Computer Methods in Applied Mechanics and Engineering 430 (2024) 117211

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

# Physics-informed deep learning of rate-and-state fault friction

Cody Rucker [a,*], Brittany A. Erickson [a,b]

[a] Department of Computer Science, University of Oregon, 1202 University of Oregon, 97403, Eugene, United States of America
[b] Department of Earth Sciences, University of Oregon, 1272 University of Oregon, Eugene, 97403, United States of America

ARTICLE INFO

Dataset link: https://github. com/Thrase/EQ_pinns

*Keywords:*
Physics-informed neural network
Rate-and-state friction
Earthquake
Inverse problem
Fully dynamic

ABSTRACT

Direct observations of earthquake nucleation and propagation are few and yet the next decade will likely see an unprecedented increase in indirect, surface observations that must be integrated into modeling efforts. Machine learning (ML) excels in the presence of large data and is an actively growing field in seismology. However, not all ML methods incorporate rigorous physics, and purely data-driven models can predict physically unrealistic outcomes due to observational bias or extrapolation. Our work focuses on the recently emergent Physics-Informed Neural Network (PINN), which seamlessly integrates data while ensuring that model outcomes satisfy rigorous physical constraints. In this work we develop a multi-network PINN for both the forward problem as well as for direct inversion of nonlinear fault friction parameters, constrained by the physics of motion in the solid Earth, which have direct implications for assessing seismic hazard. We present the computational PINN framework for strike–slip faults in 1D and 2D subject to rate-and-state friction. Initial and boundary conditions define the data on which the PINN is trained. While the PINN is capable of approximating the solution to the governing equations to low-errors, our primary interest lies in the network's capacity to infer friction parameters during the training loop. We find that the network for the parameter inversion at the fault performs much better than the network for material displacements to which it is coupled. Additional training iterations and model tuning resolves this discrepancy, enabling a robust surrogate model for solving both forward and inverse problems relevant to seismic faulting.

## 1. Context and motivation

Faults are home to a vast spectrum of event types, from slow aseismic creep, to slow-slip to megathrust earthquakes followed by postseismic afterslip. The Cascadia subduction zone in the Pacific Northwest, for example, hosts several types of slow earthquake processes including low (and very low) frequency earthquakes, non-volcanic tremor (NVT) and slow-slip events (SSE) [1], but also large, fast earthquakes, the last of which was a magnitude ~9 in the year 1700 [2]. Understanding the physical mechanisms for such diversity of slip styles is crucial for mitigating the associated hazards but major uncertainties remain in the depth-dependency of frictional properties at fault zones, which affect fault locking and therefore rupture potential [3,4]. Direct observations of earthquake nucleation and propagation are few and yet the next decade will likely see an unprecedented increase in indirect, surface observations that could be integrated into modeling efforts [5].

Traditional numerical approaches for solving the partial differential equations (PDE) governing earthquake processes (e.g. finite difference methods) have seen incredible growth in the past century, in particular in terms of convergence theory and high-performance computing. Traditional methods employ a mesh (either a finite number of grid points/nodes or elements) and a range

* Corresponding author.
*E-mail address:* crucker@uoregon.edu (C. Rucker).

https://doi.org/10.1016/j.cma.2024.117211
Received 23 April 2024; Received in revised form 25 June 2024; Accepted 3 July 2024

of time-integration schemes in order to obtain an approximate solution whose accuracy depends directly on the mesh size (with error decreasing with decreasing node spacing or element size). This mesh dependency introduces limitations when high resolution is needed, and while traditional methods are well-suited for solving forward problems, solving inverse problems require additional machinery and can be prohibitively expensive [6,7]. In addition, traditional methods require models to specify initial and boundary conditions in order to establish a well-posed problem, even when such conditions are unknown.

Machine learning (ML), on the other hand, excels in the presence of large data and is an actively growing field in seismology, with applications ranging from earthquake early warning (EEW) to ground-motion prediction, see for example Kong et al. [8], Lin et al. [9], Kubo et al. [10] and references therein. However, not all ML methods incorporate rigorous physics, and purely data-driven models can predict physically unrealistic outcomes due to observational bias or extrapolation [11]. A new Deep Learning technique has recently emerged called the Physics-Informed Neural Network (PINN), which seamlessly integrates sparse and/or noisy data while ensuring that model outcomes satisfy rigorous physical constraints. PINNs do not outperform traditional numerical methods for forward problems (except in high-dimensional settings) [12], but they offer advantages over traditional numerical methods in that both forward and inverse problems can be solved in the same computational framework and solutions may be learned even on ill-posed problems [13]. However, the majority of PINN applications are currently limited to simple mechanical models, forward problems and/or do not incorporate real-world observations [14–16]. Here we introduce a new, physically-rigorous modeling framework for both forward and inverse problems that can be integrated with observational data in order to better understand earthquake fault processes. This Deep Learning approach will vastly expand our computational abilities to explore and infer relevant parameter spaces responsible for slip complexity, and the conditions that enable the world's largest earthquakes.

Though the PINN framework lacks the robust error analysis that comes with traditional methods, a large number of publications have emerged since ∼2017 which aim to customize PINNs through the use of different activation functions, gradient optimization techniques, neural network architecture, and loss function structure [17]. Careful formulation of the loss function using the weak form of the PDE have been proposed for constructing deep learning analogues of Ritz [18] and Galerkin [19–21] methods which use numerical quadrature to reduce the order of the PDE resulting in a simpler learning problem [22,23]. In tandem, statistical learning theory has been used to deduce global error bounds for PINNs in terms of optimization error, generalization error, and approximation error [24]. For wide but shallow networks utilizing hyperbolic tangent activation functions, the approximation error has been shown to be bounded over Sobolev spaces [25]. Bounds on PINN generalization error have been derived for linear second-order PDE [26] (later extended to all linear problems [27]) and some specific cases like Navier–Stokes [28]. Moreover, the abstract framework for PINNs can leverage stability of a PDE to provide conditions under which generalization error is small whenever training error is small for both forward and inverse problems [29,30]. More recently, a PINN-specific optimization algorithm has achieved markedly improved accuracy over other optimization algorithms by incorporating a PDE energy into the backpropagation step [31]. In addition to this rapid framework development, PINNs have been shown to perform well on a variety of physical problems like Navier–Stokes [32–34], convection heat transfer [35], solid mechanics, [36,37] and the Euler equations [38]. By constraining the network to obey physical laws, PINNs are able to operate in a small data regime and continuously approximate solutions to PDE [39] which allows for model verification across multiple domain resolutions without needing to generate a new approximation each time.

In this work we focus specifically on rate-and-state friction [e.g. 40], an experimentally-motivated, nonlinear friction law capable of reproducing a wide range of observed earthquake behaviors [41] and whose increasing use in dynamic rupture and earthquake cycle simulations has motivated the recent community benchmark exercises described in Harris et al. [42] and Erickson et al. [43]. A better understanding of the depth-dependency of rate-and-state parameters – which have a direct correlation to fault locking and seismic rupture potential – is a fundamental task [3,4]. To address this task we develop a multi-network PINN for modeling a vertical, strike–slip frictional fault embedded in an elastic half-space, and consider deformation in both 1D and 2D. The paper is organized as follows: In Section 2 we first provide an overview of the physics-informed deep learning framework and PINN architecture for general initial–boundary-value problems, in order to best describe the implementation to our application problem. In Section 3 we provide specific details of the PINN framework applied problems, first illustrated in 1D with an example forward problem, then further developed to include inverse problems in 2D. In Section 4 we report details of our optimal network architecture and training methods, verifying our methods with a manufactured solution to ensure accuracy of our inversions. We conclude with a summary and discussion of future work in Section 5.

## 2. Physics-informed deep learning framework

The physics governing motion in many applications in science and engineering give rise to partial differential equations (PDE) where analytic solutions can be difficult to obtain (due, e.g. to complex material properties, boundary conditions, geometry) and we commonly turn to numerical methods. The Physics-Informed Neural Network (PINN) is a deep learning (DL) framework for approximating solutions to PDEs. Though this DL framework lacks the robust mathematical theory of the traditional methods, it shows particular promise in solving problems for which traditional numerical methods are ill-suited [39]. The DL framework produces a closed, analytic form for the solution, which is continuous and defined at every point in the domain allowing one to evaluate the solution "off-grid" without having to resolve the PDE (i.e. it is mesh-free) [39]. In addition, both forward and inverse problems can be solved in the same computational setting, as will be shown in subsequent sections.

### 2.1. Feed-forward deep neural networks

PINNs are extensions of a general feed-forward neural network. We let $\mathbf{x} \in \mathbb{R}^n$ and define a weighting matrix $W \in \mathbb{R}^{m\times n}$ and bias vector $b \in \mathbb{R}^m$. A single hidden layer of a neural network can be expressed as

$$\ell(\mathbf{x};\theta) = \varphi(W\mathbf{x} + b), \quad \text{where } \theta = (W, b), \tag{1}$$

and $\varphi$ is a known (nonlinear) activation function. Deep neural networks are obtained by repeated composition of hidden layers [44].

We let positive integer $L$ be the deep neural network depth (i.e. number of hidden layers) and let $\{\varphi_i\}_{i=1}^{L}$ be a collection of activation functions along with a sequence of trainable network parameters $\{\theta_i\}_{i=0}^{L}$ where $\theta_k = (W_k, b_k)$ for each $0 \le k \le L$. Here we assume each layer consists of $m$ neurons but this simplifying assumption may be omitted so long as one ensures that each weight and bias are of the correct dimension for matrix multiplication and addition, respectively. If $\mathbf{x} \in \mathbb{R}^n$ is the network input and network parameters $(W_0, b_0) \in \mathbb{R}^{m\times n} \times \mathbb{R}^m$, $(W_k, b_k) \in \mathbb{R}^{m\times m} \times \mathbb{R}^m$ for $1 \le k < L$, and $(W_L, b_L) \in \mathbb{R}^{d\times m} \times \mathbb{R}^d$, then the recursive composition

$$\ell_0 = \mathbf{x}, \tag{2a}$$

$$\ell_k = \varphi_k(W_k\ell_{k-1} + b_k) \quad \text{for } 0 < k < L, \tag{2b}$$

defines a feed-forward, deep neural network $\mathcal{N} : \mathbb{R}^n \to \mathbb{R}^d$ (of depth $L$ and width $m$) with network parameters $\theta$ by $\mathcal{N}(\mathbf{x};\theta) = W_L\ell_{L-1} + b_L$.

