Calculate $u_I^{n+1}(x_2)$ or $\left.\dfrac{\partial u_I^{n+1}}{\partial x}\right|_{x_2}$ and pass it to Model II $(x_2 \in \Gamma_2)$.

    Model II (NN):

    • Receive the interface information $u_I^{n+1}(x_2)$ or $\left.\dfrac{\partial u_I^{n+1}}{\partial x}\right|_{x_2}$ from Model I $(x_2 \in \Gamma_2)$.

    • Solve for $u_{II}^{n+1}/T_{II}^{n+1}$ from Model II.

    • Calculate $h_{II}^{n+1}(x_1) = R_1 u_{II}^{n+1}(x_1) + R_2 T_{II}^{n+1}(x_1)$.

    • Calculate $\tilde{h}^{n+1}(x_1) = (1 - \theta)h_{II}^{n+1}(x_1) + \theta h_I^{n+1}(x_1)$ and pass it to Model I $(x_1 \in \Gamma_1)$.

    If converged, stop;

end for

3.1. Poisson equation

We first study the feasibility of the proposed framework for solving a static problem. Let us consider a Poisson equation described by the PDE system in $\Omega := \Omega_{FEM} \cup \Omega_{NN}$:

$$-\Delta u(\mathbf{x}) = f(\mathbf{x}), \text{ in } \Omega = [0, 1]^2, \tag{6}$$

$$u(\mathbf{x}) = u|_{\Gamma_0}(\mathbf{x}), \text{ on } \Gamma_0, \tag{7}$$

where we set $f(\mathbf{x}) = 6$ for all $\mathbf{x} \in \Omega$. As shown in Fig. 4(a), we decompose one domain into two overlapping subdomains, namely $\Omega_{FEM} = [0, 1]^2/[0.4, 0.6]^2$ (with an internal boundary Γ_1) and $\Omega_{NN} = [0.3, 0.7]^2$ (with a boundary Γ_2). In Ω_{FEM}, the system is governed by Eqs. (6) and (7) with a boundary condition

$$u(\mathbf{x}) = \tilde{u}|_{\Gamma_1}(\mathbf{x}), \text{ on } \Gamma_1. \tag{8}$$

In this example, we first verify the efficacy of the coupling framework with a D-D interface condition. For this setup, in Ω_{NN}, the system is governed by Eqs. (6) and (7) with input

$$u(\mathbf{x}) = u_{FEM}|_{\Gamma_2}, \text{ on } \Gamma_2. \tag{9}$$

After presenting the results of the D-D case, we display results of parametric studies for better demonstrating the influence of other factors that may influence the convergence rate.

In the offline stage, we train a DeepONet as a surrogate of FEM for predicting solution in Ω_{NN}. First, we sample a set of boundary conditions $u|_{\Gamma_2}$ from a random field with $\alpha = 5$, a parameter in the correlation function in Eq. (D.1). Qualitatively, a smaller α results in a less smooth function. More details of the random field generation are provided in Appendix D. Then, we solve the Poisson equation with the randomly sampled boundary conditions using FEM, whose computational results in Ω_{NN} are collected as the training cases. We generate 1000 cases as the training dataset of DeepONet following the aforementioned process. As shown in Fig. 4(b), the branch network input $U_{FEM}|_{\Gamma_2}$ comes from an interpolation of the FEM solution on Γ_2, while the trunk network takes the coordinate $\mathbf{x}(\in \Omega_{NN})$ as its input. The network output is $u_{NN}(\mathbf{x})$. We present more details related to the network training in Appendix C.

After the completion of the offline training, we proceed to the coupling stage as shown in Fig. 4(c). Given a fixed boundary condition on Γ_0, we initiate the coupling framework with an initial guess of u on Γ_1, which is typically zero. Then, we solve the governing equation in Ω_{FEM} with FEM and interpolate the solution on Γ_2 (denoted as $u_{FEM}|_{\Gamma_2}$, which will be used as the input of the branch network. With this input, the DeepONet then predicts $u_{NN}|_{\Omega_{NN}}$, the solution of the Poisson equation in Ω_{NN}. Following that, a relaxation scheme is adopted to update the interface condition on Γ_1 as: $\tilde{u}|_{\Gamma_1} = (1 - \theta)u_{FEM}|_{\Gamma_1} + \theta u_{NN}|_{\Gamma_1}$, which will later be used as the boundary condition of FEM in the next iteration. The coupling iteration continues until the solution u converges.

We show the performance of our coupling framework for the Poisson equation in Fig. 5. We fix the relaxation parameter as $\theta = 0.5$ and impose a boundary condition on Γ_0 from the testing dataset (unseen to the DeepONet in the training stage). Figs. 5(a–b) show a comparison between the FEM ground truth (first column; computed directly in Ω) and the predictions from the coupling framework (second column; first row in Ω_{FEM} from FEM, second row in Ω_{NN} from DeepONet) together with their difference (third column). We observe an agreement between the coupling predictions and the ground truth. The maximum absolute error presented in the third column is less than 1%. We also note that the error is larger in Ω_{FEM}, especially in the region close to its external boundary Γ_0. The observed error is mostly dominated by the interpolation accuracy in the FEM, not the error of the coupling framework. We further show a quantitative comparison between models prediction and ground-truth on Γ_1 and Γ_2 in Fig. 5(c): the predictions from coupled FEM/NN (blue

dashed lines and green triangles, respectively) accurately reproduce the true solution (red lines) with a relative error at around 0.07%.

In addition to the results shown in Fig. 5, we also conducted parametric studies on diverse factors that influence the performance of our framework, including: convergence of the framework with different coupling methods (Fig. 6(a)); the coupling accuracy influenced by the extrapolation capability of the DeepONet (Fig. 6(b)) and the number of training cases (Fig. 6(c)). In Fig. 6(a), convergence of errors with different boundary conditions is plotted against iterations. The shaded area indicates that the relative error is less than 1%. Specifically, given various values of the relaxation parameter θ ($\theta = 0.25$, 0.5 and 0.75), the displacement errors with Dirichlet boundary conditions satisfy the stopping criterion (L_2 error less than 2×10^{-3}, or equivalently, relative error less than 1%) at iteration 37, 18, and 12, respectively. With the Aitken's relaxation strategy for θ (see, e.g., [73]), the coupling error (denoted as "Dirichlet-Aitken") converges a bit faster than that of a fixed relaxation parameter $\theta = 0.75$. We also adopt a Robin-type boundary (Robin–Aitken) with dynamic update in θ (purple line). In this case of Robin boundary condition, the system in domain Ω_{FEM} is governed by Eqs. (6) and (7) with a Robin boundary:

$$R_1 u + R_2 \frac{\partial u}{\partial n} = g, \text{ on } \Gamma_1, \tag{10}$$

where we set $R_1 = 1$ and $R_2 = 1$. Since the FEM needs both the information of the solution and its derivative in the normal direction from DeepONet to update the Robin boundary condition, we adjust the training loss of the network following the strategy. The regularization term $\widehat{R}$ in Eq. (2) is set as:

$$\widehat{R} = \left(\left. \frac{\partial u}{\partial n} \right|_{NN} - \left. \frac{\partial u}{\partial n} \right|_{true} \right)^2, \mathbf{x} \in \Gamma_1 \tag{11}$$

to regularize the partial derivative with respect to the normal direction of Γ_1. In Eq. (2), the partial derivative term is computed by taking the automatic differentiation of the network output with respect to the trunk net input $\mathbf{x}$ [59,81–84]. We assign the weight of the regularization term as 1. With the employment of the Robin boundary condition, we observe that the method only takes two iterations to reach a relatively small error.

In Fig. 6(b), we show the generalization ability of DeepONet and its impact on the accuracy of the coupling framework by testing with boundary conditions outside the training region. The minimal relative errors of $u|_{\Gamma_2}$ are plotted against the correlation length α of the random field that was used to generate training data. When the correlation length α increases, the sampled curves become smoother and vice versa. Notice that we train the network on training samples with $\alpha = 5$ and test its performance on α ranging from 1 to 10, each with 50 testing cases. The relative errors drop from around 10% to less than 1% with α increasing from 1 to 5. For $\alpha > 5$, the errors are statistically stable.

Fig. 6(c) exhibits the accuracy of the coupling framework as a function of the generalization ability of DeepONet, which is reflected by the number of training cases for DeepONet. The

median of the relative errors of $u|_{\Gamma_1}$ decreases slightly when the number of training cases increases from 100 to 300 and then stays statistically stable even with further increase. The computational results implicitly suggest that after 300 training cases, the errors are most contributed by the accuracy of boundary interpolation, not the generalization of DeepONet. The shaded area in (b–c) denotes relative error at 1%.

3.2. Heat equation

Next, we investigate the performance of the coupling framework for a dynamic problem, namely, 1D heat equation, with a N-D method. Consider a PDE system in $\Omega = \Omega_{FEM} \cup \Omega_{NN}$:

$$\frac{\partial u}{\partial t} = K\frac{\partial^2 u}{\partial x^2}, \text{ for } (x,t) \in \Omega := [0, 1.0] \times [0, 10.0], \tag{12}$$

$$u(x, 0) = u_0(x), \text{ for } x \in [0, 1.0], \tag{13}$$

$$u_x(x, t) = 0, \text{ for } (x, t) \in \Gamma_0 := \{0, 1.0\} \times [0, 10.0], \tag{14}$$

where the thermal diffusivity K is set as 0.1. As shown in Fig. 7(a), we decompose the computational domain Ω into two non-overlapping subdomains: $\Omega_{FEM} := [0, 0.5] \times [0, 10.0]$ for FEM and $\Omega_{NN} := [0.5, 1] \times [0, 10.0]$ for DeepONet. The interface is $\Gamma := \{0.5\} \times [0, 10]$. For the FEM subdomain Ω_{FEM}, we set a Dirichlet boundary condition at the interface Γ:

$$u(x, t) = \tilde{u}(x, t), \text{ for } (x, t) \in \Gamma, \tag{15}$$

where $\tilde{u}$ is the updated and relaxed solution at the interface. For the DeepONet subdomain Ω_{NN}, we impose the interfacial flux from FEM $\frac{\partial u_{FEM}}{\partial x}$ as input.

The coupling method is illustrated in Fig. 7(b–c). The solution process is a nested loop with indices n and k, where k represents the time step and n is the current iteration step. To initiate the framework, we set both n and k as zero with an initial guess of interfacial flux for the FEM. The FEM solves for the solution $(u_{FEM}^{n+1,k+1}|_{\Omega_{FEM}})$ in Ω_{FEM} at time step $k+1$ given the solution at the previous time step k, denoted as $u_{FEM}^{\infty,k}|_{\Omega_{FEM}}$. Then, we transmit the flux at Γ, $\frac{\partial u_{FEM}^{n+1,k+1}}{\partial x}\bigg|_{\Gamma}$, to DeepONet as a part of the trunk net input (Fig. 7 (c)). The branch network takes the system solution in Ω_{NN} at k th time step as input. The output of DeepONet estimates an approximation of the diffusion term $\frac{\partial^2 u_{NN}^{n+1,k+1}}{\partial x^2}$. The solution in Ω_{NN} is then calculated by a semi-discretized heat equation with the backward Euler method: $u_{NN}^{n+1,k+1}(x) = u_{NN}^{n,k}(x) + \Delta t \cdot K\frac{\partial^2 u_{NN}^{n+1,k+1}}{\partial x^2}$. Next, the interfacial flux $\tilde{u}^{n+1,k+1}$ is updated by the relaxation scheme:

$$\tilde{u}^{n+1,k+1} = (1-\theta)u_{FEM}^{n+1,k+1}|_{\Gamma_1} + \theta u_{NN}^{n+1,k+1}|_{\Gamma_1}. \tag{16}$$

In this example, we fix the relaxation parameter $\theta = 0.5$. If the updated interfacial flux is not yet converged, we continue to the next iteration. Otherwise, if the flux is converged, then we proceed to the next time step in the outer loop and restart the inner loop with the reset iteration step $n = 0$.

Herein, we summarize the training procedure of DeepONet. We generate 1000 initial conditions in $x \in [0.5, 1.0]$ from a Gaussian random field with constant mean 0 and correlation length 0.3. For each case, we randomly sample the flux at $u|_{\Gamma_1}$ 20 times based on a uniform distribution from -3 to 3 as the boundary conditions. The training data of DeepONet is then generated by running FEM simulations on Ω_{NN} with the sampled initial/boundary conditions. Note that training a DeepONet in a spatio-temporal domain with disparate boundary/initial condition is a data-demanding task: one needs to sample in the spatio-temporal domain. Also, performance of the framework may be deteriorated when the network extrapolates the solution outside the training domain. Hence, we alleviate the challenges by training the network to learn the implicit spatial derivative operator $\frac{\partial^2 u}{\partial x^2}$ with an explicit method to advance in time. In practice, we acquire the spatial derivative $\frac{\partial^2 u}{\partial x^2}$ in Ω_{NN} from FEM simulations running from $t = 0$ to 1.0 with $\Delta t = 0.1$. The simulated spatial derivative is then collected and utilized for DeepONet training. For more details of the network training, we refer the reader to Appendix C.

