Title

Learning Constitutive Relations From Soil Moisture Data via Physically Constrained Neural Networks

Journal
Water Resources Research, 60(7)

ISSN
0043-1397

Authors
Bandai, Toshiyuki
Ghezzehei, Teamrat A
Jiang, Peishi
et al.

Publication Date
2024-07-01

DOI
10.1029/2024wr037318

Peer reviewed

Learning constitutive relations from soil moisture data via physically constrained neural networks

Toshiyuki Bandai [1], Teamrat A. Ghezzehei [2], Peishi Jiang [3], Patrick Kidger [4], Xingyuan Chen [3], Carl I. Steefel [1]

[1]Earth and Environmental Sciences Area, Lawrence Berkeley National Laboratory, Berkeley, CA, USA
[2]Department of Life and Environmental Sciences, University of California Merced, Merced, CA, USA
[3]Atmospheric Sciences and Global Change Division, Pacific Northwest National Laboratory, Richland, WA, USA
[4]Cradle, Zürich, Switzerland

Key Points:

- We developed a fully-differentiable solver for the Richardson-Richards equation.
- The constitutive relations are represented by physically constrained neural networks.
- The framework can be used to extract soil hydraulic properties without assuming coupling between the constitutive relations.

Corresponding author: Toshiyuki Bandai, `tbandai@lbl.gov`

Abstract

The constitutive relations of the Richardson-Richards equation encode the macroscopic properties of soil water retention and conductivity. These soil hydraulic functions are commonly represented by models with a handful of parameters. The limited degrees of freedom of such soil hydraulic models constrain our ability to extract soil hydraulic properties from soil moisture data via inverse modeling. We present a new free-form approach to learning the constitutive relations using physically constrained neural networks. We implemented the inverse modeling framework in a differentiable modeling framework, JAX, to ensure scalability and extensibility. For efficient gradient computations, we implemented implicit differentiation through a nonlinear solver for the Richardson-Richards equation. We tested the framework against synthetic noisy data and demonstrated its robustness against varying magnitudes of noise and degrees of freedom of the neural networks. We applied the framework to soil moisture data from an upward infiltration experiment and demonstrated that the neural network-based approach was better fitted to the experimental data than a parametric model and that the framework can learn the constitutive relations.

1 Introduction

The Richardson-Richards equation (Richardson, 1922; Richards, 1931) serves as a fundamental equation to simulate water flow in saturated-unsaturated soils. Therein, soil hydraulic properties are expressed as two constitutive relations: 1) the water retention curve that relates the volumetric water content to the water potential and 2) the unsaturated permeability function (or hydraulic conductivity function) that relates the unsaturated permeability to the water potential. The two constitutive relations are called soil hydraulic functions and encode the effect of physical, chemical, and biological processes at a pore scale on the state of soils on a larger scale of interest. Hence, the soil hydraulic functions are intrinsically scaling relations (Miller et al., 1998). Because it is virtually impossible to derive such scaling relations based on first principles in practical situations, soil hydraulic functions need to be inferred from observational data.

Inverse modeling has been employed to estimate soil hydraulic functions from laboratory and field soil moisture data. In such cases, soil hydraulic functions are expressed as parametric models, and the parameters are estimated via inverse modeling. Commonly, such parametric models are built on empirical water retention functions, such as the Brooks and Corey model (Brooks & Corey, 1964) and the van Genuchten model (van Genuchten, 1980), combined with physics-based bundle tube models for relative permeability functions (Burdine, 1953; Mualem, 1976). Although this approach has been widely accepted and successful, there is a fundamental limitation to further improve our understanding of the constitutive relations. That is, we can only analyze observational data through the lens of assumed constitutive relations. When parametric models used for constitutive relations are insufficient to describe observational data, we only describe its failure as a model bias and therefore can get little clue as to how the parametric models are incorrect. This limitation is particularly crucial when analyzing soil moisture data collected in the field because complicated physical, chemical, and biological processes are not considered in commonly used parametric soil hydraulic models. Examples of such processes include the effects of hydrophobicity (Vogelmann et al., 2013), rock fragments (Naseri et al., 2023), and nonequilibrium flow (H. J. Vogel et al., 2023).

To extract the constitutive relations in a more flexible manner, Bitterlich et al. (2004) proposed a free-form approach, in which they used quadratic B-splines and piecewise cubic Hermite interpolation to represent soil hydraulic functions. They demonstrated that the free-form approach could extract soil hydraulic functions from multi-step outflow experiments via inverse modeling. Their free-form approach did not have to assume coupling between water retention functions and relative permeability functions, unlike commonly used parametric models for soil hydraulic functions. This decoupling can prevent errors in water retention functions from propagating into relative permeability functions. Subsequently, Iden and Durner (2007) modified the approach of Bitterlich et al. (2004) and further demonstrated the advantage of the free-form approach against parametric models with a limited number of parameters. Recently, Bandai and Ghezzehei (2021) used monotonic neural networks (Daniels & Velikova, 2010) to represent soil hydraulic functions as components of physics-informed neural networks and attempted to extract the constitutive relations. Although they demonstrated its feasibility against synthetic noisy data, their approach has limitations for near saturation conditions. While physics-informed neural networks have been improved and applied to many scientific domains, their application to realistic problems in vadose zone hydrology appears to be limited by the difficulty in training physics-informed neural networks with noisy sparse data (Bandai & Ghezzehei, 2022).

As a robust and scalable free-form approach to extract the constitutive relations of the Richardson-Richards equation from soil moisture data, we developed a fully-differentiable numerical model of the Richardson-Richards equation using a machine learning library JAX (Bradbury et al., 2018). In our differentiable modeling framework, soil hydraulic functions are represented by monotonic neural networks, as in Bandai and Ghezzehei (2021), but we further imposed additional physical constraints to ensure the robustness of the framework near saturation. Also, unlike their physics-informed neural networks approach, we used a finite volume method with the Backward Euler method to solve the Richardson-Richards equation to guarantee the physical laws, including the conservation of mass and the Buckingham-Darcy law. Compared to previous free-form approaches (Bitterlich et al., 2004; Iden & Durner, 2007), our inverse modeling approach is scalable and extensible because of the efficient derivative computation implemented on JAX. We first tested the performance of our numerical model written in JAX to solve a forward model by comparing it with a Fortran numerical solver. Then, we built an inverse modeling framework to estimate the constitutive relations from soil moisture and tested it against synthetic noisy data. We then applied our inverse modeling framework to extract the constitutive relations from soil moisture data measured in upward infiltration experiments conducted by Sadeghi et al. (2017). Finally, we discuss the challenges and opportunities of the differentiable modeling framework.

2 Forward modeling

In this section, we describe the forward modeling approach used to simulate water flow in variably saturated soils and demonstrate its performance. In Section 2.1, we first introduce the Richardson-Richards equation and the initial and boundary conditions used in the study. In Section 2.2, we describe the van Genuchten-Mualem model, which we used as a baseline model to provide the constitutive relations (i.e., soil hydraulic functions). In Section 2.3, we introduce a machine learning library, JAX, which we used to implement the numerical solver. Finally, in Section 2.4, we demonstrate the performance of the JAX-based forward modeling by comparing it to a Fortran-based numerical solver.