### 2.2. PINN architecture

The PINN architecture is defined by extending the feed-forward deep neural network to enforce physical conditions set by an initial–boundary value problem (IBVP). For simplicity in the notation (as well as generality) we define this extension for a general IBVP, with specific details for our application given in the subsequent section. Consider the general operator form of an IBVP given by

$$\mathcal{L}[\mathbf{u};\lambda] = \mathbf{k}, \quad \text{in } \widehat{\Omega}, \tag{3a}$$

$$\mathcal{B}[\mathbf{u};\lambda] = \mathbf{g}, \quad \text{on } \partial\widehat{\Omega}, \tag{3b}$$

where $\widehat{\Omega} \subseteq \mathbb{R}^n$, with boundary (including internal interfaces, like faults) $\partial\widehat{\Omega}$. Vector $\mathbf{u}$ is the unknown solution and $\mathbf{g}$ is boundary data. The source term $\mathbf{k}$ encompasses external and internal body forces, and $\mathcal{L}$, $\mathcal{B}$ are differential and boundary operators parameterized by $\lambda$.

The PINN first approximates the solution to the IBVP (3) using a feed-forward, deep neural network, i.e. we assume $\mathbf{u}(\mathbf{x}) \approx \mathcal{N}(\mathbf{x};\theta)$ and consider $N_\partial$ many boundary points $\{\mathbf{x}_\partial^i, \mathbf{g}^i\}_{i=1}^{N_\partial}$ where $\mathbf{g}^i = \mathbf{g}(\mathbf{x}_\partial^i)$ for each $1 \le i \le N_\partial$, and a set of $N_{\widehat{\Omega}}$ internal collocation points $\{\mathbf{x}_{\widehat{\Omega}}^i, \mathbf{k}^i\}_{i=1}^{N_{\widehat{\Omega}}}$ with $\mathbf{k}^i = \mathbf{k}(\mathbf{x}_{\widehat{\Omega}}^i)$ for $1 \le i \le N_{\widehat{\Omega}}$. Throughout this work, collocation points are generated from uniform random sampling. From (3) we construct an objective function $MSE$ (based on the mean-square error) given by

$$MSE(\theta) = MSE_{\widehat{\Omega}}(\theta) + MSE_\partial(\theta), \tag{4}$$

where

$$MSE_{\widehat{\Omega}}(\theta) = \frac{1}{N_{\widehat{\Omega}}}\sum_{i=1}^{N_{\widehat{\Omega}}}\left|\mathcal{L}[\mathcal{N};\lambda](\mathbf{x}^i_{\widehat{\Omega}};\theta) - \mathbf{k}^i\right|^2, \tag{5a}$$

$$MSE_\partial(\theta) = \frac{1}{N_\partial}\sum_{i=1}^{N_\partial}\left|\mathcal{B}[\mathcal{N};\lambda](\mathbf{x}^i_\partial;\theta) - \mathbf{g}^i\right|^2, \tag{5b}$$

are the contributions to the total loss from the PDE (3a) and boundary conditions (3b), respectively, corresponding to misfit of the network with data. Because our objective function (4) consists of multiple competing loss functions it is referred to as a multi-objective loss. Solving (3) is done by minimizing (4) with respect to the network parameters $\theta$, typically done via an optimization algorithm that approximates the set of solution parameters $\theta^*$ such that $\theta^* = \text{argmin}_\theta\, MSE(\theta)$. Fig. 1 gives a schematic representation of the objective function (4). Note that for the rest of this work with omit the notation specifying MSE dependency on network parameters $\theta$ and assume this is understood.

## 3. Learning problems for earthquakes on rate-and-state faults

In order to best illustrate the PINN computational set-up, we begin by considering a relevant problem in 1D, in order to introduce our methodology in a simplified framework. Here we focus on the forward problem; details of the inverse problem are included in the subsequent section on the 2D application problem, which also includes a schematic diagram motivating the governing equations.

### 3.1. 1D illustration

The solid Earth is governed by momentum balance and a constitutive relation defining the material rheology; in this work we assume elastic material properties. These assumptions give rise to the elastic wave equation, namely

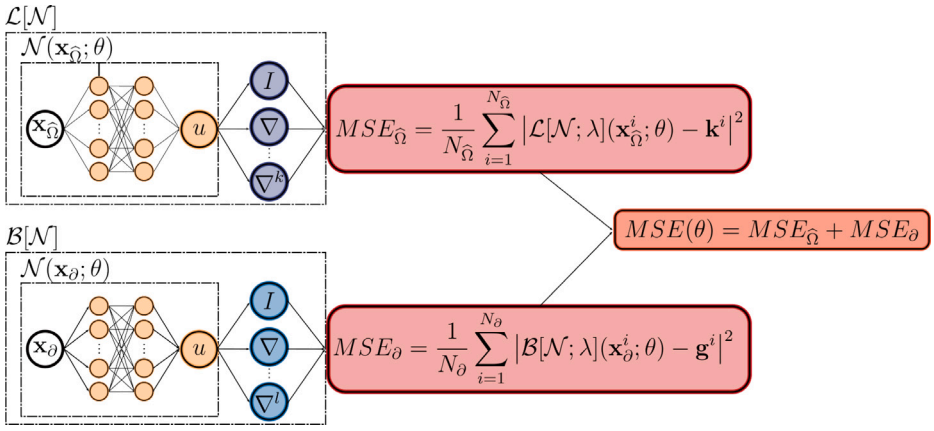


**Fig. 1.** A schematic of the PINN framework for solving the general boundary value problem from Eq. (3). Displacement approximation network $\mathcal{N}$ is trained on interior and boundary subdomains which are governed by operators $\mathcal{L}$ and $\mathcal{B}$, respectively.

$$u_{tt} = c^2 u_{xx} + s(x,t), \quad x \in [0,1], t \geq 0, \tag{6}$$

where $u(x,t)$ is the Earth's material displacement, $c = \sqrt{\mu/\rho}$ is the wavespeed where $\rho$ and $\mu$ are the density and shear modulus, and $s(x,t)$ accounts for external and/or body forces. Here we use subscripts to denote derivatives with respect to the subscript variable. Assumed initial and boundary conditions are given by

$$u(x,0) = u_0(x), \tag{7a}$$
$$u_t(x,0) = v_0(x) \tag{7b}$$
$$-\mu u_x = F + g_0(t), \quad x = 0, \tag{7c}$$
$$Z u_t + \mu u_x = g_1(t), \quad x = 1, \tag{7d}$$

where $Z = \sqrt{\mu\rho}$ is the shear impedance. If boundary data $g_0(t) = g_1(t) = 0$, Eq. (7c) corresponds to the requirement that fault shear stress be equal to frictional strength $F$, and (7d) allows waves to freely exit the domain.

In this work we consider rate-and-state dependent friction (RSF), an experimentally-motivated, nonlinear friction law, often used in earthquake simulations for its ability to reproduce a wide range of observed seismic and aseismic behaviors [41,45,46]. In this context, the frictional strength is given by

$$F = \bar{\sigma}_n f(V,\psi), \tag{8}$$

where $V$ is the slip rate (jump in velocity across a fault interface), $\bar{\sigma}_n$ is the effective normal stress (normal stress minus pore fluid pressure), and $f$ is a friction coefficient given by

$$f(V,\psi) = a \ln\left(\frac{V}{V_0}\right) + \psi, \tag{9}$$

where $V_0$ is a reference slip rate and material parameter $a$ is the "direct effect" [47]. State variable $\psi$ evolves from some initial value $\psi_0$ according to its own evolution law

$$\frac{d\psi}{dt} = G(V,\psi) + h(t), \tag{10a}$$
$$\psi(0) = \psi_0 \tag{10b}$$

where we assume the aging law, i.e. $G(V,\psi) = (bV_0/D_c)\exp(\frac{f_0-\psi}{b} - \frac{|V|}{V_0})$, which allows state to evolve even in the absence of slip [41], and $h(t)$ allows for the incorporation of additional source data. Here $D_c$ is the characteristic slip distance, $f_0$ is a reference friction coefficient for sliding at speed $V_0$ and material parameter $b$ captures time-dependent "evolution" effects [47].