The coupling results are shown in Fig. 8, where (a) shows the prediction from both models in the spatio-temporal domain Ω. In Fig. 8(b), the solution of DeepONet at $x = 0.75$ is plotted against time. The prediction from DeepONet shows a good agreement with the ground truth solution even for the temporal region outside the training dataset ($t > 1$), indicating that the proposed framework works well for extrapolation. The mean-squared errors (MSE) of both models in the coupling framework are plotted against time in Fig. 8(c): the coupling framework shows stability and accuracy over long-time integration. Although the prediction errors grow with the increase of t, we note that the errors accumulate at a relatively low rate, demonstrating that the proposed coupling framework is capable of solving a time-dependent system. It may be misleading that, from $t \in [2, 7]$, DeepONet outperforms FEM since the mean squared error of DeepONet is lower than that of FEM. In our concurrent coupling framework, the errors in FEM is induced by the oscillatory errors from DeepONet. Therefore, one can observe that the errors from FEM and DeepONet are both in a similar scale. In general, one might observe a slightly better accuracy from DeepONet, but these observations are generally problem-dependent and do not indicate that the data-driven DeepONet has a better generalization ability than FEM.

3.3. Elastoplasticity

In this section, we test the proposed framework for predicting the elastoplastic behavior of a solid material. As shown in Fig. 9(a), we consider a plane strain problem for a square-shaped solid of size $2l \times 2l$ with a circular void of radius r_i, where vertical tension is applied on its top edge. In this example, we take $l = 1$, $r_i = 0.1$. For linear elastic materials, the kinematics, constitutive relation, and equilibrium equations are as follows:

$$\varepsilon = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathrm{T}}\right), \tag{17}$$

$$\boldsymbol{\sigma} = \lambda \operatorname{tr}(\varepsilon)\mathbf{I} + 2\mu\varepsilon, \tag{18}$$

$$\mathbf{0} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{b}, \tag{19}$$

where ε, $\mathbf{u}$, and σ are the strain, displacement, and (Cauchy) stress; λ and μ are Lamé moduli, which we take as $\lambda = 0.5769$ and $\mu = 0.3846$, and $\mathbf{I}$ is the identity tensor. In this example, we consider a solid subject to no body load, and therefore set the body force term $\mathbf{b}$ as zero.

To model the plastic behavior of the material, we consider small-deformation, rate-independent elastoplasticity with isotropic hardening. The additive decomposition of the strain tensor writes $\varepsilon = \varepsilon^{\mathrm{e}} + \varepsilon^{\mathrm{p}}$, where ε^{e} and ε^{p} are the elastic and plastic strains, respectively. The elastic strain ε^{e} is related to the stress by

$$\boldsymbol{\sigma} = \lambda \operatorname{tr}(\varepsilon^{\mathrm{e}})\mathbf{I} + 2\mu\varepsilon^{\mathrm{e}}. \tag{20}$$

The plastic strain ε^{p} is purely deviatoric (i.e., $\operatorname{tr}(\varepsilon^{\mathrm{p}}) = 0$). We define the deviatoric stress $\mathbf{s}$, the increment of the equivalent plastic strain $\mathrm{d}\bar{\varepsilon}^{\mathrm{p}}$, and the equivalent tensile stress (Mises stress) $\bar{\sigma}$ as

$$\mathbf{s} = \boldsymbol{\sigma} - \frac{1}{3}\operatorname{tr}(\sigma)\boldsymbol{I}, \tag{21}$$

$$\mathrm{d}\bar{\varepsilon}^{\mathrm{p}} = \sqrt{\frac{2}{3}\,\mathrm{d}\varepsilon^{\mathrm{p}} : \mathrm{d}\varepsilon^{\mathrm{p}}}, \tag{22}$$

$$\bar{\sigma} = \sqrt{\frac{3}{2}\mathbf{s} : \mathbf{s}}, \tag{23}$$

respectively. The flow direction N^{p} and the increment of the plastic strain $\mathbf{d}\varepsilon^{\mathrm{p}}$ are given by

$$\boldsymbol{N}^{\mathrm{p}} = \sqrt{\frac{3}{2}}\frac{\mathbf{s}}{\bar{\sigma}}, \tag{24}$$

$$\mathrm{d}\varepsilon^{\mathrm{p}} = \sqrt{\frac{3}{2}}\,\mathrm{d}\bar{\varepsilon}^{\mathrm{p}}\boldsymbol{N}^{\mathrm{p}}. \tag{25}$$

Then, the yield function f can be defined as

$$f = \bar{\sigma} - Y\!\left(\bar{\varepsilon}^{\mathrm{p}}\right), \tag{26}$$

where the linear strain-hardening function Y is taken as

$$Y(\bar{\varepsilon}^{\mathrm{p}}) = Y_0 + H_0 \bar{\varepsilon}^{\mathrm{p}}. \tag{27}$$

The initial strength $Y_0 = 0.1$ and the hardening modulus $H_0 = 0.3$ are taken as two constant material parameters. The aforementioned mechanical quantities are subject to the Kuhn-Tucker complementary conditions:

$$f \leq 0, \quad \mathrm{d}\bar{\varepsilon}^{\mathrm{p}} \geq 0, \quad (\mathrm{d}\bar{\varepsilon}^{\mathrm{p}})f = 0. \tag{28}$$

In addition, when $f = 0$, the consistency condition $\mathrm{d}\bar{\varepsilon}^{\mathrm{p}}\mathrm{d}f = 0$ also needs to be satisfied.

Due to the setup of our boundary value problem, the plastic deformation concentrates around the void whereas the material is dominated by elastic behaviors in regions away from the void. Hence, we decompose the square domain into two non-overlapping subdomains (see Fig. 9(a–b)): the internal region Ω_{NN}, which is an annulus with internal radius $r_i = 0.1$ and external radius $r_o = 0.3$ (on Γ), modeled by DeepONet as a surrogate for the solid's elastoplastic response; the external region Ω_{FEM}, modeled by FEM for linear elasticity. The two regions share a common interface on Γ. In Ω_{NN}, we train several DeepONets as surrogates of each stress components to capture the plastic behavior. First, we sample 1000 displacement boundary conditions on the top edge using the sampling method described in Appendix D.2, and employ the sampled data as boundary conditions. Then, we solve for the displacement and stress fields in the entire domain with a FEM solver based on the elastoplasticity model described above. Next, we collect the simulation results in Ω_{NN}, which will be employed as the training data of DeepONets. As depicted in Fig. 9(c), DeepONet can predict the corresponding Cauchy stresses in Ω_{NN} with input as the displacement at the interface. In 2D problems, the Cauchy stress $\sigma(\mathbf{x})$ for each material point is a 2 × 2 symmetric matrix. Therefore, to model the stress we only need to predict its three components, namely, σ_{11}, σ_{12}, and σ_{22}. For each of these components, we train an independent DeepONet separately as a surrogate of these quantities. The traction at the interface $\mathbf{t}|_{\Gamma}$ is calculated accordingly based on the predicted stress from the DeepONets. More details regarding the training of this network are presented in Appendix C.

Then, we employ the trained DeepONets in the coupling framework. As depicted in Fig. 9, DeepONets and FEM communicate at the interface Γ by transmitting the information of displacement and traction (Fig. 9). The interfacial displacement, $\mathbf{u}_{FEM}|_{\Gamma}$, is computed in FEM and transmitted to the DeepONets as the input of the branch network. Then, the network solves for the corresponding stress in Ω_{NN} and calculates the traction at the interface ($\mathbf{t}_{NN}|_{\Gamma}$). In the next iteration, the computed traction $\mathbf{t}_{NN}|_{\Gamma}$ will be imposed as the boundary condition of the FEM model. In this example, we employ a relaxation scheme for the traction from the DeepONets. The relaxation parameter θ is fixed at 0.5.

We present the coupling results with a N-D interface condition in Fig. 10. In Figs. 10(a–b), we plot the ground truth solution from FEM in the first column, the predictions from our coupling framework in the second column, together with their differences in the third column. In plot (a), we show the results of the normal stress (σ_{22}) in the vertical direction

of Ω_{FEM}. The results of the equivalent plastic strain, $\bar{\varepsilon}^p$ in Ω_{NN} are provided in plot (b). Although the coupling framework has generally well reproduced the stress component σ_{22} in the bulk region of Ω_{FEM}, there exist relatively large errors at the top edge and the bottom corners. These errors either originate from numerical interpolation in the FEM solver or are caused by a reduced solution regularity. Apart from these regions, the errors are controlled at a low value with the maximum relative error lower than 10%. In Ω_{NN}, the profile of $\bar{\varepsilon}^p$ is well captured by the coupling scheme with the maximum relative error of $\bar{\varepsilon}^p$ less than 10%. The results demonstrate that plasticity in the region of interest is well predicted by the surrogate model. To provide a further quantitative verification of our coupling framework, we plot the ground-truth, the predictions of displacement, and traction components on Γ in Fig. 10(c) as functions of θ, the angle in polar coordinate. The solid lines denote the ground-truth results from the nonlinear FEM. The dashed lines denote the predicted values of the framework. The model predictions match the ground truth well, with relative errors smaller than 2%. Fig. 10(d) shows the efficiency of the coupling framework: the L_2 errors between model predictions and the ground truth reach a plateau at the fourth iteration.

3.4. Hyperelasticity

In the previous examples, DeepONet was trained based on data generated from FEM and was coupled with another FEM in the online stage. These examples illustrated the capability of our coupling framework in solving both static and dynamic problems. In this section, we demonstrate the capability of our framework in concurrently coupling a continuum model (FEM) with a surrogate from smooth particle dynamics (SPH) for describing a microscopic particle system. We consider using the coupled framework to predict the mechanics of a hyperelastic material. We first derive the energy minimization formulation of the continuum model. We denote by ψ the strain energy density of the hyperelastic model and seek to find a displacement field $\mathbf{u}: \Omega \to \mathbb{R}^2$ that minimizes the total potential energy Ψ:

$$\Psi = \int_{\Omega} \psi(\mathbf{u}) d\mathbf{x} - \int_{\Omega} \mathbf{b} \cdot \mathbf{u} d\mathbf{x} - \int_{\Gamma_N} \mathbf{T}_0 \cdot \mathbf{u} ds. \tag{29}$$

Here, $\mathbf{b}$ denotes the body force in Ω, $\mathbf{T}$ is the traction load applied on the Neumann boundary Γ_N. Hence, the total potential energy Ψ is the integration of strain energy density, ψ, over the entire domain Ω, deduced by the energy contributions from the body force $\mathbf{b}$ and the traction $\mathbf{T}$.

We consider the Holzapfel–Gasser–Odgen (HGO) model [85] to describe the constitutive behavior of the material in this example. Essentially, the material is hyperelastic, anisotropic, fiber-reinforced in diverse directions. Its strain energy density is:

$$\psi = \frac{\mu}{2}(I_1 - 3) - \mu \ln(J) + \frac{k_1}{2k_2} \sum_{i=1}^{2} \left(\exp\left(k_2 \langle E_i \rangle^2 \right) - 1 \right) + \frac{K}{2} \left(\frac{J^2 - 1}{2} - \ln J \right), \tag{30}$$

where $\langle \cdot \rangle$ denotes the Macaulay bracket. In this model, the fiber strain of the two fiber groups is expressed as:

$$E_i = \kappa(I_1 - 3) + (1 - 3\kappa)(I_{4i} - 1), \ i = 1, 2, \tag{31}$$

where k_1 and k_2 are fiber modulus and the exponential coefficient, respectively, I_1 is the first principal invariant, and I_{4i} is the fourth principal invariants corresponding to the i th fiber group. Mathematically, for the i th fiber group with angle direction α_i from the reference direction, I_{4i} is calculated by $n_i^{\mathrm{T}}\mathbf{C}n_i$, where $\mathbf{C}$ is the right Cauchy–Green tensor and $n_i = [\cos\alpha_i, \sin\alpha_i]^{\mathrm{T}}$. In our simulations, we consider a material with fiber reinforcement in the vertical direction (see Fig. 11 right for illustration). Therefore, for both fiber groups we set $\alpha_i = \pi/2$. In Eq. (31), fiber dispersion is denoted as κ, whose value ranges from 0 to $\frac{1}{3}$. Intuitively, $\kappa = 0$ means no fiber dispersion whereas $\kappa = \frac{1}{3}$ represents an isotropic fiber dispersion. In this example, we consider the fiber oriented vertically with no dispersion ($\kappa = 0$). All parameter values in this example are summarized in Table 2.