2.1 Richardson-Richards equation

One-dimensional water flow in a rigid and isotropic soil can be described by the Richardson-Richards equation (Richardson, 1922; Richards, 1931). The mass balance of water on a spatial domain $\Omega := (-Z, 0)$ leads to

$$\frac{\partial \theta}{\partial t} = -\frac{\partial q}{\partial z} \quad \text{for} \quad \Omega \times (0, T), \tag{1}$$

where t is the time [T], T is the final time [T], z is the spatial coordinate that is positive upward with $z = 0$ set to the surface of the soil [L], Z is the length of the soil [L],

θ is the volumetric water content [L^3 L^{-3}], and q is the water flux [L T^{-1}] described by the Buckingham-Darcy law (Buckingham, 1907)

$$q = -\frac{Kk_r\rho g}{\mu}\left(\frac{\partial\psi}{\partial z} + 1\right), \tag{2}$$

where K is the permeability [L^2], k_r is the relative permeability [-], ρ is the density of water [M L^{-3}] (= 0.99823×10^3 [kg m^{-3}]), μ is the dynamic viscosity of water [M L^{-1} T^{-1}] (= 1.0005 × 10^{-3} [kg m^{-1} s^{-1}]), g is the gravitational acceleration [M T^{-2}] (= 9.80665 [m s^{-2}]), and ψ is the water potential [L]. We introduce the hydraulic head h [L] as $h := \psi + z$.

We consider the following initial and boundary conditions:

$$\begin{aligned} \psi(z,0) &= \psi_i(z) \quad \text{for} \quad z \in [-Z, 0] && (3)\\ \psi(-Z,t) &= \psi_{lb} \quad \text{for} \quad t \in (0,T), && (4)\\ q(z,t) &= q_{ub} \quad \text{for} \quad z = 0, \quad t \in (0,T), && (5) \end{aligned}$$

where ψ_i is the initial condition, ψ_{lb} is the water potential at the lower boundary, and q_{ub} is the water flux at the upper boundary. Although we limited our analysis to the initial and boundary conditions above, our approach is applicable to other conditions.

2.2 Soil hydraulic functions

We need two constitutive relations, $\theta(\psi)$ and $k_r(\psi)$, to solve the Richardson-Richards equation (Equation 1 and Equation 2). These two soil hydraulic functions are referred to as the water retention curve and the relative permeability function, respectively. Both functions are nonlinear functions of the water potential ψ and represent the macroscopic water holding and water transport properties of the soil. Although the two functions can exhibit hysteresis under wetting and drying cycles, we neglected the effect of hysteresis in this study. We used the van Genuchten-Mualem model (Mualem, 1976; van Genuchten, 1980) as the baseline model.

The water retention curve of the van Genuchten-Mualem model is described as

$$\begin{aligned} \theta(\psi) &= \theta_r + (\theta_s - \theta_r)S_e(\psi) \quad \text{for} \quad \psi < 0, && (6)\\ \theta(\psi) &= \theta_s \quad \text{for} \quad \psi \geq 0, && (7) \end{aligned}$$

where θ_r is the residual volumetric water content [L^3 L^{-3}], θ_s is the saturated volumetric water content [L^3 L^{-3}], and S_e is the effective saturation [-]

$$S_e(\psi) := \frac{\theta(\psi) - \theta_r}{\theta_s - \theta_r}, \tag{8}$$

which is parameterized as

$$S_e(\psi) = (1 + (-\alpha\psi)^n)^{-m}, \tag{9}$$

where α [L^{-1}] and n [-] are van Genuchten fitting parameters, and m is defined as $m := 1 - 1/n$. The relative permeability function is derived from Mualem's bundle tube model (Mualem, 1976), resulting in

$$\begin{aligned} k_r(\psi) &= S_e(\psi)^\tau \left(\frac{\int_0^{S_e} \frac{1}{\psi(S_e)} dS_e}{\int_0^1 \frac{1}{\psi(S_e)} dS_e}\right)^2 \quad \text{for} \quad \psi < 0, && (10)\\ k_r(\psi) &= 1.0 \quad \text{for} \quad \psi \geq 0, && (11) \end{aligned}$$

where τ is the tortuosity parameter [-]. Substituting the van Genuchten's water retention curve into S_e, we obtain the analytical expression of the relative permeability function

$$\begin{aligned} k_r(\psi) &= S_e(\psi)^\tau(1 - (1 - S_e(\psi)^{1/m})^m)^2 \quad \text{for} \quad \psi < 0, && (12)\\ k_r(\psi) &= 1.0 \quad \text{for} \quad \psi \geq 0. && (13) \end{aligned}$$

2.3 Scientific computing in JAX

We solved the Richardson-Richards equation (Equation 1 and Equation 2) using a finite volume method with the Backward Euler method. The resulting system of nonlinear equations was solved by the Newton method with Armijo backtracking line search (Appendix A). We implemented the numerical method in Python using JAX (Frostig et al., 2018; Bradbury et al., 2018). JAX is a machine learning framework supported by a machine learning-focused compiler called XLA (Accelerated Linear Algebra). In the last few years, JAX has been successfully used in scientific computing in many domains, including molecular dynamics (Schoenholz & Cubuk, 2020), fluid mechanics (Kochkov et al., 2021; Bezgin et al., 2023), ocean modeling (Häfner et al., 2021), and solid mechanics (Xue et al., 2023). Scientific computing in JAX has the following distinctive features:

1. It is possible to implement numerical methods in a high-level, interpreted programming language, Python, and thus drastically reduce the development cost while achieving a high computing performance due to code optimization with the XLA compiler.
2. JAX supports automatic differentiation, which eliminates the need to linearize numerical models manually for nonlinear solvers.
3. JAX provides a function **vmap**, which automatically vectorizes a Python function.
4. XLA automatically compiles Python codes for specific accelerators, including CPUs, GPUs, and TPUs, without source code modifications.
5. JAX supports parallel computation across CPU and GPU cores, although this is not yet implemented.
6. We can capitalize on the extensive JAX ecosystem for machine learning tools and other purposes. We used Equinox (Kidger & Garcia, 2021) for handling JAX data structures (called Pytrees), Lineax (Rader et al., 2023) for linear solvers, and Optax (DeepMind et al., 2020) for optimization.

While JAX has many favorable features for scientific computing, its design also has some limitations. First, functions used in JAX need to be pure functions without any side effects. Second, JAX does not support dynamically-shaped arrays. Thus, it currently appears to be difficult to implement adaptive spatial discretization. Finally, to optimize JAX Python codes, we need first to run the codes so that JAX traces the computation and XLA optimizes it. When Python codes include native Python **for** and **while** loops, the compilation takes a long time and often fails. JAX provides structured control flow, such as **lax.fori_loop** and **lax.while_loop**, to avoid such compilation issues, but it can limit the capability of XLA to optimize the Python codes.

We refer to our differentiable numerical solver for the Richardson-Richards equation in JAX as JAX-Richards. The JAX-Richards approach is distinct from the existing unsaturated-saturated solvers, such as HYDRUS (Šimůnek et al., 2016), AmanziATS (Coon et al., 2020), PFLOTRAN (Hammond et al., 2014), and CrunchTope (Steefel et al., 2015), all of which are implemented in compiled languages Fortran and C++. The source code of JAX-Richards is shared through Bandai, Ghezzehei, et al. (2024).