The initial–boundary value problem (IBVP) formed by the PDE (6), specified initial/boundary conditions (7) and state-evolution Eq. (10a) correspond to specific differential and boundary operators $\mathcal{L}$ and $\mathcal{B}$ defined in (3), for example, here $\mathcal{L} = \partial^2/\partial t^2 - c^2\partial^2/\partial x^2$. Assuming all parameters of the IBVP are known, one can solve the forward problem for the unknown material displacement $u(x,t)$ and state variable $\psi(t)$. We define neural networks $\mathcal{N}(x,t;\theta) \approx u(x,t)$ and $\mathcal{N}^{\psi}(t;\theta_{\psi}) \approx \psi(t)$ to approximate the exact solutions. The PDE, boundary and initials conditions all give rise to a term in the objective function. We refer to these terms as the component loss functions and define them as

$$MSE_{\Omega} = \frac{1}{N_{\Omega}}\sum_{i=1}^{N_{\Omega}} \left|\mathcal{N}_{tt} - c^2\mathcal{N}_{xx} - s\right|^2, \tag{11a}$$

$$MSE_0 = \frac{1}{N_0}\sum_{i=1}^{N_0}\left| -\mu\mathcal{N}_x - F - g_0\right|^2, \tag{11b}$$

$$MSE_1 = \frac{1}{N_1}\sum_{i=1}^{N_1}\left| Z\mathcal{N}_t + \mu\mathcal{N}_x - g_1\right|^2, \tag{11c}$$

$$MSE_{u_0} = \frac{1}{N_{u_0}}\sum_{i=1}^{N_{u_0}}\left|\mathcal{N} - u_0\right|^2, \tag{11d}$$

$$MSE_{v_0} = \frac{1}{N_{v_0}}\sum_{i=1}^{N_{v_0}}\left|\mathcal{N}_t - v_0\right|^2, \tag{11e}$$

where $N_\Omega$, $N_0$, $N_1$, $N_{u_0}$, $N_{v_0}$ correspond to the number of randomly distributed collocation points in the relevant subdomains, i.e. the domain interior, left boundary ($x = 0$), right boundary ($x = 1$), and at $t = 0$ (for initial displacements and velocities), respectively. In addition to (11) we have the contributions to the total loss from the state evolution equation, namely

$$MSE_\psi = \frac{1}{N_\psi}\sum_{i=1}^{N_\psi}\left|\mathcal{N}_t^\psi - G(V,\psi) - h\right|^2, \tag{12a}$$

$$MSE_{\psi_0} = \frac{1}{N_{\psi_0}}\sum_{i=1}^{N_{\psi_0}}\left|\mathcal{N}^\psi - \psi_0\right|^2, \tag{12b}$$

so that the objective function we seek to minimize is given by $MSE = \sum_{\xi\in\chi} MSE_\xi$, where subscript $\chi = \{\Omega, 0, 1, u_0, v_0, \psi, \psi_0\}$ so that the sum involves contributions from all subdomains. Minimizing the $MSE$ over the network parameters of $\mathcal{N}$ and $\mathcal{N}^\psi$ define proxy models which produce approximations to $u$ and $\psi$.

An important detail about the network training involved in minimizing the objective function $MSE$ defined in the previous paragraph is that the inclusion of the state evolution means we are now solving a coupled system of PDE. The coupling occurs at the fault $x = 0$, where two conditions (fault friction (7d) and state evolution (10a)) are now enforced. Because of this coupling, the training networks $\mathcal{N}$ and $\mathcal{N}^\psi$ both appear in the fault loss (11b) as well as in the state evolution loss (12a). This interconnectivity may cause training to favor accuracy in the displacement network over accuracy in the state network. In particular, high temporal variations in state evolution, variations in scale, and/or network architecture may drive the training step to favor the displacement approximation. We found it helpful to isolate the state and displacement networks during the backpropagation step. To do this, we define two objective functions: The state objective function consists of loss components in Eq. (12) while the displacement objective function uses loss components from Eq. (11). Each training iteration then requires two optimization steps to update both state and displacement networks. This setup ensures that each network is only updated by one associated objective function. We found this approach to improve our network's training speed and resulted in better approximations of the state variable. The particulars of this training approach are illustrated in Algorithm 1 which provides the pseudocode for solving the 1D IBVP given by Eqs. (6) and (7) where state evolves according to (10a).

In order to verify our computational framework for the 1D problem, we generate a known, manufactured solution, which enables direct comparison with the approximate solution produced by the neural network. This approach, known as the method of manufactured solutions [48], assumes a particular analytic solution $u^e$ and derives consistent source terms and boundary data for the IBVP, without changing the underlying PDE and boundary operators. We manufacture $u^e$ and $\psi^e$ to be a solution to the IBVP and aging law, respectively, such that

$$u^e(x,t) = \tanh\left(0.5(x - ct + 1)\right), \tag{13}$$

$$\psi^e(t) = -\frac{\mu u_x^e(0,t)}{\bar{\sigma}_n} - a\ln\left(\frac{2u_t^e(0,t)}{V_0}\right) \tag{14}$$

This choice of manufactured solution defines the initial data $u_0$ and $v_0$, the source term $s$, and all boundary data, namely

$$s = u_{tt}^e - c^2 u_{xx}^e, \qquad \text{on } \Omega\times[0,T], \tag{15a}$$

$$g_0 = 0, \qquad \text{at } x = 0, \tag{15b}$$

$$g_1 = Zu_t^e + \mu u_x^e, \qquad \text{at } x = 1, \tag{15c}$$

$$h = \psi_t^e - G(2u_t^e, \psi^e), \qquad \text{at } x = 0, \tag{15d}$$

where, for the 1D problem, $V(t) = 2u_t(0,t)$, and corresponds to a commonly chosen exact solution [49,50].

Parameter values used for all studies in this work are given in Table 1. We provide Fig. 2 to illustrate how the independent networks $\mathcal{N}$ and $\mathcal{N}^\psi$ are used to build each component loss function and Algorithm 1 provides an outline of the implemented code.

Here we take $\mathcal{N} : \mathbb{R}^2 \to \mathbb{R}$ and state variable network $\mathcal{N}^\psi : \mathbb{R} \to \mathbb{R}$ to both be fully-connected, feed-forward networks with three hidden layers, 64 neurons per layer and use hyperbolic tangent activation functions. The loss function is optimized using L-BFGS [51], a quasi-Newton optimization algorithm. Minimization is done over 10 training iterations where each iteration is trained on a random sampling of $N = 100$ interior points and 50 boundary points ($N_b = 25$ points per boundary). Network weights are

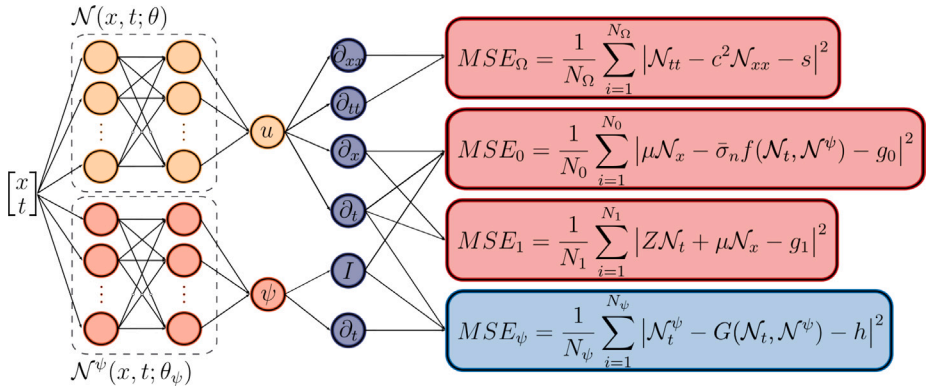


**Fig. 2.** A schematic of the PINN framework for solving the 1D IBVP (6) (7) with rate and state friction defined by (9) and (10a). Displacement network $\mathcal{N}$ and state network $\mathcal{N}^\psi$ are trained separately by defining two objective functions which are differentiated by red and blue nodes, respectively, in the final layer. At each training iteration, $\mathcal{N}$ is updated using component losses $MSE_\Omega, MSE_0$, and $MSE_1$ while $\mathcal{N}^\psi$ is updated using just $MSE_\psi$ as an objective function.

**Table 1**
Parameter values used in the manufactured solution tests.

| Parameter | Value |
|---|---|
| $L_x, L_z$ | 25 km |
| $H$ | 12 km |
| $D$ | 5 km |
| $\mu$ | 32 GPa |
| $\rho$ | 2.67 kg/m$^3$ |
| $\alpha_{\min}$ | −0.005 |
| $\alpha_{\max}$ | 0.015 |
| $f_0$ | 0.6 |
| $\bar{\sigma}_n$ | 50 MPa |
| $V_0$ | $10^{-6}$ m/s |
| $D_c$ | 2 m |

initialized using uniform Xavier initialization [52]. In Fig. 3 we show learned approximations for displacement and state compared to their respective exact solutions. These results suggest that our neural network parameters can solve the 1D forward problem to reasonable accuracy. To further explore performance however, and since we are primarily interested in higher dimensional settings that enable the inversion of depth-dependent parameters, we now move to the 2D formulation.

### 3.2. 2D application in seismic faulting

The 2D problem is obtained by first considering a bounded spatial domain in $\mathbb{R}^3$ defined by $(x, y, z) \in [-L_x, L_x] \times [-L_y, L_y] \times [0, L_z]$ where $z$ is taken to be positive downward. We assume antiplane shear deformation, taking displacements $u_x, u_z$ to be zero, and assume the out-of-plane displacement $u = u_y$ is independent of $y$. Momentum balance for an elastic solid then gives rise to the elastodynamic wave equation in 2D, namely,

$$u_{tt} = c^2 \Delta u + S(x,z,t), \quad (x,z) \in [-L_x, L_x] \times [0, L_z], \ t \geq 0, \tag{16}$$

where $\Delta = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)$ is the 2D Laplacian. Here the shear wavespeed $c = \sqrt{\mu/\rho}$ as in the 1D case, and $S$ comprises body forces. Assumed initial conditions are given by

$$u(x,z,0) = u_0(x,z), \tag{17a}$$

$$u_t(x,z,0) = v_0(x,z). \tag{17b}$$

We assume $z = 0$ corresponds to Earth's surface which we take to be traction free. An RSF vertical fault is embedded at the interface $x = 0$, corresponding to interface conditions

$$\tau^+ = -\tau^- \tag{18a}$$

$$\tau^+ = F(V, \psi). \tag{18b}$$

Here the traction $\tau$ at a boundary or interface is defined with respect to the outward pointing unit normal $\mathbf{n}$ by $\tau = \mathbf{n} \cdot \mu \nabla u$, and $\tau^\pm = \tau(0^\pm, z, t)$. Eq. (18a) corresponds to the requirement that the traction be "equal and opposite" across the fault and (18b) is a