We now present the problem set up of this example. As depicted in Fig. 11(a), we consider a 2D unit square plate with a centered circular void (radius as 0.1). The plate deforms under a uniaxial tension, $\mathbf{T}_0(\mathbf{x})$, applied on its top edge. The bottom edge is clamped. We model the material response of the entire domain using an SPH model, whose solution is taken as the ground-truth solution. More details of the SPH model are provided in Appendix A. To develop a coupling model, we consider a similar setting as in the previous example and decompose the entire domain Ω into two non-overlapping subdomains. Then, we train DeepONets using SPH data to obtain a surrogate for the internal domain while the external region is described by a continuum FEM model. Models in these two domains communicate by exchanging proper interface conditions on their common interface, Γ. In Fig. 11(b) we present a schematic of information transmission of the coupling framework with a N-D method. At the n th iteration, the FEM solver receives an updated distributed traction on Γ from DeepONets and solves the updated displacement field, $\mathbf{u}_{FEM}^n$, with the given information. Then, the FEM transmits the updated displacement information $\mathbf{u}_{FEM}^n \mid \Gamma$ to DeepONets. With the displacement on Γ as input, DeepONets estimate the first Piola-Kirchhoff (PK1) stresses in Ω_{NN}. Based on the predicted PK1 stresses, we then calculate the surface traction on Γ and other associated quantities, such as the equivalent plastic strain $\bar{\varepsilon}^{\mathrm{p}}$ and von Mises stress $\bar{\sigma}$, accordingly. At the n th iteration, the system solution at the interface Γ is updated as $\widetilde{\mathbf{T}}(\mathbf{x}) = (1 - \theta)\mathbf{T}_I^n(\mathbf{x}) + \theta\mathbf{T}_{II}^n(\mathbf{x})$.

Next, we briefly describe the training process of DeepONet. To generate the training/testing dataset, we sample 1000 different traction loading $\mathbf{T}_0(\mathbf{x})$ on the top edge from a random field (see Algorithm in Appendix D). Then, for each sampled traction loading, we perform an SPH simulation to obtain the solutions in the entire domain and collect the corresponding solutions of displacement and PK1 stress fields of in Ω_{NN}. Among these 1000 samples, 900 cases are employed as the training data while the rest is kept as testing data. As depicted in Fig. 11(c), we consider two approaches for network training. In the first approach, the network (Net_{u2s}) takes the displacement at the interface ($\mathbf{u}|_\Gamma$) as input and predicts PK1 stress as the output. In the second approach, the interfacial traction $T|_\Gamma$ is employed as the input of the network (Net_{u2s}), yielding the displacement field as output. These approaches provide

a flexibility of imposing different interface conditions. We adopt the first approach in the N-D method and combine the information of the two approaches ($\widehat{\mathbf{u}}$ and $\widehat{\mathbf{T}}$) in the R-D/R-N method.

Fig. 12 shows the coupling results of a typical testing case with a N-D method. In Fig. 12(a–b), we compare the vertical displacement u_2 in Ω_{FEM} (first row) and the PK1 stress component P_{22} in Ω_{NN} (second row) between the SPH ground truth (first column) and the FEM/NN prediction (second column). The prediction errors are displayed in the third column. We observe that the coupling results match well with the ground truth solution with the largest prediction errors distributed near the bottom corners. The errors are partially induced by the numerical interpolations in the FEM and reduced regularity in that region. To further examine the prediction accuracy, in Fig. 12(c) we plot a quantitative comparison of displacement and traction on the interface Γ as functions of the polar coordinate angle θ. Despite some numerical discrepancies between SPH and FEM (see Appendix B.1), the FEM/DeepONet predictions shown in dashed lines successfully reproduce the SPH simulation results depicted in solid lines. Therefore, our proposed method is capable of capturing the mechanics from SPH with a substantially improved efficiency: the time cost of performing an SPH simulation is approximately 4 h whereas running its surrogate just takes a fraction of a second (Table 3). Admittedly, the excessive cost of SPH is exacerbated because we use a time-dependent SPH solver to simulate a static problem. Nonetheless, we can see that replacing a particle model with its surrogate in a multiscale coupling framework poses unique advantages in both efficiency and programming easiness.

To further illustrate the flexibility and investigate the convergence rate of our coupling framework, we employ a R-D/R-N coupling method to our framework with a variety of values of the Robin coefficient R. To clarify, R-D/R-N means that the FEM, imposed with a Robin-type boundary condition, separately transmits the information of displacement and traction to Net_{u2s} and Net_{s2u} in order to update the Robin information in the next step. The continuum model with a Robin boundary condition is modified as:

$$\Psi = \int_{\Omega} \psi(\mathbf{u}) \mathrm{d}\mathbf{x} - \int_{\Omega} \mathbf{b} \cdot \mathbf{u} \mathrm{d}\mathbf{x} - \int_{\Gamma_N} \mathbf{T}_0 \cdot \mathbf{u} \mathrm{d}s - \int_{\Gamma_R}\left(\mathbf{r} - \frac{1}{2}R\mathbf{u}\right) \cdot \mathbf{u} \mathrm{d}s . \tag{32}$$

$\mathbf{r}$ is the Robin boundary condition applied on the interface Γ_R. We define $\mathbf{r}$ as:

$$\mathbf{r} = \widehat{\mathbf{T}} + R \cdot \widehat{\mathbf{u}}, \text{ for } \mathbf{x} \in \Gamma_R, \tag{33}$$

where R is the (positive) Robin coefficient, $\widehat{\mathbf{T}}$ and $\widehat{\mathbf{u}}$ are the known traction and displacement.

The coupling procedure is changed as well. At the n th iteration, we first solve the FEM model with a Robin boundary condition $\tilde{\mathbf{r}}^n(\mathbf{x})$. Then, we transmit the interfacial traction and displacement of FEM to Net_{u2s} and Net_{s2u}, respectively. The corresponding displacement $\widehat{\mathbf{u}}^{n+1}$ and tractions $\widehat{\mathbf{T}}^{n+1}$ are estimated by DeepONets and used to update the Robin boundary condition: $\tilde{\mathbf{r}}^{n+1}(\mathbf{x}) = (1 - \theta)\mathbf{r}_{FEM}^{n+1}(\mathbf{x}) + \theta\mathbf{r}_{NN}^{n+1}(\mathbf{x})$. Here, $\theta = 0.5$ is taken as a fixed relaxation parameter.

We further study the convergence of interfacial displacement errors with different values of R. In Fig. 13, we highlight the results from the N-D coupling method with the black line and the results of R-D/R-N with $R = 0.25$ (best result) in red. The shaded area indicates 2% equivalent relative error of the test case. Due to the nonlinearity of the problem, it generally takes more iterations for the coupling framework to reach the stopping criterion. When taking a R-N coupling method with a sub-optimal Robin coefficient (such as $R = 5.0$ as shown in the purple line of Fig. 13), the coupling framework fails to converge. This fact again demonstrates the importance of choosing an appropriate coupling method.

4. Discussion

In this paper, we propose an efficient concurrent coupling framework for multiscale modeling of mechanics problems. In lieu of coupling an expensive microscopic model, we propose to employ a deep neural operator, DeepONet, as a surrogate to approximate the microscopic solution in the domain with fine-scale features. The response in the coarse-scale domain is simulated by a standard numerical model, such as the finite element method. The two models are coupled concurrently by exchanging information at the interface until convergence. To verify the performance of this framework, we study four benchmarks including solving static and dynamic problems for different materials. We have also demonstrated that the framework is readily applicable for various interface conditions. The results show that the cases with a Robin boundary condition tend to converge faster than a Neumann/Dirichlet boundary condition. Moreover, the predictions of the coupled model agree well with the true solution acquired from a numerical method, indicating the generalization ability of DeepONet and the accuracy of the proposed framework.

In addition, using a neural operator as surrogate enables the model to be trained directly from data. Such a model is particularly promising for learning dynamics of complex materials without explicit constitutive models. Moreover, coupling DeepONet with FEM substantially improves the computational efficiency in multiscale modeling: DeepONet predicts the expensive microscopic behavior only at a fraction of second. Hence, the overall computational cost of the proposed framework could be shorten by orders of magnitude than the existing multiscale coupling methods. In addition, we can train the surrogate model to learn based on partial information: it can predict the information only at the interface and neglect dynamics inside the microscopic region. Thus, the overall framework behaves as an artificial-intelligence type boundary condition, which would potentially improve the efficiency especially in the scenarios where only the macroscopic dynamics is of interest.

As a further note, we would like to point out the keys to build a successful neural operator based coupling framework with guaranteed convergence and to avoid possible pitfalls. First, the generalization ability of DeepONet determines if the result converges and its convergence rate. Utilizing an accurate neural operator model takes fewer iterations to converge. However, a poorly trained model could lead to slow convergence or even divergence. Second, normalizing data is another key to convergence. In our experiments, the training data are at disparate scales, ranging from $10^{-3} - 10^{1}$. Properly normalization of the training data would be essential to the convergence of numerical iterations. In addition, choosing proper coupling strategies, such as the right Robin coefficient, also plays a critical

role in guaranteeing fast numerical convergence as it affects the condition number of the stiffness matrix in FEM and the coupling system [86]. A large condition number may lead to a slow convergence or even divergent result in the coupling framework (see Section 3.4 and [87,88]).

Certainly, more improvements and directions can be considered in the future. Learning directly from data facilitates a unique feature that DeepONet can learn data that comes from different scales, which has been demonstrated in Section 3.4. In the future, it would be interesting to develop a data-driven model from noisy data, such as molecular dynamics and dissipative particle dynamics, as presented in Fig. 1. In addition, overcoming the multiscale characteristic length and time scale would be another challenge in multiscale modeling with machine learning, that is, the time and length of a microscopic model are usually orders of magnitude smaller than the continuum model, causing challenges in network training and long-term predictions. The coupling results may drift away from the true solution or even diverge due to inaccurate predictions from surrogate models. Another improvement would be considering microstructures and geometric variations (see [39]). Developing an operator-learning neural network that can predict dynamics with different geometric variations would be a great improvement for broadening the application of the proposed method. Also, simulating fracture progression [89] is another natural and promising application of the multiscale coupling framework. As fracture progresses, the field of interest that includes the damage region may also move along with the tip of a crack. Such modification is particularly useful for problems with discontinuity and/or multiple phases, such as fracture, phase transition, shock wave capturing, and etc. Our framework can be further extended to employ other coupling methods such as quasicontinuum [90], multigrid [91], or serves as a surrogate model for heterogeneous multiscale method (HMM) [92], etc.

Acknowledgments

Appendix A.: Smoothed particle hydrodynamics

In this section, we briefly introduce the basic formulation of the Total Lagrangian Smoothed Particle Hydrodynamics (TLSPH). We refer the reader to [79,93] for more details. In the SPH framework, physical quantities are approximated with the neighboring information in a kernel. Consider a function $f(X)$ at X_i can be approximated by $\hat{g}(X_i)$ with the integration

$$\hat{g}(X_i) = \int f(X)W(X - X_i)\mathrm{d}X, \tag{A.1}$$

where $W(X)$ is a weighting kernel which is chosen as a third-order polynomial

$$W(R_j, h) = A \begin{cases} (h - R_j)^3, & R_j < h, \\ 0, & R_j \leq h, \end{cases} \tag{A.2}$$

h denotes the radius of the integration kernel; R_j is defined as the distance between X_j and the reference point in the kernel X_i with $A = 10/\pi h^5$ in two dimensions or $A = 10/\pi h^6$ in three dimensions. SPH numerically approximates the integration as

$$g(X_i) = \sum_{j \in S} f_j V_j W(R_j, h),$$

(A.3)

where V_j is the volume of particle j. The gradient of $g(X_i)$ with respect to its reference coordinate is

$$\nabla_x g(X_i) = \sum_{j \in S} f_j V_j \nabla_x W(R_j, h),$$

(A.4)

with

$$\nabla_x W(R_j, h) = \left(\frac{\partial W(R_j, h)}{\partial R_j} \right) \frac{R_j}{R_j}.$$

(A.5)

For a vector $f(X)$, its gradient to the referential coordinate is approximated as the following using the integration rule:

$$\nabla_x g(X_i) = \sum_{j \in S} f_j \otimes V_j \nabla_x W(R_j, h).$$

(A.6)

We then introduce two *ad-hoc* corrections for keeping symmetrization and first-order completeness [94].

$$\nabla_x g(X_i) = \sum_{j \in S} (f_j - f_i) V_j \nabla_x W(R_j, h).$$

(A.7)

Another correction guarantees the first-order completeness [95], the corrected gradient tensor is defined as

$$\widetilde{\nabla}_x W(R_j, h) = A^{-1} \nabla_x W(R_j, h),$$

(A.8)

where the shape tensor A is

$$A_i = \sum_{j \in S} V_j \nabla_x W(R_j, h) \otimes R_j.$$

(A.9)

We introduce the SPH integration to the governing equation of solid at the continuum level. The equilibrium equation is:

$$\nabla \cdot P + \rho_0 \mathbf{b} = \rho_0 \ddot{x},$$

(A.10)

where P is the first Piola–Kirchhoff stress tensor, $\mathbf{b}$ the body force vector, ρ_0 the referential mass density, and $\ddot{x}$ the acceleration. We define the deformation gradient as

$$F_i = \frac{\partial x_i}{\partial X_i} = \sum_{j \in S} r_j \otimes V_j \widetilde{\nabla}_x W(R_j, h).$$