2.4 Performance

We investigated the performance of JAX-Richards by comparing it with a Fortran program that implemented the same mathematical algorithm. As a benchmark problem, we simulated one-dimensional vertical infiltration into a dry homogeneous soil with a length of 6.0 m. This benchmark test was used in previous studies (Forsyth et al., 1995; T. Vogel et al., 1996). We used the van Genuchten-Mualem model for the soil hydraulic functions, and its parameters are as follows: $\theta_r = 0.0$, $\theta_s = 0.33$, $\alpha = 1.43$ m^{-1}, $n = 1.506$, $\tau = 0.5$, and $K = 2.95\times10^{-13}$ m^2. We set the initial water potential as $\psi_i =$ -7.26139

Table 1. Wall time [s] to solve the benchmark problem 1 (Figure 1) by Fortran, JAX-Richards on a CPU and a GPU, respectively, for varying numbers of the spatial cells N_s. The number of time steps was 650.

N_s	Fortran	JAX-CPU	JAX-GPU
60	0.078	0.144	0.859
120	0.264	2.05	1.27
240	1.081	11.1	3.00
480	4.277	59.0	8.17
960	20.646	180	30.9
1920	91.381	516	128

m, corresponding to a volumetric water content θ of 0.1. A constant flux boundary condition $q_{ub} = -0.2$ m day^{-1} was applied to the top boundary, while a constant Dirichlet boundary condition $\psi_{lb} = -7.26139$ m was used for the lower boundary. The final time was set to $T = 6.5$ days, and a fixed time-stepping of 0.01 days was used. We uniformly discretized the spatial domain and varied the number of cells N_s as follows: $N_s = 60, 120, 240, 480, 960, 1920$. Figure 1 shows the volumetric water content θ at $t = 0.0, 1.0, 4.0, 6.5$ days for $N_s = 120$. We verified that the results from the Fortran program and JAX-Richards matched up to 14 digits in double precision.

Table 1 summarizes the performance of the Fortran program and JAX-Richards. Here, linear systems were solved by the DGESV routine in LAPACK for Fortran and by **lx.linear_solve** (a function in Lineax library to call) for JAX-Richards. We compiled the Fortran program with Intel Fortran Compiler Classic 2021.10.0 and ran it on a CPU (13th Gen Intel(R) Core(TM) i9-13900H 2.60 GHz). JAX-Richards was optimized by XLA during the first runtime. We ran JAX-Richards on the CPU and a GPU (GeForce RTX 4070 Laptop) in the Windows Subsystem for Linux Kernel 2. The version of Python and JAX was 3.9.18 and 0.4.19, respectively. The wall time in Table 1 is only for the time-stepping of the benchmark problem and does not include the time for the compilation and the input/output. The result demonstrated that the Fortran program was the fastest, although the JAX on the GPU was competitively fast. As the problem size increased, the wall time for JAX-Richards on the GPU approached that of the Fortran program. This is because for large-scale problems, overhead by Python operations (e.g., data transfer between the host CPU and the GPU) becomes negligible relative to the cost for array operations, which are efficiently computed on the GPU. In future work, we aim to speed up the forward modeling in JAX by implementing variable time steps and parallel computations across GPU cores.

3 Inverse modeling

In this section, we describe a framework to extract the constitutive relations (i.e., soil hydraulic functions) from soil moisture data. Figure 2 shows the overview of the inverse modeling framework. In Section 3.1, we introduce physically constrained neural networks as a free-form approach to parameterize the soil hydraulic functions for inverse modeling. In Section 3.2, we explain the inverse modeling framework. In Section 3.3, we describe implicit differentiation, which enables us to compute derivatives through the nonlinear solver used to solve the Richardson-Richards equation. In Section 3.4, we show the feasibility of the framework against noisy synthetic data. Finally, in Section 3.5, we demonstrate the performance of the framework to extract the soil hydraulic functions from soil moisture data from upward infiltration experiments conducted by Sadeghi et al. (2017).

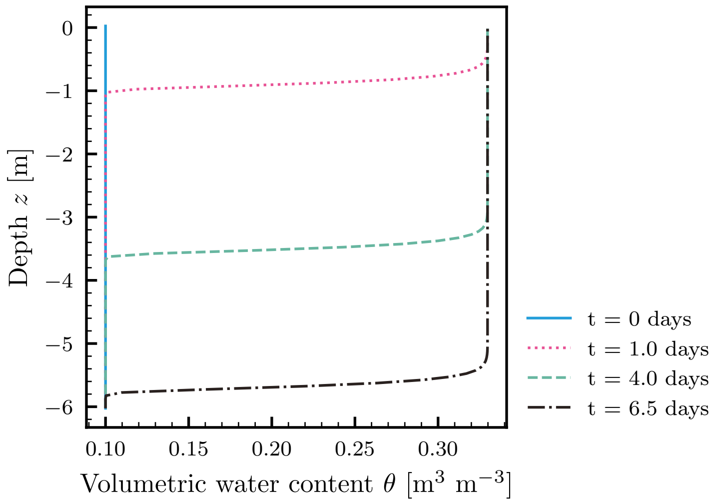

Figure 1. Benchmark problem 1: Infiltration into a homogeneous dry soil. The solution was obtained with the number of cells $N_s = 120$.

3.1 Physically constrained neural networks

We introduce physically constrained neural networks to represent soil hydraulic functions (Figure 2 (**a**) and (**b**)). Assuming there is no hysteresis, we enforced the following physical constraints: (1) water retention curves $\theta(\psi)$ and relative permeability functions k_r are monotonically non-decreasing functions of the water potential ψ; (2) $0 \leq \theta \leq \theta_s$ and $0 \leq k_r \leq 1.0$; and (3) $\theta = \theta_s$ and $k_r = 1.0$ at saturation (i.e., $\psi = 0.0$). Bandai and Ghezzehei (2021) used monotonic neural networks (Daniels & Velikova, 2010) to enforce the monotonicity constraint (constraint (1)), although the other two constraints were not met. Below, we describe our modified monotonic neural networks to represent soil hydraulic functions satisfying all three physical constraints.

We used a feedforward neural network with one hidden layer. The input to the neural network $\mathcal{N}$ is a scalar x, and the output is also a scalar value $\hat{y}$:

$$\hat{y} := \mathcal{N}(x). \tag{14}$$

The input variable x is transformed by the composition of affine transformation and nonlinear activation functions in the following way:

$$\begin{aligned} \mathbf{h} &:= \tanh(\mathbf{W}_h x + \mathbf{b}_h), \\ \hat{y} &:= o(\mathbf{W}_o \mathbf{h} + \mathbf{b}_o), \end{aligned} \tag{15}$$

where $\mathbf{h} \in \mathbb{R}^{n_h}$ is the vector corresponding to the hidden layer with n_h units, $\mathbf{W}_h \in \mathbb{R}^{n_h \times 1}$ and $\mathbf{b}_h \in \mathbb{R}^{n_h}$ are weight matrix and bias vector, respectively, for the hidden layer, $\mathbf{W}_o \in \mathbb{R}^{1 \times n_h}$ and $\mathbf{b}_o \in \mathbb{R}$ are weight matrix and bias vector, respectively, for the output layer, o is the output function, which is defined as $o(x) := 2\sigma(x)$ with σ being the sigmoid function for water retention curves and $o(x) := 10^x$ for relative permeability functions. The neural network $\mathcal{N}$ was used to represent water retention curves $\theta(\psi)$ and relative permeability functions $k_r(\psi)$ in the following way:

$$\theta(\psi) = \theta_s \mathcal{N}_\theta(\psi) \quad \text{for} \quad \psi < 0, \tag{16}$$

$$\theta(\psi) = \theta_s \quad \text{for} \quad \psi \geq 0, \tag{17}$$

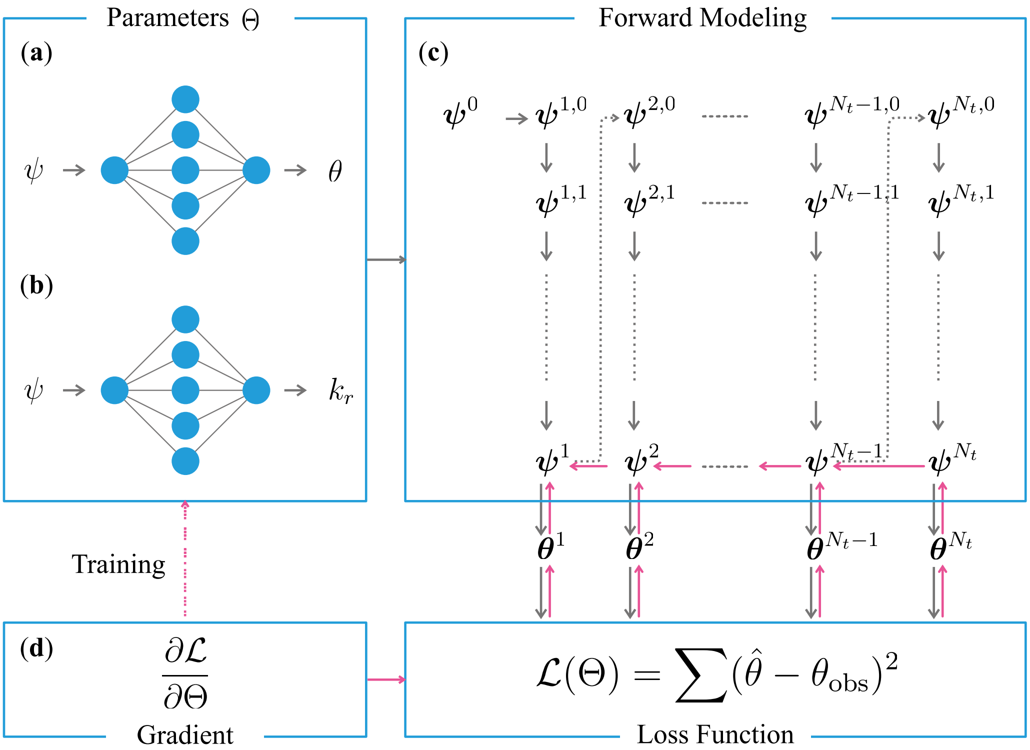

Figure 2. The overview of the inverse modeling framework. (**a**): Physically constrained neural network for the water retention curve (i.e., the volumetric water content θ with respect to the water potential ψ). (**b**): Physically constrained neural network for the relative permeability function (i.e., the relative permeability k_r with respect to the water potential ψ). (**c**): Forward modeling to solve the Richardson-Richards equation (Appendix A). The solution for each time step is iteratively obtained by the Newton method. Here, $\boldsymbol{\psi}^{n,k}$ is the water potential at all the spatial nodes at the nth time step and the kth Newton iteration step. The solution for each time step was converted into the volumetric water content by the neural network for the water retention curve and inserted into the loss function as the predicted volumetric water content $\hat{\theta}$. (**d**): The gradient of the loss function $\mathcal{L}$ is computed by the reverse-mode automatic differentiation with implicit differentiation, shown as the pink solid arrows. The gradient is used to update the set of parameters Θ.

for water retention curves and

$$k_r(\psi) = \mathcal{N}_{k_r}(\psi) \quad \text{for} \quad \psi < 0, \tag{18}$$
$$k_r(\psi) = 1.0 \quad \text{for} \quad \psi \geq 0, \tag{19}$$

for relative permeability functions. Here, we used the subscript θ and k_r to emphasize the fact that the soil hydraulic functions do not share a single neural network. Thus, the soil hydraulic functions are not coupled, unlike commonly used parametric models like the van Genuchten-Mualem model. While not necessary, we used the same number of the hidden units n_h for both neural networks (i.e., $\mathcal{N}_\theta$ and $\mathcal{N}_{k_r}$). We enforced the three physical constraints in the following manner. First, we forced the weight parameters $\mathbf{W}_h$ and $\mathbf{W}_o$ to be non-negative to make the neural network monotonically non-decreasing function of the input (Daniels & Velikova, 2010), which guarantees that the resulting soil hydraulic functions (Equation 16 and Equation 18) are monotonically non-decreasing functions of the water potential ψ (physical constraint (1)). Second, we set the bias parameters $\mathbf{b}_h$ and $\mathbf{b}_o$ to zero vectors to ensure $\mathcal{N}(0) = 1.0$. This setting ensures that the resulting soil hydraulic functions satisfy the physical constraints (2) and (3), as well as the continuity of the soil hydraulic functions at saturation $\psi = 0.0$, which is critical for solving the system of nonlinear equations resulting from the discretization of the Richardson-Richards equation.

3.2 Inverse modeling framework

Here, we describe the inverse modeling framework used to estimate soil hydraulic functions from soil moisture data. We used physically constrained neural networks to represent water retention curves $\theta(\psi)$ and relative permeability functions $k_r(\psi)$. We initialized the parameters of the two neural networks $\mathcal{N}_\theta$ and $\mathcal{N}_{k_r}$ by the Xavier initialization (Glorot & Bengio, 2010) and assembled all the weight matrices as $\mathbf{W} := \{\mathbf{W}_{h,\theta}, \mathbf{W}_{o,\theta}, \mathbf{W}_{h,k_r}, \mathbf{W}_{o,k_r}\}$. In addition to the neural network parameters $\mathbf{W}$, we also estimated the two physical parameters, the saturated volumetric water content θ_s and the permeability K. Because we need to constrain the range of the parameters to prevent the nonlinear solver from not converging, we used the following transformations:

$$\theta_s = \mu_{\theta_s} + \sigma_{\theta_s} \tanh \theta_s^t, \tag{20}$$
$$\log_{10} K = \mu_{\log_{10} K} + \sigma_{\log_{10} K} \tanh K^t, \tag{21}$$

where μ_{θ_s} and $\mu_{\log_{10} K}$ are the mean for the saturated water content θ_s and the permeability K in log scale, respectively, σ_{θ_s} and $\sigma_{\log_{10} K}$ are half of the range of the saturated water content θ_s and the permeability K in log scale, respectively, θ_s^t and K^t are the transformed physical parameters. While the transformed parameters were initialized to zero, the mean and the range values were predetermined based on available prior information. Thus, the set of parameters estimated in the inverse modeling framework Θ is the neural network parameters $\mathbf{W}$ and the transformed physical parameters θ_s^t and K^t. The set of the parameters $\Theta = \{\mathbf{W}, \theta_s^t, K^t\}$ were simultaneously estimated by minimizing the empirical loss function $\mathcal{L}$:

$$\mathcal{L}(\Theta) = \frac{1}{N_\theta} \sum_{i=1}^{N_\theta} \left(\frac{\hat{\theta}(z^i, t^i; \Theta) - \theta_{\text{obs}}(z^i, t^i)}{\sigma_\theta^i} \right)^2, \tag{22}$$

where $\theta_{\text{obs}}(z^i, t^i)$ is the volumetric water content data observed at $z = z^i$ and $t = t^i$ for $i = 1, 2, ..., N_\theta$, $\hat{\theta}(z^i, t^i; \Theta)$ is the predicted volumetric water content at $z = z^i$ and $t = t^i$ for $i = 1, 2, ..., N_\theta$ by solving the forward problem given the set of the parameters Θ, and σ_θ^i is the inverse of the weight for each measurement data $\theta_{\text{obs}}(z^i, t^i)$. Commonly, the standard deviation of the measurement error is used for σ_θ^i, which makes the minimization problem the maximum likelihood estimation. Unfortunately, however, it is also common for the measurement error to be unknown. In the current framework, we fixed σ_θ^i rather than estimating them during the inverse modeling.

To minimize the loss function $\mathcal{L}$, we used the Adam optimizer implemented in the JAX-based optimization library Optax (DeepMind et al., 2020) with the default parameters ($b1 = 0.9$, $b2 = 0.999$, $eps = 10^{-8}$, $eps\_root = 0.0$). We set the learning rate to 10^{-2}. The number of iterations of the Adam optimizer was determined for each inverse problem. To avoid stopping at a bad solution, we additionally ran the Adam optimizer to obtain the lowest loss value. The Adam optimizer requires the gradient of the loss function with respect to the parameters Θ, which were computed by reverse-mode automatic differentiation implemented in JAX with implicit differentiation through the nonlinear solver.

3.3 Implicit differentiation through nonlinear solver

If we were to naively apply reverse-mode automatic differentiation to the numerical solver for the Richardson-Richards equation, JAX would need to trace all the Newton iterations and the line searches conducted in the nonlinear solver every time step. This would result in a huge computational and memory cost. To avoid this issue, we implemented implicit differentiation through the nonlinear solver (Griewank & Walther, 2008). We consider the system of nonlinear equations

$$\mathbf{F}(\mathbf{x}, \mathbf{a}) = \mathbf{0}, \tag{23}$$

where $\mathbf{F} : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is the system of nonlinear equations with the solution vector $\mathbf{x} \in \mathbb{R}^n$ and the parameter vector $\mathbf{a} \in \mathbb{R}^p$. In our case, the solution vector $\mathbf{x}$ is the solution of the Richardson-Richards equation for each time step, and the parameter vector $\mathbf{a}$ corresponds to the set of parameters used for the soil hydraulic functions (i.e., Θ). We aim to compute the derivative $\frac{\partial \mathbf{x}}{\partial \mathbf{a}}$ for inverse modeling, where $\mathbf{x}$ is implicitly defined by the system of nonlinear equations (Equation 23). If we assume that $\mathbf{F}$ is continuously differentiable and the Jacobian matrix $\frac{\partial \mathbf{F}}{\partial \mathbf{x}}$ is not singular, implicit function theorem leads to

$$\frac{\partial \mathbf{F}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{a}} + \frac{\partial \mathbf{F}}{\partial \mathbf{a}} = \mathbf{0}, \tag{24}$$

which results in

$$\frac{\partial \mathbf{x}}{\partial \mathbf{a}} = -\left[\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right]^{-1} \frac{\partial \mathbf{F}}{\partial \mathbf{a}}. \tag{25}$$

While the equation above gives the desired derivative $\frac{\partial \mathbf{x}}{\partial \mathbf{a}}$, we actually need the Jacobian-vector product given a vector $\mathbf{v} \in \mathbb{R}^p$

$$\frac{\partial \mathbf{x}}{\partial \mathbf{a}} \mathbf{v} = -\left[\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right]^{-1} \mathbf{u}, \tag{26}$$

where $\mathbf{u} := \frac{\partial \mathbf{F}}{\partial \mathbf{a}} \mathbf{v}$ can be computed by the Jacobian-vector product of $\mathbf{F}$ via automatic differentiation. The Jacobian-vector product $\frac{\partial \mathbf{x}}{\partial \mathbf{a}} \mathbf{v}$ is the solution to the linear system

$$\frac{\partial \mathbf{F}}{\partial \mathbf{x}} \left[\frac{\partial \mathbf{x}}{\partial \mathbf{a}} \mathbf{v}\right] = -\mathbf{u}. \tag{27}$$

The Jacobian-vector product $\frac{\partial \mathbf{x}}{\partial \mathbf{a}} \mathbf{v}$ was implemented as a custom derivative rule for the nonlinear solver by using **jax.custom_jvp** class. While our inverse modeling framework requires the vector-Jacobian product ($\mathbf{w}^T \frac{\partial \mathbf{x}}{\partial \mathbf{a}}$ for a vector $\mathbf{w} \in \mathbb{R}^n$), JAX automatically derives it from the custom Jacobian-vector product rule, which is further automatically incorporated into the reverse-mode automatic differentiation (Radul et al., 2023). This automation is independent of the type of numerical solvers and loss functions, which makes this approach scalable with respect to the development time.

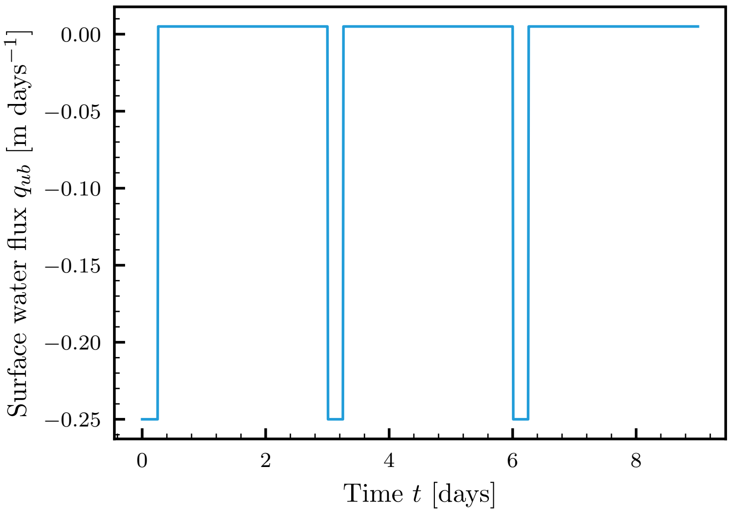

Figure 3. Transient upper flux boundary condition used in Benchmark problem 2 for $C_{ub} = 3$. The positive and negative values represent evaporation and rainfall, respectively.

3.4 Numerical test against synthetic noisy data

We tested the inverse modeling framework against noisy synthetic soil moisture data. We generated the noisy synthetic data by solving a forward problem given van Genuchten-Mualem model parameters (Section 3.4.1). We verified the derivative of the loss function against finite difference methods (Section 3.4.2). Then, we applied the inverse modeling framework to recover the original constitutive relations from the synthetic noisy data (Section 3.4.3).