**Algorithm 1** Multi-network training for 1D forward problem with RSF.

$\mathcal{N} \leftarrow$ NeuralNetwork$(x, t; \theta)$ ▷ initialize network parameters $\theta$ for $\mathcal{N}$
$\mathcal{N}^{\Psi} \leftarrow$ NeuralNetwork$(t; \theta_{\psi})$ ▷ initialize network parameters $\theta_{\psi}$ for $\mathcal{N}^{\Psi}$
$\mathcal{N}$opt $\leftarrow$ SetOptimizer$(\theta)$ ▷ Instantiate an optimizer which only tracks parameters of $\mathcal{N}$
$\mathcal{N}^{\Psi}$opt$\leftarrow$ SetOptimizer$(\theta_{\psi})$ ▷ Instantiate an optimizer which only tracks parameters of $\mathcal{N}^{\Psi}$

$\widetilde{\mathcal{N}} \leftarrow u_0(x) + tv_0(x) + t^2\mathcal{N}(x,t)$ ▷ Employ hard enforcement of initial conditions
$\widetilde{\mathcal{N}^{\Psi}} \leftarrow \psi_0 + t\mathcal{N}^{\Psi}(t)$

**for** $\xi$ **in** $\chi$ **do** ▷ Loop over each subdomain and define the relevant component loss
  **function** $\text{MSE}_{\xi}(x,t)$
    output $\leftarrow$ Condition$[\xi](\widetilde{\mathcal{N}}(x,t),\ \widetilde{\mathcal{N}^{\Psi}}(t))$ ▷ Conditions specified by equations (6), (7), (8), and (10a)
    data $\leftarrow$ SourceData$[\xi](x,t)$ ▷ Source terms are determined in equation (15)
    **return** MeanSquareError(output, data)

**for** i=1 **to** training_iterations **do**
  $\text{Loss}_{\mathcal{N}} \leftarrow 0$
  $\text{Loss}_{\mathcal{N}^{\Psi}} \leftarrow 0$
  **for** $\xi$ **in** $\chi \setminus \{\Psi\}$ **do**
    $x_{\xi}, t_{\xi} \leftarrow$rand$(N_{\xi}, \xi)$ ▷ Generate $N_{\xi}$ randomly sampled points from subdomain $\xi$
    $\text{Loss}_{\mathcal{N}} \leftarrow \text{Loss}_{\mathcal{N}} + \text{MSE}_{\xi}(x_{\xi}, t_{\xi})$
  $x_{\psi}, t_{\psi} \leftarrow$rand$(N_{\psi}, \Psi)$
  $\text{Loss}_{\mathcal{N}^{\Psi}} \leftarrow \text{MSE}_{f_{\psi}}(x_{\psi}, t_{\psi})$
  $\nabla_{\psi} \leftarrow \text{grad}(\text{Loss}_{\mathcal{N}^{\Psi}}, [\theta_{\psi}])$ ▷ Compute network gradient
  $\theta_{\psi} \leftarrow \mathcal{N}^{\Psi}$opt.Step$(\nabla_{\psi})$ ▷ Update network weights using the optimization algorithm
  $\nabla_{\mathcal{N}} \leftarrow \text{grad}(\text{Loss}_{\mathcal{N}}, [\theta])$
  $\theta \leftarrow \mathcal{N}$opt.Step$(\nabla_{\mathcal{N}})$

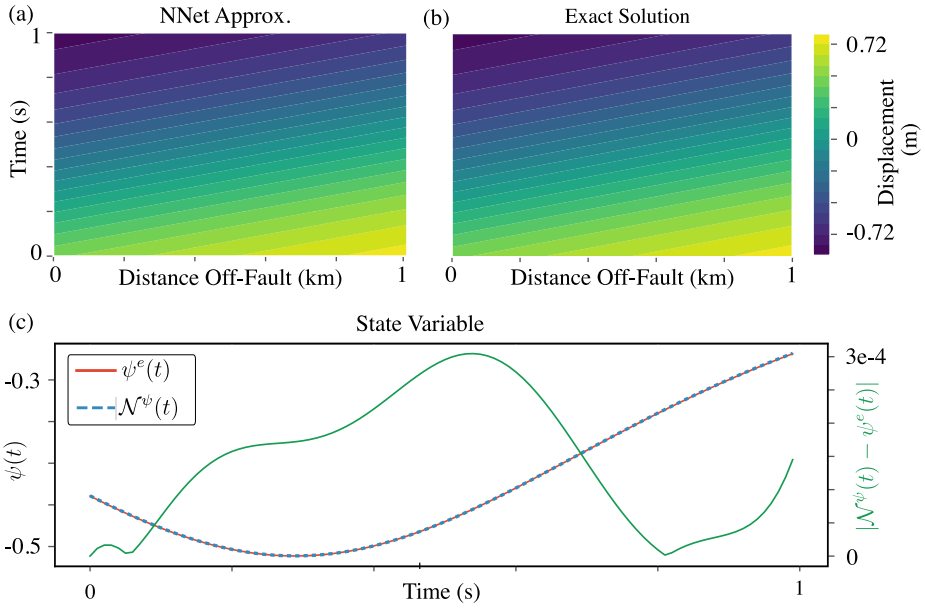


**Fig. 3.** Comparison of results from 1D illustration showing the (a) displacement network approximation $\mathcal{N}$ with (b) manufactured solution $u^e$. Additionally, the (c) state approximation network $\mathcal{N}^{\Psi}$ is plotted against the manufactured state $\psi^e$ along with their absolute error $|\mathcal{N}^{\Psi}(t) - \psi^e(t)|$. Absolute displacement error was averaged over 1000 randomly sampled points and measured to be $|\mathcal{N} - u^e|_{\text{avg}} = 1.57e-5$.

requirement that fault shear stress be equal to frictional strength $F$ as in the 1D case. As in the 1D illustration, $F$ is a nonlinear function of the slip rate $V(z,t) = \dot{u}(0^+, z, t) - \dot{u}(0^-, z, t)$ and an empirical state variable $\psi(z,t)$ the particular form of which is given by (8). At the remote boundaries $(x,z) = (\pm L_x, L_z)$ we assume the boundaries are non-reflecting. Next we make the assumption that the displacement field $u(x,z,t)$ is antisymmetric about the fault interface $x = 0$, which automatically satisfies (18a), so that we may restrict our focus to one side of the fault as in Erickson and Dunham [49] and reduce the computational redundancy. Our final spatial domain is then $\Omega = [0, L_x] \times [0, L_z]$ where the fault interface now becomes a boundary, see Fig. 4. For generality (and to

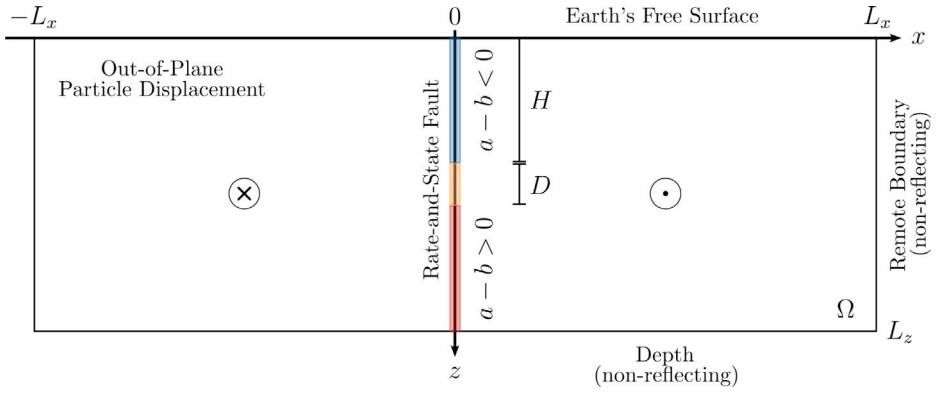


**Fig. 4.** A schematic of the 2D domain where $u(x,z,t)$ is the particle displacement in the $y$-direction. Out-of-plane motion is denoted with circles. The assumption that $u$ is antisymmetric about $x=0$ allows us to consider the one-sided domain $\Omega=[0,L_x]\times[0,L_z]$ to alleviate computational cost. Displacements within $\Omega$ are governed by the 2D wave equation (16). The fault, Earth's free surface, and the remaining boundaries (the remote and depth boundaries) are labeled and subject to conditions (19). At the rate-and-state fault ($x=0$), the fault is velocity weakening ($a-b<0$) down to a seismogenic depth $H$, where it transitions over distance $D$ to velocity strengthening $a-b>0$.

aid in later verification steps) we state the final boundary conditions on $\partial\Omega$ using generic data $g_f, g_s, g_r, g_d$ for the fault, surface, remote, and depth boundaries, respectively, namely

$$\tau = F(V) + g_f(z,t), \quad x=0, \tag{19a}$$

$$\tau = g_s(x,t), \quad z=0, \tag{19b}$$

$$Zu_t + \tau = g_r(z,t), \quad x=L_x, \tag{19c}$$

$$Zu_t + \tau = g_d(z,t), \quad z=L_z, \tag{19d}$$

although the Earth's free surface and non-reflecting conditions are the far field boundaries correspond to $g_s = g_r = g_d = 0$. Here $\tau=\tau^+$, $V=2u_t^+$ and $Z=\sqrt{\mu\rho}$ is the shear impedance as in the 1D case.