(A.11)

According to basic continuum mechanics law,

$$P_i = F_i S_i = 2F_i \frac{\partial \mathscr{W}}{\partial C}.$$

(A.12)

As proved in [96], the internal forces emerge from the divergence of the first Piola–Kirchhoff stress P:

$$m_i \ddot{x}_i = f^{int} + f^{ext},$$

(A.13)

where the internal force is expressed as

$$f_i^{int} = \sum_{j \in S} V_i V_j \left(P_i \widetilde{\nabla}_x W(R_i, h) - P_j \widetilde{\nabla}_x W(R_i, h) \right).$$

(A.14)

We adopt another correction for suppressing spurious hourglassing mode [79]

$$f_i^{hg} = \sum_{j \in S} -\alpha \frac{E V_i V_j W(R_j, h)}{2 R_j^2} (\delta_i + \delta_j) \frac{r_j}{r_j}.$$

(A.15)

We adopt the strain energy function $\mathscr{W}(\mathbf{C}, \mathbf{M})$ for a fiber-enhanced tissue proposed in [85]

$$\mathscr{W}(C, M) = \frac{c}{2}(I_1 - 3) - c\ln(J) + \frac{k_1}{2k_2} \sum_{i=1}^{2} \left(\exp\left(k_2 \langle E_i \rangle^2\right) - 1 \right)$$
$$+ \frac{K_0}{2} \left(\frac{J^2 - 1}{2} - \ln J \right),$$

(A.16)

where principal invariants, I_1 and I_4 are defined as,

$$I_1 = C:I, J = \det F, I_4 = C:M \otimes M.$$

(A.17)

The parameters are set the same as the FEM model in Section 3.4. We refer the reader to [93,97] for more details.
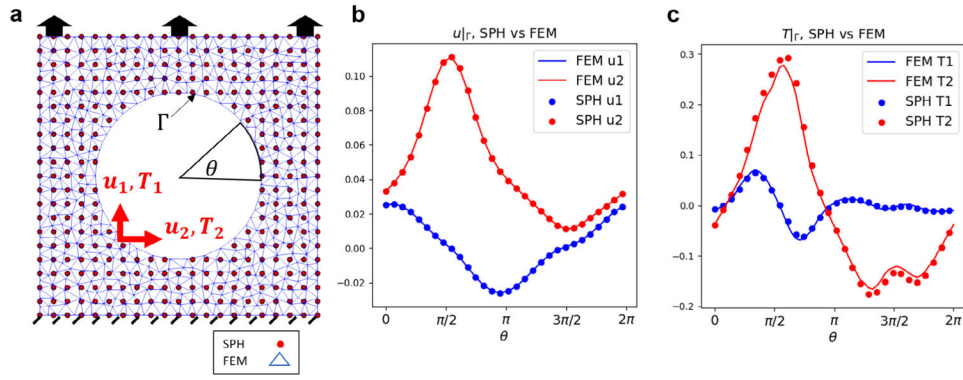
Fig. B.14.
Comparison of SPH and FEM. (a) We impose the same Dirichlet boundary condition on the top and internal circle for a FEM and SPH model. (b–c) Interfacial displacement and traction of both models.

Problem	Branch Size	Trunk Size	Train Epochs
Poisson	[84, 100, 100, 100]	[2, 100, 100, 100]	100,000
Heat	[51, 100, 100, 100]	[2, 100, 100, 100]	100,000
Elastoplastic	[120, 150, 150, 150]	[2, 150, 150, 150]	1,000,000
Hyperelastic	[120, 150, 150, 150]	[2, 150, 150, 150]	1,000,000

Fig. C.15.
Network size for various problems. We tabulate the branch/trunk network size for the four examples in Section 3 with the number of training epochs.

Appendix B.: Accuracy of FEM and SPH

B.1. Accuracy of SPH

To compare the numerical solution of the FEM and SPH model, We present their computational results for a same benchmark problem. The problem is set up as shown in Fig. B.14(a). We impose a distributed traction boundary condition on the top edge and impose a displacement boundary condition (Dirichlet-type) at the interface for both models shown in Fig. B.14(b): the displacement in x and y at the interface are plotted against the angle θ. The corresponding tractions at the interface are presented in Fig. B.14(c). The solid lines are the FEM tractions in x and y directions with the dashed lines denoting the SPH results. The results correspond well in general with some deviation around $\theta = \pi/2$ and $\theta = 3\pi/2$ due to the difference in numerical integration methods. This discrepancy partially contributes to the overall errors observed in Section 3.4.

Appendix C.: Network training details

We explain the details of network training in this section. Network architecture for each problem is presented in Fig. C.15. We adopt a fully-connected neural network (FNN) with four layers for all the sub-networks. The input layer dimension of the branch net depends

on the number of points at the interface, which ranges from 51 to 120 in our examples. Three hidden layers are concatenated with the input layer where we adopt the hyperbolic tangent function as the activate function. Each hidden layer has 100 or 150 neurons shown in Fig. C.15. To minimize the loss function for each case, we set the training epoch as 100,000 steps for the heat and Poisson example and 1,000,000 for the other two examples. Moreover, we illustrate the error history of a few cases in Fig. C.16 where each case shows a relatively small difference between the training and testing errors. The low testing errors indicating a good generalization capability of the network. More specifically, we tabulate the input/output of each network in Fig. C.17 with MSEs of the training and testing data for each case.
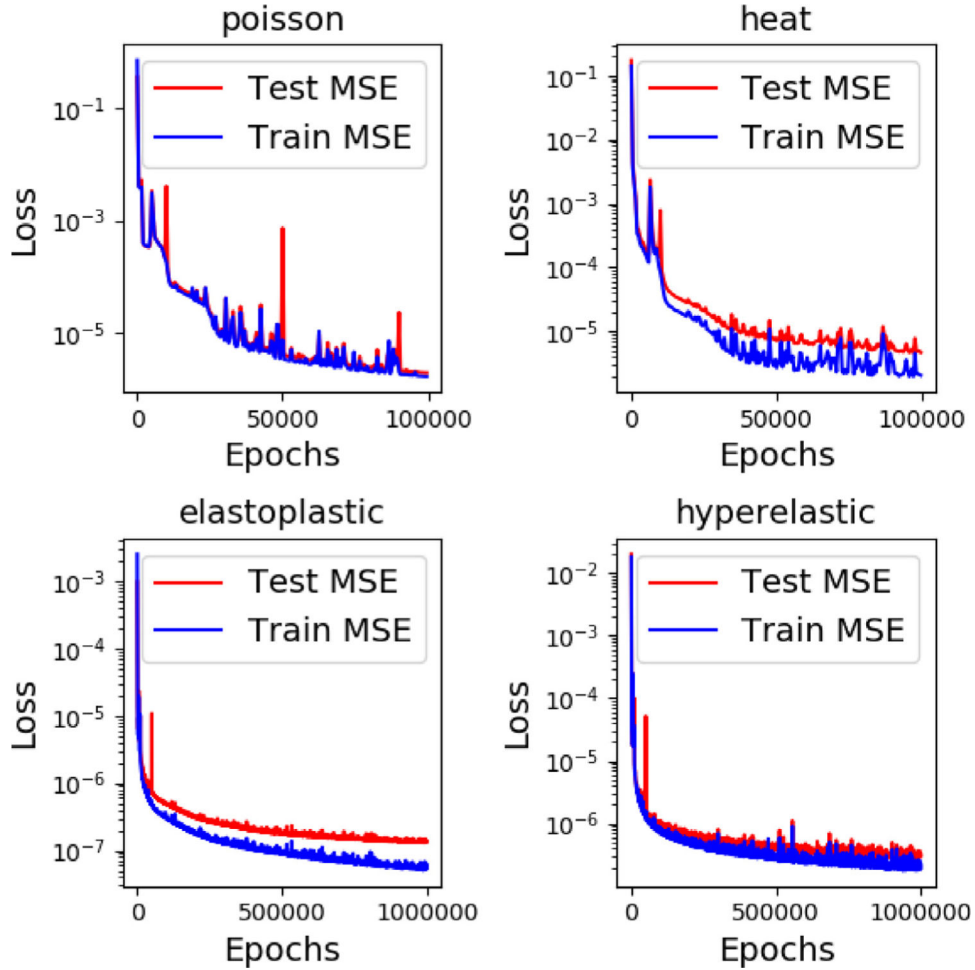
Fig. C.16.
History of training/testing errors of DeepONet for different problems.

We define the following metrics proposed in [98] to quantitatively characterize the efficiency of our framework.

$$R_t = \frac{C_t}{N_s C_s}, R_e = \frac{C_e}{C_s}, N_e^* = N_s + \frac{C_t}{C_s}. \tag{C.1}$$

Here, R_t is the training ratio where the time cost for training the DeepONet (C_t) is divided by the number of simulations (N_s) and the cost in time (C_s) for each simulation in the training dataset. R_e represents the ratio of time on one forward prediction in DeepONet over performing a standard numerical solver. The break even time N_e^* indicates the number of evaluations in the numerical solver where training a DeepONet costs as much as using numerical solvers. Beyond this point, surrogate modeling will be more efficient than a classical PDE solver. Taking the elastoplasticity case as an example, each elastoplastic FEM takes approximately 30 s($C_s = 30$ s) on a single CPU where we generated 1000($N_s = 1000$) samples as the training data. The DeepONet training takes 62 min ($C_t = 3720$ s) on a single GPU. The time cost of each forward evaluation using DeepONet (C_e) is 0.0014 s. Thus, using this information, we obtained $R_t = 0.124$, $R_e = 4.6 \times 10^{-5}$, and $N_e^* = 1124$. That means, as far as the resultant surrogate model is used for more than 1124 times in prediction tasks, DeepONet will be a more efficient choice.

Problem	Branch Input	Trunk Input	Output	MSE_{train}	MSE_{test}
Poisson	$u\|_\Gamma$	$[x, y]$	u	1.675×10^{-6}	1.946×10^{-6}
Heat	u^n	$[x, \frac{\partial u}{\partial x}\|_\Gamma]$	$\frac{\partial^2 u^{n+1}}{\partial x^2}$	1.877×10^{-6}	4.471×10^{-6}
Elastoplastic	$\mathbf{u}\|_\Gamma$	$[x, y]$	ε_{11}	1.028×10^{-8}	4.523×10^{-8}
			ε_{12}	5.578×10^{-9}	1.183×10^{-8}
			ε_{22}	4.854×10^{-8}	1.262×10^{-7}
Hyperelastic	$\mathbf{u}\|_\Gamma$	$[x, y]$	P_{11}	1.355×10^{-8}	2.347×10^{-8}
			P_{12}	6.393×10^{-9}	1.278×10^{-8}
			P_{21}	7.540×10^{-9}	$1,798 \times 10^{-8}$
			P_{22}	1.627×10^{-7}	2.701×10^{-7}
	$\mathbf{T}\|_\Gamma$	$[x, y]$	u_1	3.396×10^{-10}	2.430×10^{-9}
			u_2	1.716×10^{-9}	3.841×10^{-9}

Fig. C.17.
Details of network setup (inputs/outputs and MSE errors of training/testing) for each example in Section 3.

Appendix D.: Data generation

D.1.   Random field generation

We provide an overview of the algorithms of random fields generation using Fast Fourier Transformation (FFT). Let $W(\mathbf{x})$ be a Gaussian white noise random field on $\mathbb{R}^d$. A random field $\phi(\mathbf{x})$ can be sampled by

$$\phi(\mathbf{x}) = \mathscr{F}^{-1}\big(\gamma^{1/2}\mathscr{F}(W)\big)(\mathbf{x}), \tag{D.1}$$

where $\mathscr{F}$ and $\mathscr{F}^{-1}$ denote Fourier transformation and its inverse. γ represents a correlation function $\| k \|^{-\alpha}$ where $\| k \|$ is the L_2 norm of the wave number $k \in \mathbb{R}^d$. In Sections 3.1, 3.2,

and 3.4, we adopt this method for generating the training data with $\alpha = 5$. We refer the reader to find more theoretical details in [99]

D.2. Sampling in elastoplasticity

In the case of elastoplasticity, the non-uniform tension $t_0(x)$ (see Fig. 9) is generated by

$$t_0(x) = \sum_{i=1}^{3} \frac{A_i}{i}\cos(ix) + \sum_{i=1}^{3} \frac{B_i}{i}\sin(ix), \tag{D.2}$$

where $A_i, B_i \sim N\left(0, 0.05^2\right)(i \in \{1, 2, 3\})$ are independent normal random variables. We choose the standard deviation as 0.05 to make sure that the dataset contains similar numbers of cases with and without plastic deformation.

References

[1]. Weinan E, Principles of Multiscale Modeling, Cambridge University Press, 2011.