3.4.1 Generating noisy synthetic data

We generated synthetic data by solving the one-dimensional Richardson-Richards equation (Equation 1 and Equation 2). The soil length is $Z = 1.5$ m, which was uniformly discretized into 150 cells. We simulated wetting and drying cycles by applying a transient upper flux boundary condition as follows:

$$q_{ub} = q_{\text{rain}} \quad \text{for} \quad T_{ub}(C_{ub} - 1) \leq t < T_{ub}(C_{ub} - 1) + T_{\text{rain}}, \tag{28}$$

$$q_{ub} = q_{\text{eva}} \quad \text{for} \quad T_{ub}(C_{ub} - 1) + T_{\text{rain}} \leq t < T_{ub}C_{ub}, \tag{29}$$

where T_{ub} is the period of the cycle [T], T_{rain} is the duration of rainfall in each cycle [T], C_{ub} is the number of the cycles. We set $q_{\text{rain}} = -0.25$ m days^{-1}, $q_{\text{eva}} = 0.005$ m days^{-1}, $T_{ub} = 3.0$ days, and $T_{\text{rain}} = 0.25$ days. We varied the number of cycles C_{ub} in each test conducted later. Figure 3 shows the transient upper flux boundary condition for $C_{ub} = 3$. The final time T is $T = T_{ub}C_{ub}$, and a fixed time-stepping of 0.01 days was used. The other setting is the same as the benchmark problem used for forward modeling (Section 2.4). We added Gaussian noise to the numerical solution to generate noisy synthetic data. Figure 4 shows the noisy synthetic data with $C_{ub} = 3$ with Gaussian noise with a standard deviation of 0.005.

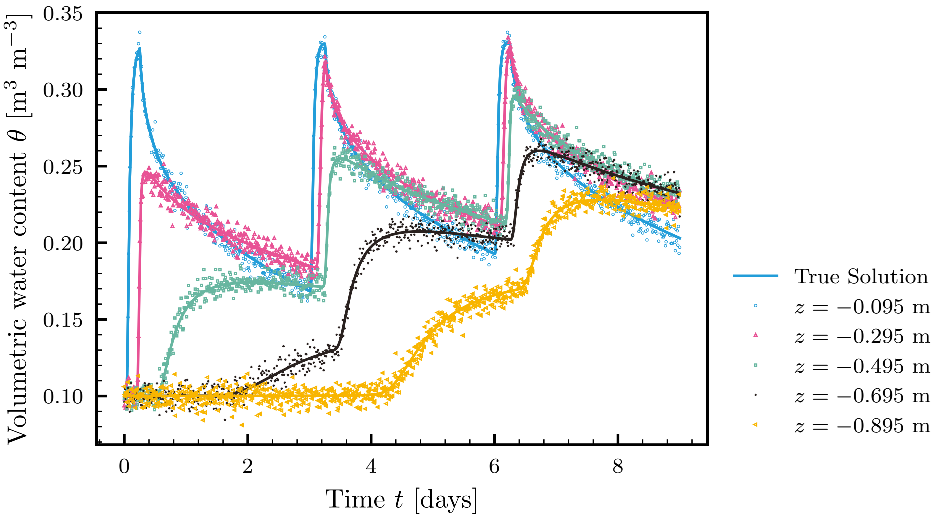

Figure 4. Noisy synthetic data for Benchmark problem 2. The transient upper flux boundary condition (Figure 3) was used.

3.4.2 Testing reverse-mode automatic differentiation

Before we applied the inverse modeling framework to the synthetic noisy data, we tested the implementation and performance of reverse-mode automatic differentiation with implicit differentiation. We conducted the testing in the parameter space near the true solution. To achieve that, we trained the neural networks $\mathcal{N}_\theta$ and $\mathcal{N}_{k_r}$ by the true soil hydraulic functions. We ran the Adam optimizer with the default parameters 20,000 times to obtain the neural network parameters $\mathbf{W}$. We used the true parameters for the saturated volumetric water content θ_s and the permeability K.

We verified the gradient of the loss function $\mathcal{L}$ with respect to the parameters Θ computed by reverse-mode automatic differentiation with implicit differentiation by comparing it with first and second order finite difference methods, following Hückelheim et al. (2023). As observational soil moisture data θ_{obs}, we used the synthetic data with Gaussian noise with a standard deviation of 0.005 obtained by setting $C_{ub} = 3$ and sampled them at every time step at five different depths ($z = -0.1, -0.3, -0.5, -0.7, -0.9$ m). We set the number of hidden layer units to 10 for both neural networks. We computed a directional derivative of the loss function $\nabla\mathcal{L}\cdot\mathbf{v}$ in a random direction $\mathbf{v}$ given the parameters Θ obtained above. Because the tolerance of the nonlinear solver affects the accuracy of the gradient by all the methods, we tightened the tolerance of the nonlinear solver to $\tau_a = 10^{-12}$ for this test. Figure 5 demonstrates the trade-off between truncation errors (for a large step size) and round-off errors (for a small step size) and suggests that reverse-mode automatic differentiation with implicit differentiation provides accurate gradients.

Next, we evaluated the computational time spent on updating the set of parameters Θ. To investigate the scalability of the framework, we changed the number of units in the hidden layer n^h to $n^h = 10, 20, 40, 80, 160$ for both neural networks, which led to the total neural network parameters $N_{\mathcal{N}} = 40, 80, 160, 320, 640$, respectively. We fixed the final simulation time T by setting $C_{ub} = 3$. The other settings remained the same as in the previous test. To demonstrate the efficiency of reverse-mode automatic differentiation with implicit differentiation, we compared the wall time for updating param-

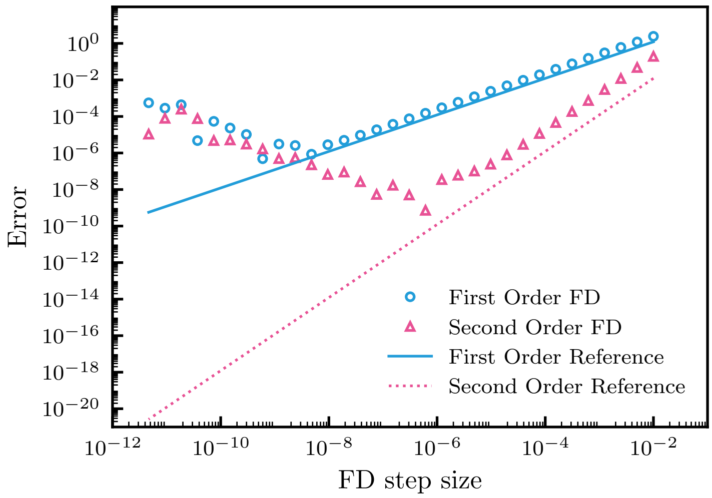

Figure 5. Finite difference (FD) test for the reverse-mode automatic differentiation with implicit differentiation. The x-axis is the step of a finite difference method, and the y-axis is the difference in the computed directional derivatives between the reverse-mode automatic differentiation and first and second order finite difference methods.