For this 2D application problem we are primarily interested in the RSF parameter $a-b$, which describes the velocity-dependence of friction at steady state. Positive values (i.e. $a>b$) correspond to stable sliding, while negative values correspond to frictional instabilities. Knowledge of the depth-distribution of $a-b$ is therefore of central importance to understanding slip behavior and deformation [3]. In other words, heterogeneities along fault interfaces can be characterized at least in part by velocity-weakening frictional behavior ($a-b<0$) indicating that seismic rupture may nucleate and easily propagate, while stable regions are characterized by velocity-strengthening frictional behavior ($a-b>0$) that inhibits the sliding at seismogenic speeds [40]. In order to focus our study on the inference of the depth-dependency of $a-b$, we assume that the state variable is at steady state, namely $\psi = f_0 + b\ln\left(\frac{V_0}{V}\right)$, so that the friction coefficient reduces to the rate-dependent form

$$f(V) = f_0 + a\ln\left(\frac{V}{V_0}\right), \tag{20}$$

where we have introduced $a=a-b$. This choice of rate-dependent friction is made to simplify the optimization problem by limiting the number of trainable networks from three to two for computational ease. We assume that $a$ is piecewise linear with depth, defining a shallow seismogenic zone, namely,

$$a(z) = \begin{cases} a_{\min} & 0<z<H \\ (z-H)*((a_{\max}-a_{\min})/D)+a_{\min} & H\le z\le H+D \\ a_{\max} & H+D<z, \end{cases} \tag{21}$$

where $H$ defines the seismogenic depth, $a_{\min}$ and $a_{\max}$ are constants defining the minimum and maximum values assumed by $a$, and $D$ the transition distance. In this work we consider PINN solutions to both the forward and inverse problems. For forward problem we solve for the unknown displacements $u$ in the IBVP (3) assuming $a(z)$ has been specified. For the inverse problems, in addition to solving for $u$ we also infer the friction parameter $a$. In this latter case, (21) sets the data for the boundary conditions, but $a$ is not assumed to be known a priori when enforcing the frictional interface condition, as will be described in the next section.

### 3.3. Forward and inverse problems for the 2D application

Governing Eq. (16) along with initial and boundary conditions (17), (19) provide specifics of the loss terms (5) that define the PINN. However, there is more than one way to formulate the associated learning problem. First, one can consider either a forward

or inverse problem [39]. As this work is concerned with generating approximations to both forward and inverse problems, we will refer to the *primal* solution as the network approximation $\mathcal{N}$ to the displacement $u$ specified by the IBVP (3). The *forward* problem is only concerned with approximating a primal solution and requires that all model parameters $\lambda$ are known a priori. A solution to the *inverse* problem aims to approximate the primal solution while also being tasked with learning a set of system parameters. If the $\lambda$ in (3) are known, the forward problem is solved and the problem reduces to an unsupervised learning task [17]. In the inverse problem, however, the PINN has the same loss function (5) but with minor changes: Instead of knowing the system parameters $\lambda$, we establish them as trainable networks, which we detail shortly.

In addition to specifying whether a forward or inverse problem is being solved, in either case one must also choose between soft or hard enforcement of initial conditions [12,17,53,54]. Soft enforcement uses loss terms to learn boundary data whereas hard enforcement encodes boundary data into a trial function to satisfy the conditions exactly. We proceed with details concerning each of these below.

### 3.3.1. Forward problem

As in the 1D case, we begin by supposing that $\mathcal{N}(x,z,t;\theta) \approx u(x,z,t)$ for neural network $\mathcal{N}$ and solution $u$ of the IBVP (16), (17), (19). The 2D component loss functions are therefore given by

$$MSE_\Omega = \frac{1}{N_\Omega}\sum_{i=1}^{N_\Omega}\left|\mathcal{N}_{tt} - c^2\Delta\mathcal{N} - S\right|^2, \tag{22a}$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left| - \mu\mathcal{N}_x - \bar{\sigma}_n f - g_f\right|^2, \tag{22b}$$

$$MSE_s = \frac{1}{N_s}\sum_{i=1}^{N_s}\left| - \mu\mathcal{N}_z - g_s\right|^2, \tag{22c}$$

$$MSE_r = \frac{1}{N_r}\sum_{i=1}^{N_r}\left|Z\mathcal{N}_t + \mu\mathcal{N}_x - g_r\right|^2, \tag{22d}$$

$$MSE_d = \frac{1}{N_d}\sum_{i=1}^{N_d}\left|Z\mathcal{N}_t + \mu\mathcal{N}_z - g_d\right|^2 \tag{22e}$$

along with initial loss terms $MSE_{u_0}, MSE_{v_0}$ (translated to 2D) from Eqs. (11d) and (11e), respectively. As in the 1D case, summations are over randomly distributed collocation points in the interior and domain boundaries. The 2D objective function we seek to minimize is given by $MSE = \sum_{\xi\in\chi} MSE_\xi$, where (as in the 1D case), subscript $\chi = \{\Omega, f, s, r, d, u_0, v_0\}$ implies that the sum is taken over all relevant subdomains.

### 3.3.2. Inverse problem

While the primal solution to an IBVP is desirable, replacing an assumed model parameter (which often comes with much uncertainty) with a trainable network could leverage real-world data to physically constrain parameter distributions. The inverse problem is derived by making a slight modification to the forward problem (22). As we are interested in learning the fault friction parameter $\alpha$, we modify the loss component (22b) to include a trainable network. Because $\alpha$ is depth-variable it must be explicitly trained on random collocation points along the fault. So in addition to the first network $\mathcal{N}(x,z,t;\theta) \approx u(x,z,t)$, we introduce a secondary network $\mathcal{N}^\alpha(z;\theta_\alpha) \approx \alpha(z)$. Only one loss component needs be changed from the forward problem (22), namely, component (22b), which enforces the fault friction condition. A modified friction coefficient is considered, defined by

$$\tilde{f} = f_0 + \mathcal{N}^\alpha(z)\ln\left[\frac{2\mathcal{N}_t(z,t)}{V_0}\right], \tag{23}$$

which gives rise to the modified fault loss

$$\overline{MSE}_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left| - \mu\mathcal{N}_x - \bar{\sigma}_n\tilde{f} - g_f\right|^2. \tag{24}$$

Replacing loss component (22b) in the forward problem with (24) defines the inverse problem. We give a diagram of the fault network for the inverse problem in Fig. 5 showing how the fault loss is used to update two networks during training. The inverse problem is trained over the same subdomains as the forward problem (22) but the depth coordinate for fault training data is passed to both $\mathcal{N}$ and $\mathcal{N}^\alpha$.

### 3.3.3. Soft vs. Hard enforcement of boundary conditions

Both the forward and inverse problems require that we specify initial and boundary conditions, which may be enforced in two possible ways. *Soft* enforcement is done by penalizing the objective function, as the network output need not satisfy the condition exactly (as was done in the 1D example of the previous section). Up until now, we have presented soft enforcement, as given in (22), which shows component losses corresponding to the boundary and initial conditions. Soft enforcement does not give us any guarantee regarding accuracy of the condition being enforced and each additional loss term in the objective function increases the complexity of the optimization landscape.

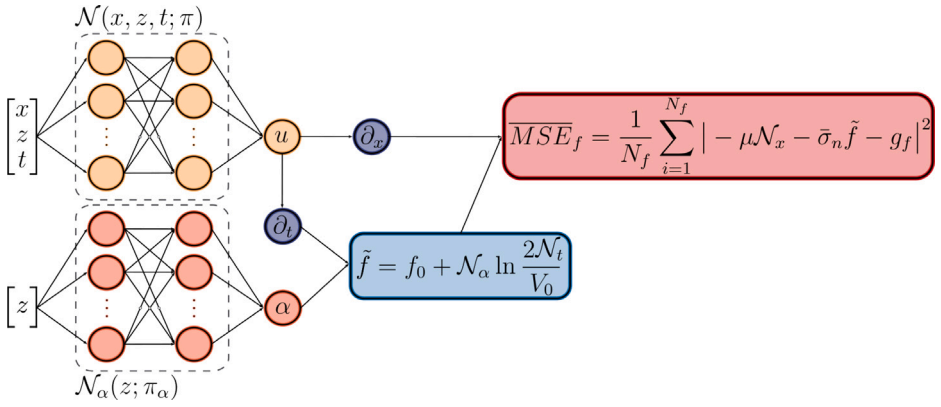


**Fig. 5.** Network diagram for enforcing the stress condition along the fault while also learning the depth-dependent friction parameter $\alpha$. During training, the $z$-coordinate of a training point is passed to both $\mathcal{N}$ and $\mathcal{N}^\alpha$.

Alternatively, we may enforce initial/boundary conditions in a manner that reduces the complexity of the network training space by encoding such conditions into the network architecture. This approach, known as *hard* enforcement, requires that we specify a trial function which automatically satisfies the initial/boundary conditions. Here we apply this technique to enforce initial conditions, which reduces the number of constraints on the system. Let $\mathcal{N}(\mathbf{x},t;\theta)$ be a feed-forward neural network and define the trial function $\widetilde{\mathcal{N}}$ to be

$$\widetilde{\mathcal{N}}(\mathbf{x},t;\theta) = u_0(\mathbf{x}) + tv_0(\mathbf{x}) + t^2\mathcal{N}(\mathbf{x},t;\theta) \tag{25}$$

where $\mathbf{x} = (x,z)$, so that $\widetilde{\mathcal{N}}$ satisfies both initial conditions (17) exactly and is trained by training the network $\mathcal{N}$. It is worth noting that $\widetilde{\mathcal{N}}$ represents an approximation to displacement and consists of a static component (first two terms) that enforce the initial conditions exactly, and a trainable component $\mathcal{N}$. The final term in Eq. (25) is quadratic in time to ensure that the trainable component of $\widetilde{\mathcal{N}}$ (and $\widetilde{\mathcal{N}}_t$) vanishes at $t = t_0$. If $\widetilde{\mathcal{N}}$ is used in place of $\mathcal{N}$ in the formulation of the objective function, there is no longer a need to include loss terms for initial displacements (11d), or initial velocities (11e) so the network can be trained on a less restrictive set of conditions.

## 4. 2D verification, validation and applications

When computational methods for physical problems are used to address science questions, verification is an essential first step to ensure credible results [42,43]. While validation with observational data is the focus of future work, we must first verify that our physics-informed deep learning framework is able to solve both forward and inverse problems to reasonable accuracy.