[2]. Alber M, Tepole AB, Cannon WR, De S, Dura-Bernal S, Garikipati K, Karniadakis GE, Lytton WW, Perdikaris P, Petzold L, et al. , Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences, NPJ Digit. Med 2 (1) (2019) 1–11. [PubMed: 31304351]

[3]. Dobson M, Luskin M, Ortner C, Stability, instability, and error of the force-based quasicontinuum approximation, Arch. Ration. Mech. Anal 197 (1) (2010) 179–202.

[4]. Tinsley Oden J, Prudhomme S, Romkes A, Bauman PT, Multiscale modeling of physical phenomena: Adaptive control of models, SIAM J. Sci. Comput 28 (6) (2006) 2359–2389.

[5]. Bazilevs Y, Calo V, Cottrell J, Hughes T, Reali A, Scovazzi G, Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, Comput. Methods Appl. Mech. Engrg 197 (1–4) (2007) 173–201.

[6]. Holian BL, Ravelo R, Fracture simulations using large-scale molecular dynamics, Phys. Rev B 51 (17) (1995) 11275.

[7]. Fish J, Wagner GJ, Keten S, Mesoscopic and multiscale modelling in materials, Nature Mater. 20 (6) (2021) 774–786. [PubMed: 34045697]

[8]. Wang Y, Li Z, Xu J, Yang C, Karniadakis GE, Concurrent coupling of atomistic simulation and mesoscopic hydrodynamics for flows over soft multi-functional surfaces, Soft Matter 15 (8) (2019) 1747–1757. [PubMed: 30672954]

[9]. Zhang T, Li X, Gao H, Fracture of graphene: a review, Int. J. Fract 196 (1–2) (2015) 1–31.

[10]. Jing N, Xue Q, Ling C, Shan M, Zhang T, Zhou X, Jiao Z, Effect of defects on Young's modulus of graphene sheets: a molecular dynamics simulation, RSC Adv. 2 (24) (2012) 9124–9129.

[11]. Ortiz M, A method of homogenization of elastic media, Internat. J. Engrg. Sci 25 (7) (1987) 923–934.

[12]. Lin P, Theoretical and numerical analysis for the quasi-continuum approximation of a material particle model, Math. Comp 72 (242) (2003) 657–675.

[13]. Xiao S, Belytschko T, A bridging domain method for coupling continua with molecular dynamics, Comput. Methods Appl. Mech. Engrg 193 (17–20) (2004) 1645–1669.

[14]. Nie X, Chen S, Robbins M, et al. , A continuum and molecular dynamics hybrid method for micro-and nano-fluid flow, J. Fluid Mech 500 (2004) 55–64.

[15]. Tran JS, Schiavazzi DE, Ramachandra AB, Kahn AM, Marsden AL, Automated tuning for parameter identification and uncertainty quantification in multi-scale coronary simulations, Comput. & Fluids 142 (2017) 128–138.

[16]. Kevrekidis IG, Samaey G, Equation-free multiscale computation: Algorithms and applications, Annu. Rev. Phys. Chem 60 (2009) 321–344. [PubMed: 19335220]

[17]. Theodoropoulos C, Qian Y-H, Kevrekidis IG, "Coarse" stability and bifurcation analysis using time-steppers: A reaction-diffusion example, Proc. Natl. Acad. Sci 97 (18) (2000) 9840–9843. [PubMed: 10963656]

[18]. Kevrekidis IG, Gear CW, Hyman JM, Kevrekidis PG, Runborg O, Theodoropoulos C, et al. , Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis, Commun. Math. Sci 1 (4) (2003) 715–762.

[19]. D'Elia M, Littlewood D, Trageser J, Perego M, Bochev P, An optimization-based strategy for peridynamic-FEM coupling and for the prescription of nonlocal boundary conditions, 2021, arXiv preprint arXiv:2110.04420.

[20]. Zohdi TI, Homogenization methods and multiscale modeling, in: Encyclopedia of Computational Mechanics Second Edition, Wiley Online Library, 2017, pp. 1–24.

[21]. Bensoussan A, Lions J-L, Papanicolaou G, Asymptotic Analysis for Periodic Structures, Vol. 374, American Mathematical Soc., 2011.

[22]. Weinan E, Engquist B, Multiscale modeling and computation, Notices Amer. Math. Soc 50 (9) (2003) 1062–1070.

[23]. Efendiev Y, Galvis J, Hou TY, Generalized multiscale finite element methods (GMsFEM), J. Comput. Phys 251 (2013) 116–135.

[24]. You H, Yu Y, Silling S, D'Elia M, A data-driven peridynamic continuum model for upscaling molecular dynamics, Comput. Methods Appl. Mech. Engrg 389 (2022) 114400.

[25]. Blumers A, Yin M, Nakajima H, Hasegawa Y, Li Z, Karniadakis GE, Multiscale parareal algorithm for long-time mesoscopic simulations of microvascular blood flow in zebrafish, Comput. Mech 68 (2021) 1131–1152.

[26]. Milton GW, The Theory of Composites, Cambridge University Press, 2002

[27]. Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, Vogt-Maranto L, Zdeborová L, Machine learning and the physical sciences, Rev. Modern Phys 91 (4) (2019) 045002.

[28]. Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L, Physics-informed machine learning, Nat. Rev. Phys 3 (6) (2021) 422–440

[29]. Zhang L, Han J, Wang H, Car R, Weinan E, Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics, Phys. Rev. Lett 120 (14) (2018) 143001. [PubMed: 29694129]

[30]. Cai S, Mao Z, Wang Z, Yin M, Karniadakis GE, Physics-informed neural networks (PINNs) for fluid mechanics: A review, Acta Mech. Sinica (2022) 1–12.

[31]. Pfau D, Spencer JS, Matthews AG, Foulkes WMC, Ab initio solution of the many-electron Schrödinger equation with deep neural networks, Phys. Rev. Res 2 (3) (2020) 033429.

[32]. Wang K, Sun W, A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning, Comput. Methods Appl. Mech. Engrg 334 (2018) 337–380.

[33]. Arbabi H, Bunder JE, Samaey G, Roberts AJ, Kevrekidis IG, Linking machine learning with multiscale numerics: data-driven discovery of homogenized equations, JOM 72 (12) (2020) 4444–4457.

[34]. Rahman AS, Hosono T, Quilty JM, Das J, Basak A, Multiscale groundwater level forecasting: coupling new machine learning approaches with wavelet transforms, Adv. Water Resour 141 (2020) 103595

[35]. Peng GC, Alber M, Tepole AB, Cannon WR, De S, Dura-Bernal S, Garikipati K, Karniadakis GE, Lytton WW, Perdikaris P, et al. , Multiscale modeling meets machine learning: What can we learn? Arch. Comput. Methods Eng 28 (3) (2021) 1017–1037. [PubMed: 34093005]

[36]. Regazzoni F, Dedè L, Quarteroni A, Machine learning of multiscale active force generation models for the efficient simulation of cardiac electromechanics, Comput. Methods Appl. Mech. Engrg 370 (2020) 113268.

[37]. Chattopadhyay A, Hassanzadeh P, Subramanian D, Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network, Nonlinear Process. Geophys 27 (3) (2020) 373–389.

[38]. Bhatia H, Carpenter TS, Ingólfsson HI, Dharuman G, Karande P, Liu S, Oppelstrup T, Neale C, Lightstone FC, Van Essen B, et al. , Machine-learning-based dynamic-importance sampling for adaptive multiscale simulations, Nat. Mach. Intell 3 (5) (2021) 401–409.

[39]. Masi F, Stefanou I, Thermodynamics-based artificial neural networks (TANN) for multiscale modeling of materials with inelastic microstructure, 2021, arXiv preprint arXiv:2108.13137.

[40]. Wu L, Zulueta K, Major Z, Arriaga A, Noels L, Bayesian inference of non-linear multiscale model parameters accelerated by a deep neural network, Comput. Methods Appl. Mech. Engrg 360 (2020) 112693.

[41]. Pled F, Desceliers C, Zhang T, A robust solution of a statistical inverse problem in multiscale computational mechanics using an artificial neural network, Comput. Methods Appl. Mech. Engrg 373 (2021) 113540.

[42]. Xu X, D'Elia M, Glusa C, Foster JT, Machine-learning of nonlocal kernels for anomalous subsurface transport from breakthrough curves, 2022, arXiv preprint arXiv:2201.11146.

[43]. Park J, Zhu X, Physics-informed neural networks for learning the homogenized coefficients of multiscale elliptic equations, 2022, arXiv preprint arXiv:2202.09712v1.

[44]. Chan S, Elsheikh AH, A machine learning approach for efficient uncertainty quantification using multiscale methods, J. Comput. Phys 354 (2018) 493–511.

[45]. Rocha I, Kerfriden P, van der Meer F, On-the-fly construction of surrogate constitutive models for concurrent multiscale mechanical analysis through probabilistic machine learning, J. Comput. Phys.: X 9 (2021) 100083.

[46]. Pyrialakos S, Kalogeris I, Sotiropoulos G, Papadopoulos V, A neural network-aided Bayesian identification framework for multiscale modeling of nanocomposites, Comput. Methods Appl. Mech. Engrg 384 (2021) 113937.

[47]. Ingólfsson HI, Neale C, Carpenter TS, Shrestha R, López CA, Tran TH, Oppelstrup T, Bhatia H, Stanton LG, Zhang X, et al. , Machine learning-driven multiscale modeling reveals lipid-dependent dynamics of RAS signaling proteins, Proc. Natl. Acad. Sci 119 (1) (2022)

[48]. Liu B, Kovachki N, Li Z, Azizzadenesheli K, Anandkumar A, Stuart AM, Bhattacharya K, A learning-based multiscale method and its application to inelastic impact problems, J. Mech. Phys. Solids 158 (2022) 104668.

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

[50]. Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, Anandkumar A, Neural operator: Graph kernel network for partial differential equations, 2020, arXiv preprint arXiv:2003.03485.

[51]. Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell 3 (3) (2021) 218–229.

[52]. You H, Yu Y, D'Elia M, Gao T, Silling S, Nonlocal kernel network (NKN): a stable and resolution-independent deep neural network, 2022, arXiv preprint arXiv:2201.02217.

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

[54]. Cai S, Wang Z, Lu L, Zaki TA, Karniadakis GE, DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks, J. Comput. Phys 436 (2021) 110296

[55]. Lin C, Maxey M, Li Z, Karniadakis GE, A seamless multiscale operator neural network for inferring bubble dynamics, J. Fluid Mech 929 (2021).

[56]. Yin M, Ban E, Rego BV, Zhang E, Cavinato C, Humphrey JD, Karniadakis GE, Simulating progressive intramural damage leading to aortic dissection using DeepONet: an operator-regression neural network, J. R. Soc. Interface 19 (2021) 20140397.

[57]. Lin C, Li Z, Lu L, Cai S, Maxey M, Karniadakis GE, Operator learning for predicting multiscale bubble growth dynamics, J. Chem. Phys 154 (10) (2021) 104118. [PubMed: 33722055]

[58]. Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, Anandkumar A, Multipole graph neural operator for parametric partial differential equations, 2020, arXiv preprint arXiv:2006.09535.

[59]. Goswami S, Yin M, Yu Y, Karniadakis GE, A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials, Comput. Methods Appl. Mech. Engrg 391 (2022) 114587.

[60]. Mao Z, Lu L, Marxen O, Zaki TA, Karniadakis GE, DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators, J. Comput. Phys 447 (2021) 110698.

[61]. Lu L, Meng X, Cai S, Mao Z, Goswami S, Zhang Z, Karniadakis GE, A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data, 2021, arXiv preprint arXiv:2111.05512.

[62]. Kovachki N, Li Z, Liu B, Azizzadenesheli K, Bhattacharya K, Stuart A, Anandkumar A, Neural operator: Learning maps between function spaces, 2021, arXiv preprint arXiv:2108.08481

[63]. Chen T, Chen H, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Trans. Neural Netw 6 (4) (1995) 911–917. [PubMed: 18263379]

[64]. Lanthaler S, Mishra S, Karniadakis GE, Error estimates for deeponets: A deep learning framework in infinite dimensions, 2021, arXiv preprint arXiv:2102.09618.

[65]. Patera AT, A spectral element method for fluid dynamics: laminar flow in a channel expansion, J. Comput. Phys 54 (3) (1984) 468–488.

[66]. Karniadakis GE, Sherwin S, Spectral/Hp Element Methods for Computational Fluid Dynamics, Oxford University Press, 2005.

[67]. Hughes TJ, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Courier Corporation, 2012.

[68]. Versteeg HK, Malalasekera W, An Introduction to Computational Fluid Dynamics: The Finite Volume Method, Pearson education, 2007.

[69]. Monaghan JJ, Smoothed particle hydrodynamics, Annu. Rev. Astron. Astrophys 30 (1) (1992) 543–574.

[70]. Espanol P, Warren P, Statistical mechanics of dissipative particle dynamics, Europhys. Lett 30 (4) (1995) 191.

[71]. Groot RD, Warren PB, Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation, J. Chem. Phys 107 (11) (1997) 4423–4435

[72]. Rapaport DC, Rapaport DCR, The Art of Molecular Dynamics Simulation, Cambridge University Press, 2004.