$N_{\mathcal{N}}$	JAX-CPU-forward	JAX-CPU-update	JAX-GPU-forward	JAX-GPU-update
40	1.65	4.41	1.35	2.34
80	5.59	7.13	1.45	2.61
160	12.0	21.3	1.59	2.63
320	7.27	16.5	1.63	2.88
640	15.6	26.7	1.98	3.28

Table 2. Wall time [s] spent on solving the forward problem and updating the parameters Θ on the CPU or the GPU for varying numbers of neural network parameters $N_{\mathcal{N}}$.

eters Θ (i.e., solving the forward problem, computing the gradient, and updating the parameters Θ) with that for solving the forward problem. Table 2 demonstrates that the ratio of the wall time is less than 2 for the GPU. This efficiency was due to the implicit differentiation, in which we only need to solve a linear system for each time step during the reverse-mode automatic differentiation, not the system of nonlinear equations required to solve the forward problem (pink arrows in Figure 2). We emphasize that this would not be the case if we had used a finite difference method to compute the gradient because the wall time for updating the parameters in this case would scale with the number of parameters. We observed a degraded performance on the CPU to solve the forward problem (and thus updating neural network parameters) with an increasing number of neural network parameters. Furthermore, we observed a long compilation time for the CPU and large variances in the wall time among multiple runs for both solving the forward problem and updating the parameters. On the other hand, the GPU's performance was consistent regardless of the number of neural network parameters, which demonstrates the advantage of using a GPU (He, 2023).

3.4.3 Learning constitutive relations

We applied the inverse modeling framework to extract the constitutive relations from the noisy synthetic volumetric water content data. In these numerical experiments, we specifically studied the effects of (1) the number of units in the hidden layer, (2) the magnitude of the noise in the data, (3) the initialization of the neural network parameters, and (4) the amount of data in time and space. For the first three goals (1, 2, 3), we varied the number of units in the hidden layer from 5, 10, 20, 40, and 80 and the magnitude of the noise by changing the standard deviation of Gaussian noise from 0.005, 0.01 to 0.02. These simulations were conducted with a fixed data amount ($C_{ub} = 3$ and five locations as above) and three different random seeds to generate different neural network initializations, leading to a total of 45 runs. For the last goal (4), we varied the number of cycles C_{ub} from one to three and the number of observation locations from one ($z = -0.5$ m), three ($z = -0.1, -0.5, -0.9$ m), to five ($z = -0.1, -0.3, -0.5, -0.7, -0.9$ m). These simulations were conducted with a fixed magnitude of noise (a standard deviation of 0.005) and a number of units in the hidden layer (10 units). We initialized the transformed physical parameters θ_s^t and K^t to zero and set $\mu_{\theta_s} = 0.3$, $\sigma_{\theta_s} = 0.1$, $\mu_{\log_{10} K} = -13.0$, and $\sigma_{\log_{10} K} = 2.0$. In the loss function (Equation 22), we set σ_θ^i to the magnitude of the noise for each case. We ran the Adam optimizer 10,000 times in the numerical experiments for each case.

To evaluate the performance of the framework, we computed the relative L2 error for the fitted volumetric water content $\hat{\theta}(z^i, t^i)$:

$$\epsilon_\theta = \frac{1}{N_\theta} \sum_{i=1}^{N_\theta} \left(\frac{\hat{\theta}(z^i, t^i) - \theta_{\text{true}}(z^i, t^i)}{\theta_{\text{true}}(z^i, t^i)} \right)^2, \tag{30}$$

where θ_{true} is the true numerical solution used to generate the synthetic noisy data. We also evaluated the accuracy of the estimated constitutive relations. To achieve that, we evaluated the estimated constitutive relations at ψ_{test} that are uniformly distributed in log scale and computed the relative L2 error for the constitutive relations

$$\epsilon_\gamma^c = \frac{1}{N_\gamma} \sum_{\psi \in \{\psi \in \psi_{\text{test}} \mid \min \theta_{\text{true}} \leq \hat{\theta}(\psi) \leq \max \theta_{\text{true}}\}} \left(\frac{\hat{\gamma}(\psi) - \gamma(\psi)}{\gamma(\psi)} \right)^2, \tag{31}$$

where $\gamma = \theta, \log_{10} Kkr$, and N_γ is the number of the evaluation points.

Figure 6 summarizes the performance of the neural network approach with a number of hidden units n_h being 20 to extract the constitutive relations from the noisy synthetic soil moisture data at five observation points. We set $C_{ub} = 3$, and the magnitude of the noise was 0.02. Figure 6 (**a**) demonstrates that the recovered soil moisture dynamics matched very well with the ground truth volumetric water content data. Figure 6 (**b**) and (**c**) show that the estimated water retention curve and unsaturated permeability function agreed well with the true functions in the range of the data, respectively. On the other hand, the extrapolation capability of the neural network approach was poor. This could be improved by setting the water potential at the dry end (e.g., water potential for oven-dry soils).

Figure 7 demonstrates that the inverse modeling framework consistently succeeds in recovering the constitutive relations regardless of the number of hidden units and the magnitude of the noise. Also, the results show that the neural networks were robust against noise even when the neural networks had many parameters. The neural networks' robustness against noise is probably due to implicit bias, in which over-parametrized neural networks tend to learn a simple structure from data (Chou et al., 2024). This property is not trivial because traditional approaches (e.g., using linear interpolation functions to represent the constitutive relations) would be overfitted to the noisy data and require some regularization techniques.