### 4.1. Verification with the method of manufactured solutions

As in the 1D case, we verify the PINN framework by generating a known, manufactured solution $u^e$ which solves the IBVP (3), (17), (19). We take

$$u^e(x,z,t) = \tanh\left((x+z+ct)/20\right), \tag{26}$$

which defines the initial data $u_0$ and $v_0$, the source term $S$, and all boundary data, which for the forward problem is

$$S = u^e_{tt} - c^2\Delta u^e, \qquad \text{on } \Omega\times[0,T], \tag{27a}$$
$$g_f = -\mu u^e_x - \bar{\sigma}_n f\left(2u^e_t\right), \qquad \text{at } x = 0, \tag{27b}$$
$$g_s = -\mu u^e_z, \qquad \text{at } z = 0, \tag{27c}$$
$$g_r = Zu^e_t + \mu u^e_x, \qquad \text{at } x = L_x, \tag{27d}$$
$$g_d = Zu^e_t + \mu u^e_z, \qquad \text{at } z = L_z. \tag{27e}$$

The parameters used are given in Table 1 whose values align with other recent works with similar model set-ups [50,55], and we default to this parameter set unless stated otherwise. Note that in the case of the inverse problem, $\alpha_{\min}$ and $\alpha_{\max}$ define the manufactured exact solution $\alpha^e(z)$, which is used only to set the data in (27b); $\alpha(z)$ is a learned parameter through the use of the network $\mathcal{N}^\alpha$. In all scenarios we consider, we take the primal network $\mathcal{N} : \mathbb{R}^3 \to \mathbb{R}$ and friction network $\mathcal{N}^\alpha : \mathbb{R} \to \mathbb{R}$ to both be fully-connected, feed-forward networks with three hidden layers and 128 neurons per layer. The wave Eq. (16) requires displacements to be sufficiently smooth with respect to the input but no such requirement is imposed on the friction parameters.

**Algorithm 2** Multi-network training for solving the inverse problem

```
1: 𝒩 ← NeuralNetwork(x, z, t; θ)                          ▷ θ initial network parameters for 𝒩
2: 𝒩^α ← NeuralNetwork(z; θ_α)                            ▷ θ_α initial network parameters for 𝒩^α
3: θ ← [θ, θ_α]
4:
5: 𝒩̃ ← u_0(x, z) + t v_0(x, z) + t^2 𝒩(x, z, t)          ▷ Employ hard enforcement of initial conditions
6:
7: for ξ in χ do                                          ▷ Loop over each subdomain and define the relevant component loss
8:     function MSE_ξ(x, z, t)
9:         output ← Condition[ξ](𝒩̃(x, z, t), 𝒩^α(z))
10:        data ← SourceData[ξ](x, z, t)
11:        return MeanSquareError(output, data)
12:
13: for i=1 to training_iterations do
14:     Loss ← 0
15:
16:     for ξ in χ do
17:         x_ξ ←rand(N_ξ, ξ)                             ▷ Generate N_ξ randomly sampled points from subdomain ξ
18:         Loss ← Loss + MSE_ξ(x_ξ)
19:
20:     ∇ ← grad(Loss, [θ])
21:     θ ← OptimizerStep(∇)                              ▷ Update network weights using the optimization algorithm
```

Thus we use hyperbolic tangent activation functions for the displacement network but use linear units in the friction network. We found that the friction network performed well when we used both Rectified Linear Unit (ReLU) and Sigmoid Linear Unit (SiLU) together. We use ReLU activation for the outer hidden layers and a SiLU activation on the interior layer. Algorithm 2 is a sketch of the implemented code used to solve the inverse problem with hard enforcement of initial conditions.

$\mathcal{N}$ and (in the case of the inverse problem) $\mathcal{N}^\alpha$ are trained by minimizing the MSE (the sum of component MSE given in (22)) where all loss functions are optimized using L-BFGS [51], a quasi-Newton optimization algorithm. Minimization is done over 30 training iterations where each iteration is trained on a mini-batch of $N = 400$ interior points and 400 boundary points ($N_b = 100$ points per boundary). Additionally, if soft enforcement of initial conditions is used, we sample 400 additional interior points (200 for displacements and 200 for velocities). After the 30 training iterations are complete we evaluate each of the component loss functions using 1000 randomly sampled points. Likewise we generate the same number of sample points for $\mathcal{N}$ and $\mathcal{N}^\alpha$ and use their respective manufactured solutions to compute the relative $\ell^2-$norm of their errors, namely,

$$\left\|\mathcal{N} - u^e\right\|_{1,\,\text{rel}}^2 = \frac{\sum_{i=1}^{N} \left|\mathcal{N}(x_i, z_i, t_i) - u^e(x_i, z_i, t_i)\right|^2}{\sum_{i=1}^{N} \left|u^e(x_i, z_i, t_i)\right|^2}, \tag{28}$$

$$\left\|\mathcal{N}_\alpha - \alpha^e\right\|_{1,\,\text{rel}}^2 = \frac{\sum_{i=1}^{N_b} \left|\mathcal{N}^\alpha(z_i) - \alpha^e(z_i)\right|^2}{\sum_{i=1}^{N_b} \left|\alpha^e(z_i)\right|^2}. \tag{29}$$

All collocation points are drawn uniformly from the space–time domain and are resampled at each training iteration. This choice of sampling method, in addition to being simple to implement, has been shown to perform well compared to fixed residual point methods (e.g. uniformly-spaced grids, uniform random points, or Latin hypercube sampling for which collocation points are not resampled) [56].

Network weights are initialized using uniform Xavier initialization [52]. Variation due to weight initialization involving randomization is accounted for by averaging error data across 50 trained solutions. Table 2 presents error data for the forward and inverse problems where both hard and soft enforcement of initial conditions are considered for each. As the table illustrates, for both the forward and inverse problems, hard enforcement of initial conditions are accompanied with a smaller $\ell^2$-errors after 30 training iterations, which is most likely due at least in part to the fact that hard enforcement means that the network approximation satisfies the initial conditions exactly. Also illustrated in the table, is that all component mean-square errors (MSE) are lower for the soft enforcement of initial conditions compared to hard enforcement. Soft enforcement uses two additional loss components (i.e. those enforcing initial conditions) which results in a more biased model, or a lower variance model (i.e. the MSE is less sensitive to training data). The trade off between bias and variance (introduced via randomness in weight initialization and collocation points) may account for such lower MSE when using soft enforcement (which involves lower variances) and averaging over 50 trained models.

Next we test network performance across various subdomains and resolutions. Let $\mathcal{N}$ be a displacement network trained to solve the inverse problem using hard enforcement of initial conditions. This time, errors are measured with respect to the continuous $L^2$-norm

$$\left\|\mathcal{N} - u^e\right\|_{2,\Lambda}^2 = \int_\Lambda \left|\mathcal{N}(x, z, t) - u^e(x, z, t)\right|^2 d\lambda, \tag{30}$$

and we consider error accumulation across $\Omega$, $[0, T]$ and $\hat{\Omega} = \Omega \times [0, T]$, the spatial, temporal and space–time domains, respectively. The integral in (30) is approximated using Simpson's composite quadrature rule which converges to the exact integral with rate

**Table 2**
Errors in the $\ell^2$-norm and MSE (loss) associated with hard and soft enforcement of initial conditions for both the forward and inverse problem. Each term is computed over 1000 randomly sampled points and averaged across 50 trained solutions. Error ratios are computed by dividing soft enforcement error by the hard enforcement error.

| Errors | Forward problem | | | Inverse problem | | |
|---|---|---|---|---|---|---|
| | Soft | Hard | Error ratio | Soft | Hard | Error ratio |
| $\|\mathcal{N} - u^e\|_{1,\ \text{rel}}$ | 1.364e−02 | 7.148e−03 | 1.908e+00 | 2.931e−02 | 1.633e−02 | 1.795e+00 |
| $\|\mathcal{N}_\alpha - \alpha^e\|_{1,\ \text{rel}}$ | | | | 2.977e−01 | 7.220e−02 | 4.123e+00 |
| $MSE_{PDE}$ | 1.149e−03 | 8.883e−03 | 1.293e−01 | 1.133e−03 | 6.675e−03 | 1.697e−01 |
| $MSE_f$ | 2.633e−03 | 4.115e−02 | 6.399e−02 | 8.579e−03 | 1.615e−01 | 5.311e−02 |
| $MSE_s$ | 8.393e−05 | 7.868e−04 | 1.067e−01 | 1.251e−04 | 6.545e−04 | 1.911e−01 |
| $MSE_d$ | 8.216e−05 | 2.115e−03 | 3.885e−02 | 4.681e−04 | 7.199e−03 | 6.502e−02 |
| $MSE_r$ | 1.195e−05 | 3.233e−04 | 3.698e−02 | 1.064e−04 | 8.074e−04 | 1.318e−01 |
| $MSE_{u_0}$ | 1.703e−04 | | | 3.722e−04 | | |
| $MSE_{v_0}$ | 6.308e−04 | | | 1.375e−03 | | |

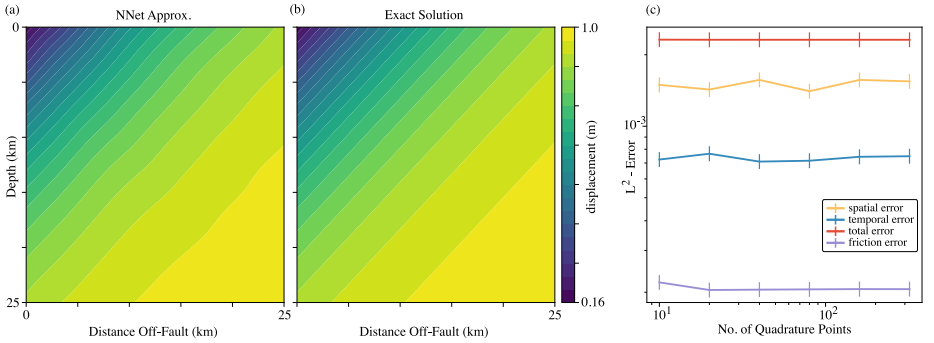


**Fig. 6.** (a) 2D displacement plot for a PINN trained to solve the inverse problem using hard enforcement of initial conditions compared to (b) the manufactured displacements. (c)$L^2$-errors for displacement in space, time, and spacetime (along with $L^2$-errors for the friction parameter) are computed on a uniform grid using Simpson's rule as a quadrature. Errors are then recorded over several mesh refinements.