[73]. Yu Y, Bargos FF, You H, Parks ML, Bittencourt ML, Karniadakis GE, A partitioned coupling framework for peridynamics and classical theory: analysis and simulations, Comput. Methods Appl. Mech. Engrg 340 (2018) 905–931.

[74]. Mok D, Wall W, Ramm E, Accelerated iterative substructuring schemes for instationary fluid-structure interaction, Comput. Fluid Solid Mech 2 (2001) 1325–1328.

[75]. Lions P-L, et al., On the Schwarz alternating method. I, in: First International Symposium on Domain Decomposition Methods for Partial Differential Equations, Vol. 1, Paris, France, 1988, p. 42.

[76]. Mota A, Tezaur I, Alleman C, The Schwarz alternating method in solid mechanics, Comput. Methods Appl. Mech. Engrg 319 (2017) 19–51.

[77]. Funaro D, Quarteroni A, Zanolli P, An iterative procedure with interface relaxation for domain decomposition methods, SIAM J. Numer. Anal 25 (6) (1988) 1213–1236.

[78]. Abaqus Abaqus 2020 Documentation, Dassault Systèmes, 2020

[79]. Ganzenmüller GC, An hourglass control algorithm for Lagrangian smooth particle hydrodynamics, Comput. Methods Appl. Mech. Engrg 286 (2015) 87–106.

[80]. Alnæs M, Blechta J, Hake J, Johansson A, Kehlet B, Logg A, Richardson C, Ring J, Rognes ME, Wells GN, The FEniCS project version 1.5, Arch. Numer. Softw 3 (100) (2015).

[81]. Wang S, Wang H, Perdikaris P, Learning the solution operator of parametric partial differential equations with physics-informed DeepONets, Sci. Adv 7 (40) (2021) eabi8605. [PubMed: 34586842]

[82]. Yin M, Zheng X, Humphrey JD, Karniadakis GE, Non-invasive inference of thrombus material properties with physics-informed neural networks, Comput. Methods Appl. Mech. Engrg 375 (2021) 113603.

[83]. Zhang E, Yin M, Karniadakis GE, Physics-informed neural networks for nonhomogeneous material identification in elasticity imaging, 2020, arXiv preprint arXiv:2009.04525.

[84]. Zhang E, Dao M, Karniadakis GE, Suresh S, Analyses of internal structures and defects in materials using physics-informed neural networks, Sci. Adv 8 (7) (2022) eabk0644. [PubMed: 35171670]

[85]. Holzapfel GA, Gasser TC, Ogden RW, A new constitutive framework for arterial wall mechanics and a comparative study of material models, J. Elast. Phys. Sci. Solids 61 (1) (2000) 1–48.

[86]. Dijkstra W, Mattheij R, The condition number of the BEM-matrix arising from Laplace's equation, Electron. J. Bound. Elem 4 (2) (2006).

[87]. Gustafson K, Domain decomposition, operator trigonometry, Robin condition, Contemp. Math 218 (1998) 432–437.

[88]. Douglas J, Huang C-S, An accelerated domain decomposition procedure based on Robin transmission conditions, BIT Numer. Math 37 (3) (1997) 678–686.

[89]. Jin H, Jiao T, Clifton RJ, Kim K-S, Dynamic fracture of a bicontinuously nanostructured copolymer: A deep-learning analysis of big-data-generating experiment, J. Mech. Phys. Solids (2022) 104898.

[90]. Tadmor EB, Ortiz M, Phillips R, Quasicontinuum analysis of defects in solids, Phil. Mag. A 73 (6) (1996) 1529–1563.

[91]. Brandt A, Multi-level adaptive solutions to boundary-value problems, Math. Comp 31 (138) (1977) 333–390.

[92]. Weinan E, Engquist B, et al. , The heterognous multiscale methods, Commun. Math. Sci 1 (1) (2003) 87–132.

[93]. Rausch M, Karniadakis GE, Humphrey JD, Modeling soft tissue damage and failure using a combined particle/continuum approach, Biomech. Model. Mechanobiol 16 (1) (2017) 249–261. [PubMed: 27538848]

[94]. Monaghan JJ, An introduction to SPH, Comput. Phys. Comm 48 (1) (1988) 89–96.

[95]. Randles P, Libersky LD, Smoothed particle hydrodynamics: some recent improvements and applications, Comput. Methods Appl. Mech. Engrg 139 (1–4) (1996) 375–408.

[96]. Bonet J, Lok T-S, Variational and momentum preservation aspects of smooth particle hydrodynamic formulations, Comput. Methods Appl. Mech. Engrg 180 (1–2) (1999) 97–115.

[97]. Ahmadzadeh H, Rausch M, Humphrey JD, Modeling lamellar disruption within the aortic wall using a particle-based approach, Sci. Rep 9 (1) (2019) 1–17 [PubMed: 30626917]

[98]. Di Leoni PC, Lu L, Meneveau C, Karniadakis G, Zaki TA, Deeponet prediction of linear instability waves in high-speed boundary layers, 2021, arXiv preprint arXiv:2105.08697.

[99]. Lang A, Potthoff J, Fast simulation of Gaussian random fields, Monte Carlo Methods Appl. 17 (3) (2011) 195–214.
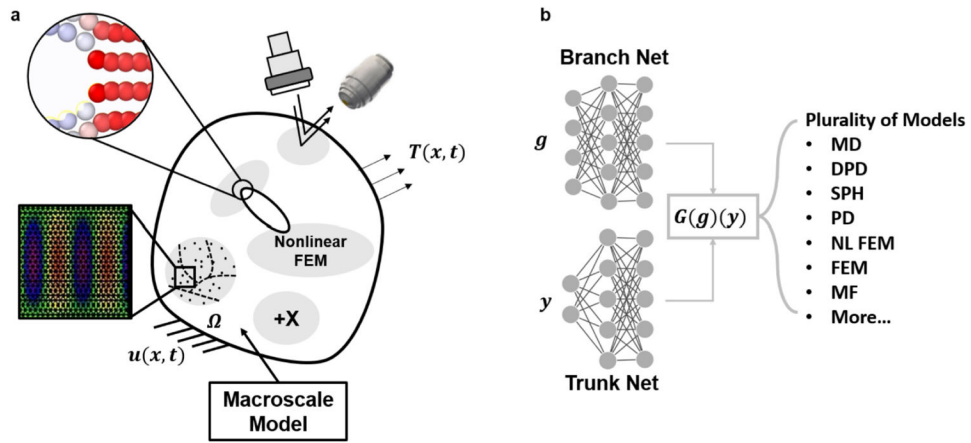
Fig. 1.

Schematic of multiscale modeling with DeepONet. (a) In a multiscale mechanics system, traction $T(x, t)$ acts on the boundary of domain Ω, which is decomposed into sub-domains described by a variety of microscopic models. A macroscopic model describes the response in the bulk region whereas nonlinear, microscopic, or data-driven models capture the detailed response in the small regions. (b) DeepONet, composed of a branch and a trunk net, is able to learn the response of the microscopic systems (from simulated or multi-modal data) and serve as a surrogate in the multiscale system. MD: molecular dynamics, DPD: dissipative particle dynamics, SPH: smoothed particle hydrodynamics, PD: peridynamics, NL FEM: nonlinear finite element, FEM: finite element, MF: multifidelity.
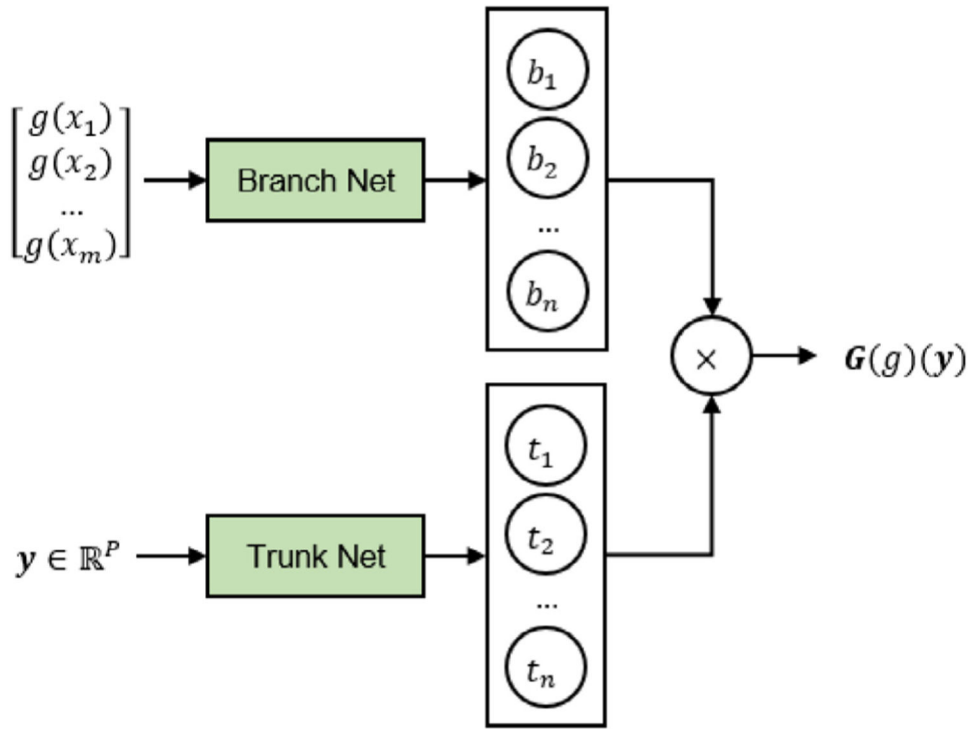
Fig. 2.
Schematic architecture of DeepONet. DeepONet learns the mapping operator G from an input function g to its corresponding output function $G(g)$. The input of branch and trunk net are g and $\mathbf{y} \in \mathbb{R}^P$, which represents a discretized function from $g(x_1)$ to $g(x_m)$ and information such as coordinates and time, respectively. Note that the output dimension of the trunk net, n, is consistent with that of the branch net. The final output $G(g)(\mathbf{y})$ is computed as the dot product of $\mathbf{b}$ and $\mathbf{t}$
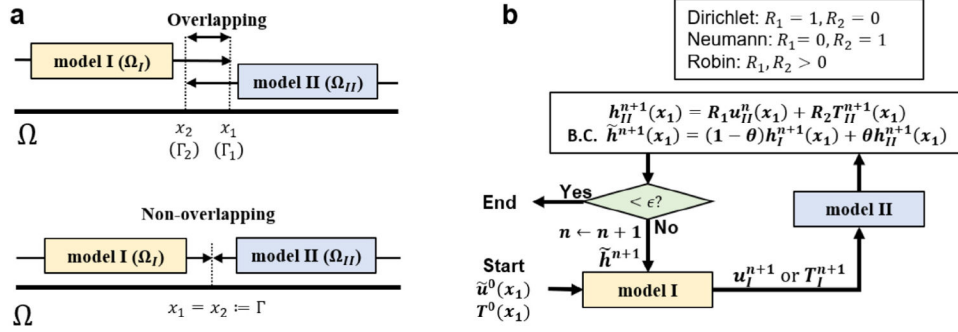
Fig. 3.
A flexible framework in domain decomposition. The proposed coupling framework is able to adopt either (a) overlapping or non-overlapping domain decomposition coupled with Dirichlet, Neumann, or Robin boundary conditions at the interface. As an illustration, (b) presents a Robin-type boundary condition imposed on model I with interfacial solution $\hat{u}(x_1)$ updated by a relaxation scheme. Dirichlet and Neumann boundary can be imposed by adjusting the value of R_1 and R_2. The relaxation parameter, θ, is either fixed at a value or updated dynamically. $T_{II}^n(x_1)$ represents the Neumann-related information at x_1 from Model II, e.g., traction or derivatives.
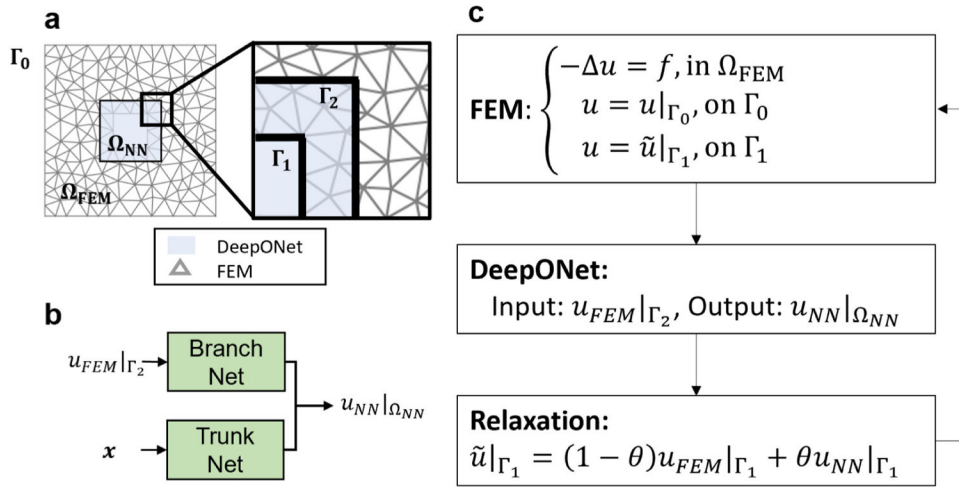
Fig. 4.
Setup of the Poisson equation. (a) A 1×1 unit square is decomposed into two overlapping regions, Ω_{NN} (light blue) and Ω_{FEM} (meshed). Ω_{FEM} is bounded by Γ_1 and Γ_0, whereas Ω_{NN} is a 0.3×0.3 square in the center with boundary Γ_2. (b) DeepONet takes $u_{FEM}|_{\Gamma_2}$ as input and yields the solution in Ω_{NN}, denoted as $u_{NN}|_{\Omega_{NN}}$. (c) Formulation of an overlapping D-D method. Given the boundary condition on Γ_0, the coupling framework iteratively updates the corresponding boundary condition on Γ_1. The relaxation parameter θ is either fixed or updated based on the Aitken's rule [73].
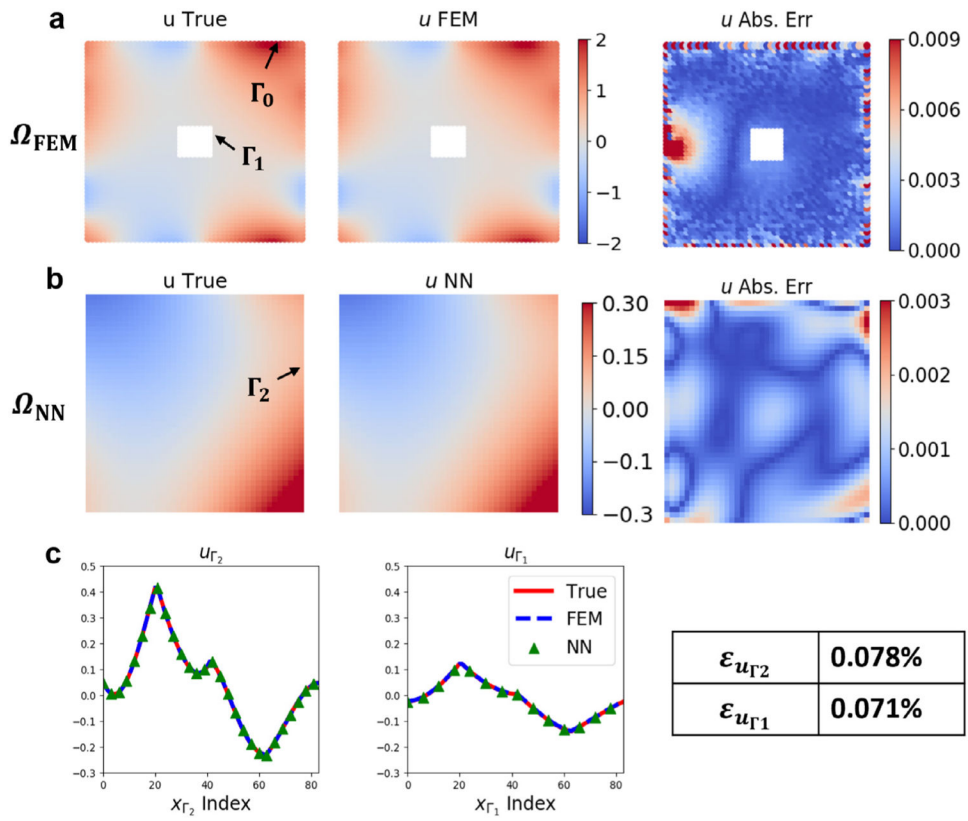
Fig. 5.
Results of coupling FEM and DeepONet for the Poisson equation. (a–b) Model predictions from the FEM and DeepONet with the corresponding absolute errors in Ω_{FEM} and Ω_{NN}. (c) Model predictions at the interfaces ($u|_{\Gamma_1}$ and $u|_{\Gamma_2}$) and the true solution (red line). The relative errors of model predictions at the interfaces are far less than 1%. Please see Fig. 4 for Ω_{NN} and Ω_{FEM}.
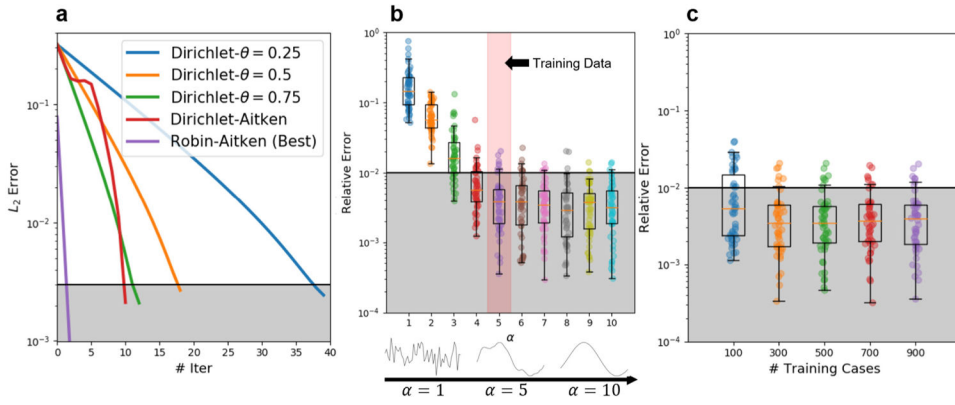
Fig. 6.