$\mathcal{O}(k^4)$, where $k$ is the quadrature grid size (i.e. the subinterval length). To account for variations in the trained network, we average errors over points outside of the domain of integration. For example, the average temporal error given by

$$\overline{\|\mathcal{N} - u^e\|}_{2,[0,T]} = \frac{1}{N_T}\sum_{k=1}^{N_t}\left[\int_{[0,T]} \left|\mathcal{N}(x_k, z_k, t) - u^e(x_k, z_k, t)\right|^2 d\lambda(t)\right]^{1/2} \tag{31}$$

measures expected error accumulation (over time) given $N_T$ randomly sampled spatial points.

Fig. 6 shows displacement plots for the trained network and the exact solution at final time $T = 1$ in addition to network $L^2$-error approximations for displacement in space, time and space–time (along with an $L^2$-error approximation of the friction parameter) for increasingly higher grid resolution used in the numerical quadrature. The errors remain relatively constant with decreasing subinterval size, suggesting that the quadrature approximation is approaching the actual $L^2$-error, and that the PINN approximation maintains good accuracy even when evaluated on higher-resolution grids. Fig. 7(a) shows network component loss functions against training iteration, revealing a non-monotonic decrease across all components. This behavior is likely due to how data is batched during training with smaller batches tending towards smaller, less accurate loss updates while larger batches tend towards fewer, more accurate loss updates [57,58]. Fig. 7(b) illustrates convergence of the inferred parameter $\alpha = a - b$ to the exact distribution $\alpha^e$. Here we show iterations 1 and 30 merely because after the first iteration the inferred parameter performs well and does not vary significantly.

## 5. Summary and future work

We have presented a computational framework for physics-informed neural networks (PINNs) for solving the elastodynamic wave equation with a rate-and-state frictional fault boundary in both 1D and 2D. We consider both forward and inverse problems, with the latter obtained by extending to a multi-network architecture in order to learn depth-dependent friction parameters alongside deformations in the Earth's crust. We verified the computational framework by applying the method of manufactured solutions to probe various error measurements. We show that in general, hard enforcement of boundary conditions result in trained networks that better approximate displacements and friction parameters but tend to be worse at minimizing component loss functions when compared to soft enforcement. We found that a PINN defined by hard enforcement of initial conditions produces reasonable approximations for displacements and the desired friction parameter distribution. Though the network is mesh-free, the $L^2$-error yields near constant values across various quadrature grids, suggesting that the PINN provides a reasonable approximation to the

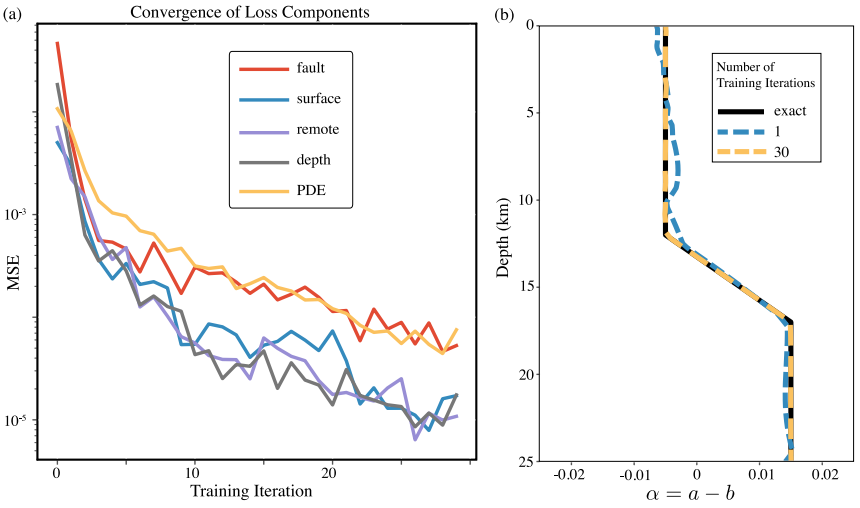


**Fig. 7.** 2D inversion results showing (a) convergence of loss components and (b) convergence of the inferred parameter approximation.

solution even when evaluated on increasingly finer grids. Moreover, although the network requires sufficient training iterations to properly learn displacements, the desired friction parameter is learned within the first couple of iterations. This suggests that PINNs may be a highly effective tool in inferring subsurface friction properties along faults, constrained by both physics and observational surface data.

While the PINN is shown to perform well when learning the state variable in 1D, and inferring depth-dependency of RSF parameter $a - b$ at steady-state in 2D, we plan to explore the capabilities of the 2D framework with non-steady-state-evolution. This extension requires additional networks in the 2D setting in order to approximate the state variable, and two inference networks to capture the empirical parameters $a$ and $b$ (which become separated across governing equations), and/or networks to learn other frictional parameters, such as $D_c$, whose scaling from laboratory values to actual fault zones is the subject of many studies [e.g.59]. Approaching such a problem may be aided by a better understanding of the PINN dependence on problem configuration as well as network architecture. Additionally, it would be worthwhile to investigate methods which hybridize PINNs with traditional numerical methods similar to the discrete time PINN in Raissi et al. [39]. And finally, with a PINN solution that can handle learning full rate-and-state fault friction in 2D (and eventually 3D), we would be ready to compare model outcomes against community benchmark problems concerning dynamic rupture simulations [42] and sequences of earthquakes and aseismic slip [43].

## CRediT authorship contribution statement

**Cody Rucker:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Brittany A. Erickson:** Writing – review & editing, Supervision, Methodology, Formal analysis, Conceptualization.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

All code for building and training models is open-source, written in Python using the machine learning library Pytorch, and is available at https://github.com/Thrase/EQ_pinns.

## Acknowledgments

This work benefited from helpful feedback from two anonymous reviewers. The authors were supported by National Science Foundation Award Nos. 2313659 and 2339996.

## References


[1] S. Ide, G.C. Beroza, D.R. Shelly, T. Uchide, A scaling law for slow earthquakes, Nature 447 (7140) (2007) 76–79.

[2] B.F. Atwater, M.-R. Satoko, S. Kenji, T. Yoshinobu, U. Kazue, D.K. Yamaguchi, The Orphan Tsunami of 1700: Japanese Clues to a Parent Earthquake in North America, second ed., University of Washington Press, 2005, URL http://www.jstor.org/stable/j.ctvcwnbrv.

[3] E.E. Brodsky, J.J. Mori, L. Anderson, F.M. Chester, M. Conin, E.M. Dunham, N. Eguchi, P.M. Fulton, R. Hino, T. Hirose, M.J. Ikari, T. Ishikawa, T. Jeppson, Y. Kano, J. Kirkpatrick, S. Kodaira, W. Lin, Y. Nakamura, H.S. Rabinowitz, C. Regalla, F. Remitti, C. Rowe, D.M. Saffer, S. Saito, J. Sample, Y. Sanada, H.M. Savage, T. Sun, S. Toczko, K. Ujiie, M. Wolfson-Schwehr, T. Yang, The state of stress on the fault before, during, and after a major earthquake, Annu. Rev. Earth Planet. Sci. 48 (1) (2020) 49–74, http://dx.doi.org/10.1146/annurev-earth-053018-060507.

[4] National Academies of Sciences, Engineering, and Medicine, et al., A Vision for NSF Earth Sciences 2020–2030: Earth in Time, National Academies Press, Washington, DC, 2020.

[5] K.J. Bergen, P.A. Johnson, M.V. de Hoop, G.C. Beroza, Machine learning for data-driven discovery in solid earth geoscience, Science 363 (6433) (2019) eaau0323, http://dx.doi.org/10.1126/science.aau0323.

[6] M. Kern, Numerical Methods for Inverse Problems, John Wiley & Sons, 2016.

[7] D. Givoli, A tutorial on the adjoint method for inverse problems, Comput. Methods Appl. Mech. Engrg. 380 (2021) 113810, http://dx.doi.org/10.1016/j.cma.2021.113810.

[8] Q. Kong, D.T. Trugman, Z.E. Ross, M.J. Bianco, B.J. Meade, P. Gerstoft, Machine Learning in Seismology: Turning Data into Insights, Seismol. Res. Lett. 90 (1) (2018) 3–14, http://dx.doi.org/10.1785/0220180259.

[9] J.T. Lin, D. Melgar, A.M. Thomas, J. Searcy, Early warning for great earthquakes from characterization of crustal deformation patterns with deep learning, J. Geophys. Res.: Solid Earth 126 (10) (2021) e2021JB022703.

[10] H. Kubo, M. Naoi, M. Kano, Recent advances in earthquake seismology using machine learning, Earth, Planets Space 76 (1) (2024) 36.

[11] J. Zhao, H. Ling, J. Liu, J. Wang, A.F. Burke, Y. Lian, Machine learning for predicting battery capacity for electric vehicles, eTransportation 15 (2023) 100214, http://dx.doi.org/10.1016/j.etran.2022.100214.

[12] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440, http://dx.doi.org/10.1038/s42254-021-00314-5.

[13] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.

[14] R. Fukushima, M. Kano, K. Hirahara, Physics-informed neural networks for fault slip monitoring: simulation, frictional parameter estimation, and prediction on slow slip events in a spring-slider system, ESS Open Arch. (2023) http://dx.doi.org/10.22541/essoar.168988460.01601423/v1.

[15] T. Okazaki, T. Ito, K. Hirahara, N. Ueda, Physics-informed deep learning approach for modeling crustal deformation, Nature Commun. 13 (1) (2022) 7092, http://dx.doi.org/10.1038/s41467-022-34922-1.

[16] S. Karimpouli, P. Tahmasebi, Physics informed machine learning: Seismic wave equation, Geosci. Front. 11 (6) (2020) 1993–2001, http://dx.doi.org/10.1016/j.gsf.2020.07.007.

[17] S. Cuomo, V.S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics–informed neural networks: Where we are and what's next, J. Sci. Comput. 92 (3) (2022) 88, http://dx.doi.org/10.1007/s10915-022-01939-z.

[18] B. Yu, et al., The deep ritz method: A deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12, http://dx.doi.org/10.1007/s40304-018-0127-z.

[19] E. Kharazmi, Z. Zhang, G.E. Karniadakis, Variational physics-informed neural networks for solving partial differential equations, 2019, http://dx.doi.org/10.48550/arXiv.1912.00873, arXiv preprint arXiv:1912.00873.

[20] E. Kharazmi, Z. Zhang, G.E. Karniadakis, hp-VPINNs: Variational physics-informed neural networks with domain decomposition, Comput. Methods Appl. Mech. Engrg. 374 (2021) 113547, http://dx.doi.org/10.1016/j.cma.2020.113547.

[21] A.D. Jagtap, E. Kharazmi, G.E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, Comput. Methods Appl. Mech. Engrg. 365 (2020) 113028, http://dx.doi.org/10.1016/j.cma.2020.113028.

[22] P.G. Ciarlet, The finite element method for elliptic problems, vol. 40, Siam, 2002.

[23] A. Ern, J.-L. Guermond, Theory and Practice of Finite Elements, vol. 159, Springer, 2004, http://dx.doi.org/10.1007/978-1-4757-4355-5.

[24] G. Kutyniok, The mathematics of artificial intelligence, 2022, http://dx.doi.org/10.48550/arXiv.2203.08890, arXiv preprint arXiv:2203.08890.

[25] T. De Ryck, S. Lanthaler, S. Mishra, On the approximation of functions by tanh neural networks, Neural Netw. 143 (2021) 732–750, http://dx.doi.org/10.1016/j.neunet.2021.08.015.

[26] J. Shin, G. Em Karniadakis, On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs, Commun. Comput. Phys. 28 (5) (2020) 2042–2074, http://dx.doi.org/10.4208/cicp.OA-2020-0193.

[27] Y. Shin, Z. Zhang, G.E. Karniadakis, Error estimates of residual minimization using neural networks for linear PDES, J. Mach. Learn. Model. Comput. 4 (4) (2023) 73–101, http://dx.doi.org/10.1615/JMachLearnModelComput.2023050411.

[28] T. De Ryck, A.D. Jagtap, S. Mishra, Error estimates for physics informed neural networks approximating the Navier-Stokes equations, 2022, http://dx.doi.org/10.48550/arXiv.2203.09346, arXiv preprint arXiv:2203.09346.

[29] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating PDEs, IMA J. Numer. Anal. (2022) http://dx.doi.org/10.1093/imanum/drab093.

[30] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs, IMA J. Numer. Anal. 42 (2) (2022) 981–1022, http://dx.doi.org/10.1093/imanum/drab032.

[31] J. Müller, M. Zeinhofer, Achieving high accuracy with PINNs via energy natural gradients, 2023, http://dx.doi.org/10.48550/arXiv.2302.13163, arXiv preprint arXiv:2302.13163.

[32] B. Wang, W. Zhang, W. Cai, Multi-scale deep neural network (MscaleDNN) methods for oscillatory Stokes flows in complex domains, Commun. Comput. Phys. 28 (5) (2020) 2139–2157, http://dx.doi.org/10.4208/cicp.OA-2020-0192.

[33] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Comput. Methods Appl. Mech. Engrg. 361 (2020) 112732, http://dx.doi.org/10.1016/j.cma.2019.112732.

[34] X. Jin, S. Cai, H. Li, G.E. Karniadakis, NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations, J. Comput. Phys. 426 (2021) 109951, http://dx.doi.org/10.1016/j.jcp.2020.109951.

[35] S. Cai, Z. Wang, S. Wang, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks for heat transfer problems, J. Heat Transfer 143 (6) (2021) http://dx.doi.org/10.1115/1.4050542.

[36] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Comput. Methods Appl. Mech. Engrg. 379 (2021) 113741, http://dx.doi.org/10.1016/j.cma.2021.113741.

[37] S. Goswami, C. Anitescu, S. Chakraborty, T. Rabczuk, Transfer learning enhanced physics informed neural network for phase-field modeling of fracture, Theor. Appl. Fract. Mech. 106 (2020) 102447, http://dx.doi.org/10.1016/j.tafmec.2019.102447.

[38] Z. Mao, A.D. Jagtap, G.E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Engrg. 360 (2020) 112789, http://dx.doi.org/10.1016/j.cma.2019.112789.

[39] M. Raissi, P. Perdikaris, G.E. Karniadakis, 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, http://dx.doi.org/10.1016/j.jcp.2018.10.045.
[40] C.H. Scholz, The Mechanics of Earthquakes and Faulting, third ed., Cambridge University Press, 2019, http://dx.doi.org/10.1017/9781316681473.
[41] C. Marone, Laboratory-derived friction laws and their application to seismic faulting, Ann. Rev. Earth Planet Sci. 26 (1) (1998) 643–696, http://dx.doi.org/10.1146/annurev.earth.26.1.643.
[42] R.A. Harris, M. Barall, R. Archuleta, E.M. Dunham, B. Aagaard, J.P. Ampuero, H. Bhat, V. Cruz-Atienza, L. Dalguer, P. Dawson, S. Day, B. Duan, G. Ely, Y. Kaneko, Y. Kase, N. Lapusta, Y. Liu, S. Ma, D. Oglesby, K. Olsen, A. Pitarka, S. Song, E. Templeton, The SCEC/USGS dynamic earthquake rupture code verification exercise, Seismol. Res. Lett. 80 (2009) 119–126, http://dx.doi.org/10.1785/gssrl.80.1.119.
[43] B.A. Erickson, J. Jiang, M. Barall, N. Lapusta, E.M. Dunham, R. Harris, L.S. Abrahams, K.L. Allison, J.P. Ampuero, S. Barbot, C. Cattania, A. Elbanna, Y. Fialko, B. Idini, J.E. Kozdon, V. Lambert, Y. Liu, Y. Luo, X. Ma, M.B. Mckay, P. Segall, P. Shi, M. van den Ende, M. Wei, The community code verification exercise for simulating sequences of earthquakes and aseismic slip (SEAS), Seismol. Res. Lett. 91 (2020) 874–890, http://dx.doi.org/10.1785/0220190248.
[44] S. Kollmannsberger, D. D'Angella, M. Jokeit, L. Herrmann, et al., Deep Learning in Computational Mechanics, Springer, 2021.
[45] J.H. Dieterich, Modeling of rock friction 1. Experimental results and constitutive equations, J. Geophys. Res., [Solid Earth] 84 (B5) (1979) 2161–2168, http://dx.doi.org/10.1029/JB084iB05p02161.
[46] A. Ruina, Slip instability and state variable friction laws, J. Geophys. Res.: Solid Earth 88 (B12) (1983) 10359–10370, http://dx.doi.org/10.1029/JB088iB12p10359.
[47] M. van den Ende, J. Chen, J.P. Ampuero, A. Niemeijer, A comparison between rate-and-state friction and microphysical models, based on numerical simulations of fault slip, Tectonophysics 733 (2018) 273–295, http://dx.doi.org/10.1016/j.tecto.2017.11.040.
[48] P. Roache, Verification and Validation in Computational Science and Engineering, first ed., Hermosa Publishers, Albuquerque, NM, 1998.
[49] B.A. Erickson, E.M. Dunham, An efficient numerical method for earthquake cycles in heterogeneous media: Alternating subbasin and surface-rupturing events on faults crossing a sedimentary basin, J. Geophys. Res.-Solid Earth 119 (4) (2014) 3290–3316, http://dx.doi.org/10.1002/2013JB010614.
[50] T.W. Harvey, B.A. Erickson, J.E. Kozdon, A high-order accurate summation-by-parts finite difference method for fully-dynamic earthquake sequence simulations within sedimentary basins, J. Geophys. Res.: Solid Earth (2023) http://dx.doi.org/10.1029/2022JB025357, e2022JB025357.
[51] R.H. Byrd, P. Lu, J. Nocedal, C. Zhu, A limited memory algorithm for bound constrained optimization, SIAM J. Sci. Comput. 16 (5) (1995) 1190–1208, http://dx.doi.org/10.1137/0916069.
[52] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
[53] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000, http://dx.doi.org/10.1109/72.712178.
[54] I.E. Lagaris, A.C. Likas, D.G. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, IEEE Trans. Neural Netw. 11 (5) (2000) 1041–1049, http://dx.doi.org/10.1109/72.870037.
[55] B.A. Erickson, J. Jiang, V. Lambert, S.D. Barbot, M. Abdelmeguid, M. Almquist, J.-P. Ampuero, R. Ando, C. Cattania, A. Chen, L. Dal Zilio, S. Deng, E.M. Dunham, A.E. Elbanna, A.-A. Gabriel, T.W. Harvey, Y. Huang, Y. Kaneko, J.E. Kozdon, N. Lapusta, D. Li, M. Li, C. Liang, Y. Liu, S. Ozawa, A. Perez-Silva, C. Pranger, P. Segall, Y. Sun, P. Thakur, C. Uphoff, Y. van Dinther, Y. Yang, Incorporating Full Elastodynamic Effects and Dipping Fault Geometries in Community Code Verification Exercises for Simulations of Earthquake Sequences and Aseismic Slip (SEAS), Bull. Seismol. Soc. Am. 113 (2) (2023) 499–523, http://dx.doi.org/10.1785/0120220066.
[56] C. Wu, M. Zhu, Q. Tan, Y. Kartha, L. Lu, A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. 403 (2023) 115671.
[57] L. Bottou, F.E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM Rev. 60 (2) (2018) 223–311.
[58] X. Qian, D. Klabjan, The impact of the mini-batch size on the variance of gradients in stochastic gradient descent, 2020, arXiv preprint arXiv:2004.13146.
[59] C. Marone, B. Kilgore, Scaling of the critical slip distance for seismic faulting with shear strain in fault zones, Nature 362 (6421) (1993) 618–621, http://dx.doi.org/10.1038/362618a0.