Parametric studies of the Poisson equation. (a) Given a boundary condition on Γ_0, the convergence history of the coupling model varies with the coupling method (Robin or Dirichlet) and parameter θ. (b) The model is trained with data generated from a random field with $\alpha = 5$. We test the generalization ability represented by the relative errors of the coupling model for cases corresponding to $\alpha = 1$ to 10. (c) Box plots of relative errors for DeepONets trained with 100 to 900 cases. Each column shows the coupling results by testing with 50 different cases. The shaded area indicates that the relative error on Γ_1 is less than 1%. Notice that we present the error with respect to the ground truth to show the convergence and accuracy of our framework.
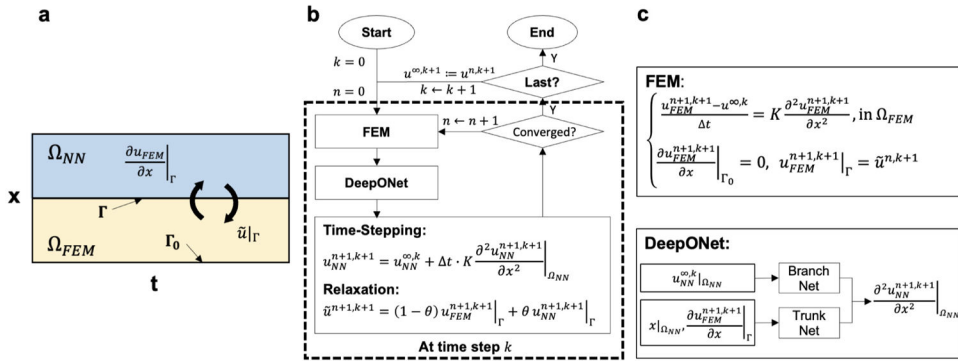
Fig. 7.
Setup of the time-dependent problem (heat equation). (a) The spatio-temporal domain is decomposed into two sub-domains with Γ as the interface between FEM and NN domain. (b–c) Formulation of the coupling framework for the heat equation. The framework first solves for $u_{FEM}^{n+1,k+1}$ from FEM with updated information at the interface $\tilde{u}$ at k th step. Then, the computed derivative at Γ is transmitted into DeepONet, which predicts the spatial derivative in Ω_{NN} (c). The solution in Ω_{NN} is computed based on a time-stepping scheme, followed by a relaxation update (b). The initialization at $k = 0$ and $n = 0$ is described in the main text.
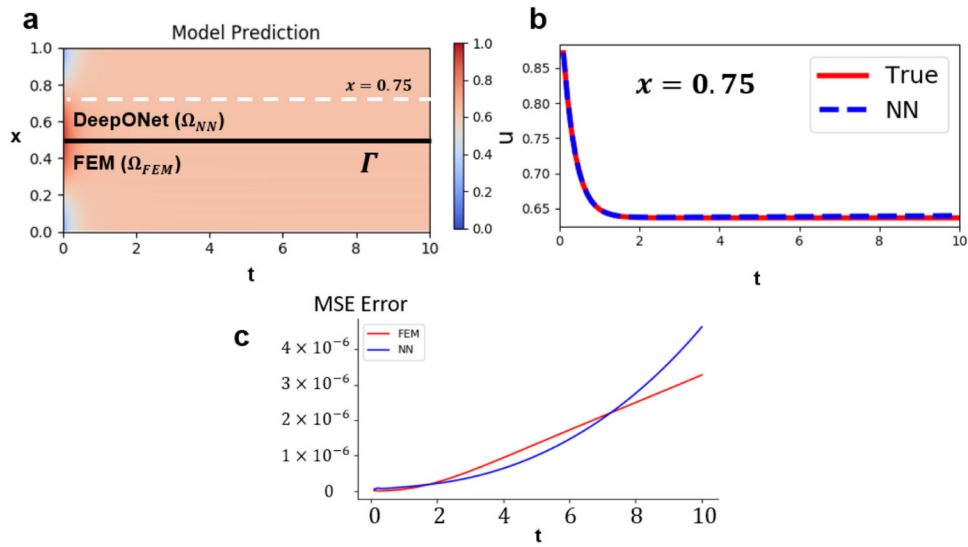
Fig. 8.

Results of coupling FEM and DeepONet for the heat equation. (a) Model predictions from DeepONet and FEM. DeepONet predicts the solution in $x \in \Omega_{NN}$ while FEM predicts the solution for $x \in \Omega_{FEM}$. The black line at $x = 0.5$ denotes the interface of the two domains. The solution at $x = 0.75$ (indicated by the white dashed line) is presented in (b). (c) The mean square errors of FEM (red) and DeepONet (blue) vs. time (t).
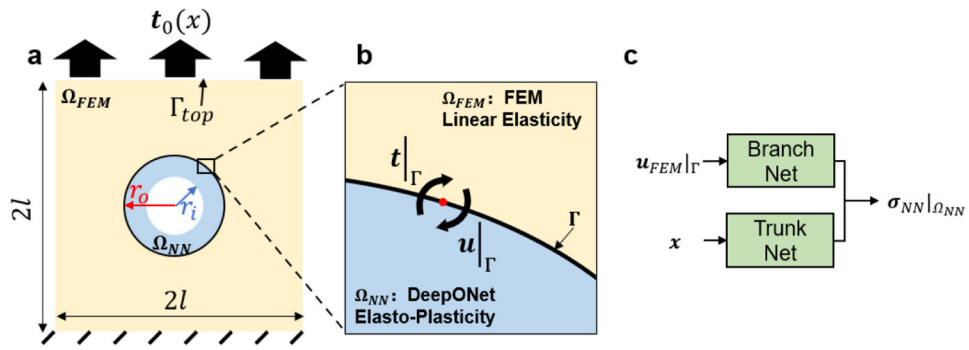
Fig. 9.

Setup of the elastoplasticity problem. (a) A $2l \times 2l$ solid plate is clamped on the bottom edge with a traction $\mathbf{t}_0$ distributed on the top edge. (b) The yellow region (Ω_{FEM}) is modeled with linear elasticity, whereas the blue region (Ω_{NN}) is modeled with elastoplasticity. Interfacial displacement (Dirichlet boundary condition) $\mathbf{u}|_\Gamma$ computed from FEM is transmitted to DeepONet as input. (c) The DeepONet estimates the stress components in Ω_{NN}, $\sigma_{NN}|_{\Omega_{NN}}$, based on which the interfacial traction $t_{NN}|_\Gamma$ can be calculated accordingly. The predicted traction is updated and provided to FEM as a Neumann-type boundary condition. In this example, we set the interior circle radius $r_i = 0.1$, the interface circle radius $r_o = 0.3$, and the plate size $l = 1$.
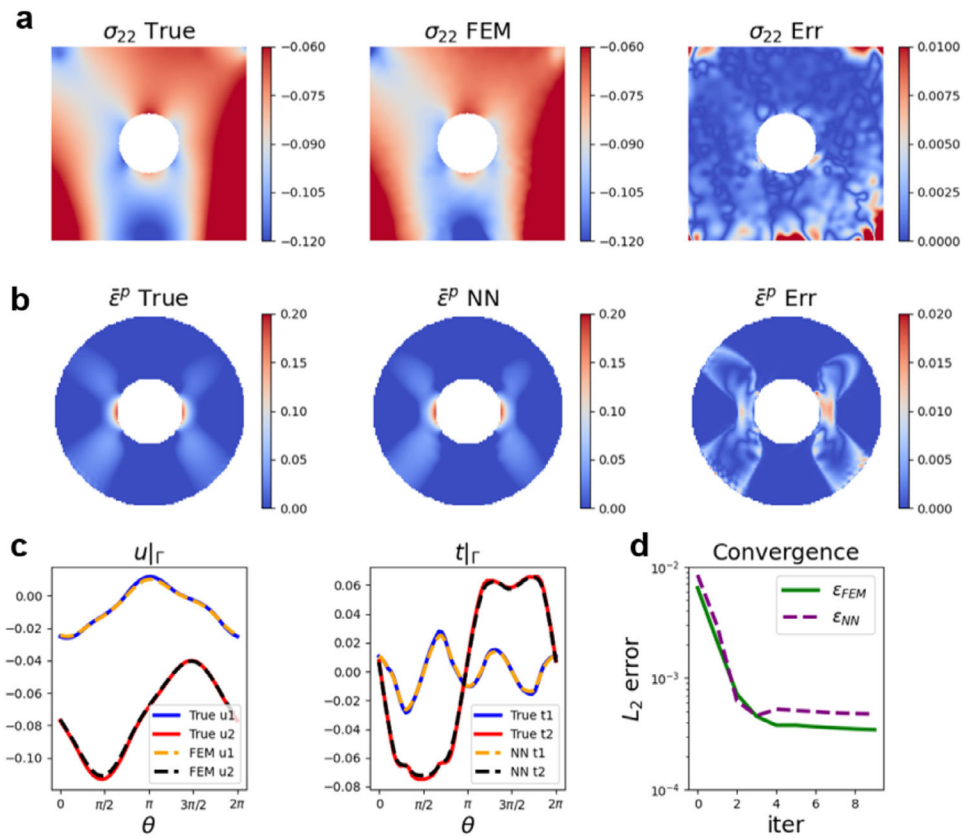
Fig. 10.

Results of coupling FEM and DeepONet in elastoplasticity. (a-b) Results of the normal stress in *y* direction (σ_{22}) and equivalent plastic strain ($\bar{\varepsilon}^p$). From left to right: True value, FEM/DeepONet predicted value from coupling, and their absolute errors. (c) Displacement and traction at the interface Γ. True values and predicted values from coupling are presented. Solid lines: displacement and traction of true data in x (blue) and y (red) directions. Dashed lines: displacement and traction of the model predictions in x (yellow) and y (black) directions. (d) The history of the relative errors of the coupling model at the interface. ε_{FEM} and ε_{NN} refer to the L_2 error of displacement and traction from FEM and DeepONet, respectively. Notice that we present the error with respect to the ground truth to show the convergence and accuracy of our framework.
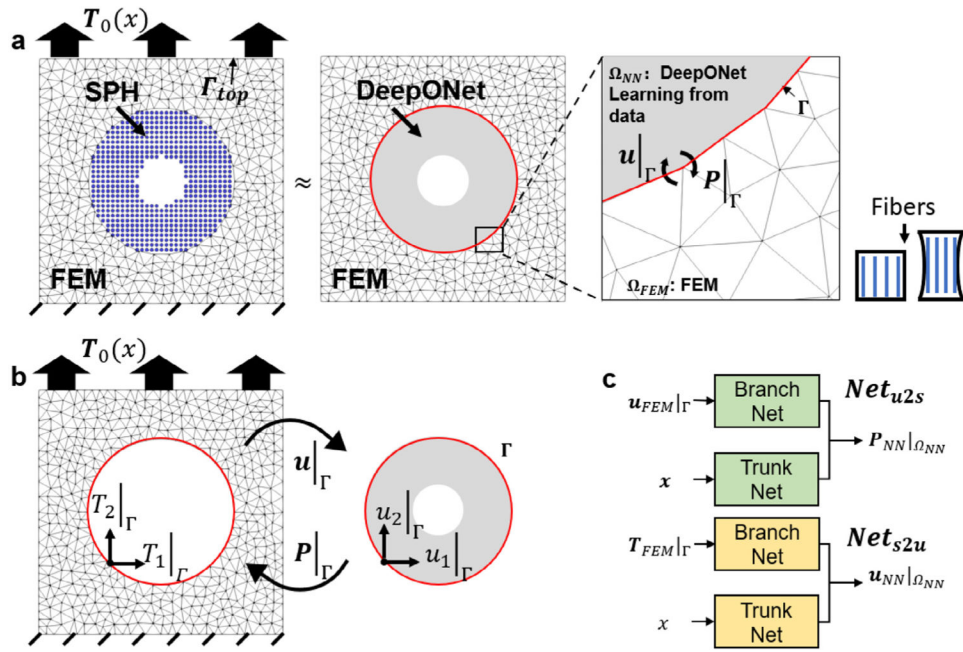
Fig. 11.

Setup of the hyperelasticity problem. (a) A unit square is decomposed into Ω_{NN} and Ω_{FEM}. We impose a non-uniform traction boundary condition on the top edge and fix the displacement at the bottom and train multiple DeepONets to represent the mechanics of an SPH model. The material is reinforced by fibers in the vertical direction. Information at the interface (displacement **u** and first Piola–Kirchhoff stress **P**) is transmitted between DeepONet and FEM. (b) Traction and displacement in different directions are exchanged at the interface Γ. FEM predicts the external domain while the DeepONet is trained based on SPH data. (c) Two types of DeepONet are proposed to predict the mechanics of the system: predicting stresses based on displacement information (green boxes) and vice versa (yellow boxes).
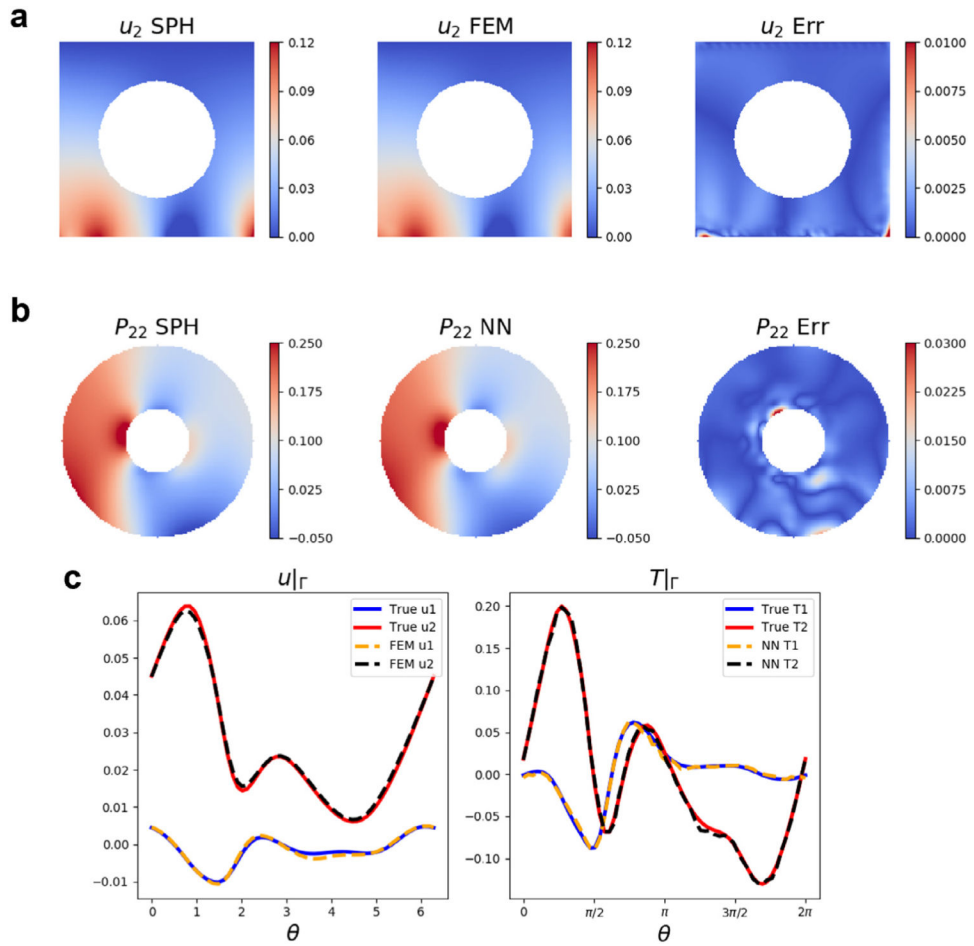
Fig. 12.
Results of coupling FEM and DeepONet in the hyperelasticity problem. Results of (a) displacement in Ω_{FEM} and (b) P_{22} in Ω_{NN}. From left to right: prediction from SPH, prediction from FEM, and their absolute differences. (c) Predicted displacement and traction at the interface. Predictions from both FEM and DeepONets are compared with the true data. Solid lines: displacement and traction of true data in x (blue) and y (red) directions. Dashed lines: displacement and traction of the model predictions in x (yellow) and y (black) directions.
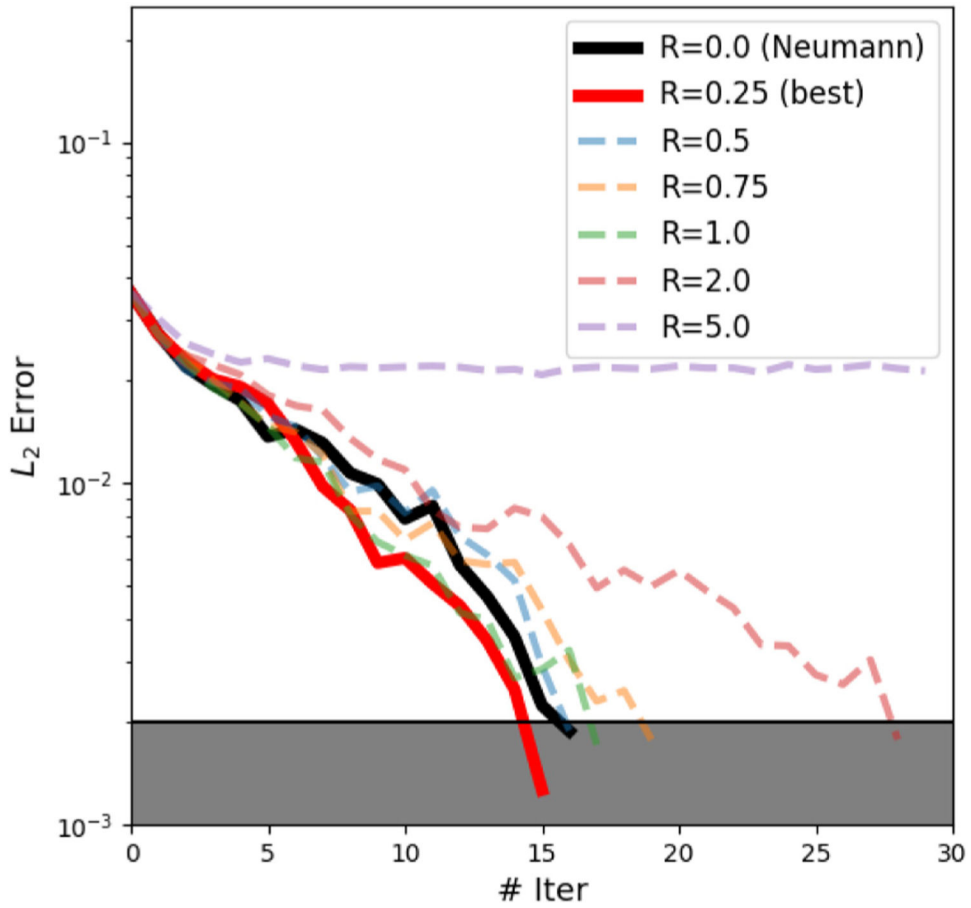
Fig. 13.

L_2 errors of the interfacial displacement in the hyperelastic multiscale model with a Robin boundary. The red line with Robin coefficient R = 0.25 converges fastest among the testing cases. The black solid line represents the relative error history of a Neumann boundary condition (R = 0). The shaded area indicates that the relative error is less than 2%. Notice that we present the error with respect to the ground truth to show the convergence and accuracy of our framework.

Table 1

Setup for the four examples.

Problem	Model I	Model II/Training data	Interface condition	Overlap
2D Poisson	FEM	DeepONet/FEM	D-D, R-D	Yes
1D Heat	FEM	DeepONet/FEM	N-D	No
2D Elastoplasticity	Linear elastic FEM	DeepONet/Elastoplastic FEM	N-D	No
2D Hyperelasticity	Hyperelastic FEM	DeepONet/Hyperelastic SPH	N-D, R-D/R-N	No

D-D: Dirichlet–Dirichlet, R-D: Robin–Dirichlet, N-D: Neumann–Dirichlet, R-N: Robin–Neumann.

Table 2

Parameters value of the HGO model.

Parameter	μ	K	k_1	k_2	α
Value	0.3846	0.8333	0.1	1.5	$\pi/2$

We set the value of k_1, k_2, and α the same for $i = 1$ and 2.

Table 3

Wall time comparison for SPH and DeepONet per iteration in the hyperelastic problem. The excessive cost of SPH is exacerbated because we use a time-dependent SPH solver to simulate a static problem.

Model	Wall time
FEM-SPH	~4 h
FEM-DeepONet	<1 s