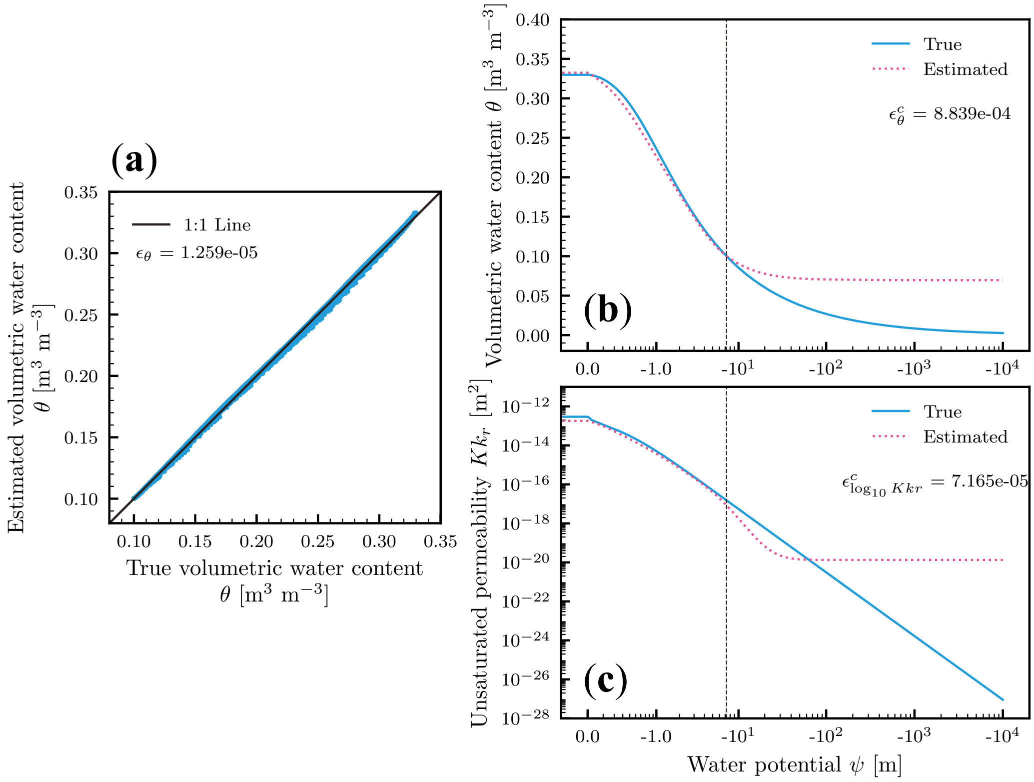

Figure 6. The performance of the neural network approach to extract the constitutive relations in the Richardson-Richards equation from the noisy synthetic soil moisture data. (**a**): 1:1 comparison between the true and the simulated volumetric water content at all temporal and spatial points. (**b**): The estimated water retention curve. (**c**): The estimated unsaturated permeability function. The vertical lines in (**b**) and (**c**) represent the minimum water potential observed in the ground truth soil moisture data.

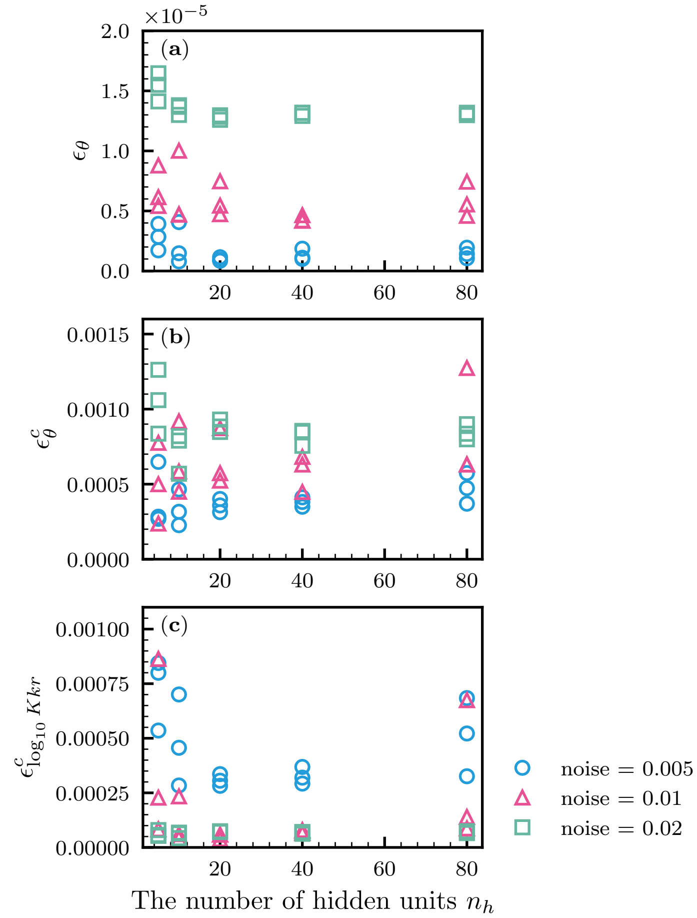

Figure 7. The effects of the number of hidden units n_h and the magnitude of the noise. Each setting was tested with three different neural network initializations. (**a**): L2 error for the simulated volumetric water content. (**b**): L2 error for the water retention curve. (**c**): L2 error for the unsaturated permeability function in log scale.

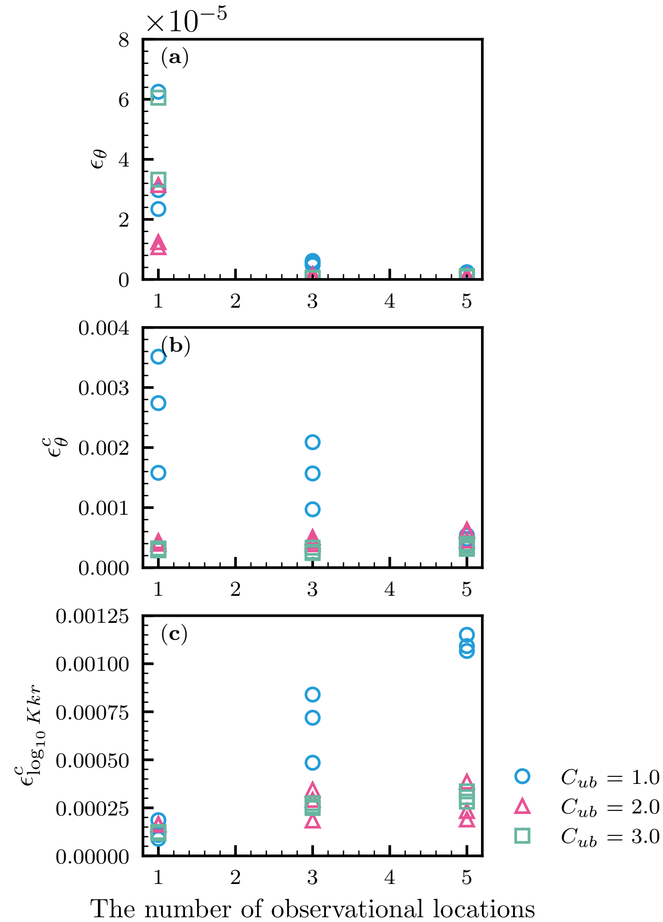

Figure 8. The effects of the amount of data by varying the number of cycles of the rainfall events C_{ub} and the number of observational locations. Each setting was tested with three different neural network initializations. (**a**): L2 error for the simulated volumetric water content. (**b**): L2 error for the water retention curve. (**c**): L2 error for the unsaturated permeability function in log scale.

Figure 8 shows the effect of the amount of data on the performance of the neural network approach. For the simulated volumetric water content (Figure 8 (**a**)) and the water retention curve (Figure 8 (**b**)), the performance consistently improved with the amount of data (i.e., higher C_{ub} and the number of observational locations). However, we observed the opposite trend for the unsaturated permeability function (Figure 8 (**c**)). We note that we obtained a similar opposite trend for the effect of the magnitude of the noise (Figure 7 (**c**)). Nevertheless, in all cases, the recovered relative permeability functions were reasonably accurate.

3.5 Application to upward infiltration experimental data

Lastly, we applied the inverse modeling framework to the upward infiltration experiment data conducted by Sadeghi et al. (2017). These investigators packed seven different oven-dried soils into a rectangular box and induced upward infiltration by connecting the bottom of the box to a constant hydraulic head device. They used a shortwave infrared imaging camera to estimate soil moisture based on a physics-based model (Sadeghi et al., 2015). We used the soil moisture data for sandy loam soil (AZ7 soil) to demonstrate the feasibility of our inverse modeling framework against real experimental data. We selected AZ7 soil because the experimental data show homogeneous upward infiltration and are compatible with the one-dimensional Richardson-Richards equation. Further details of the experiments can be found in Sadeghi et al. (2017) and Bandai, Sadeghi, et al. (2024).

To extract the constitutive relations from the shortwave infrared based volumetric water content data, we defined a one-dimensional soil column with a length $Z = 0.1$ m, which was uniformly discretized into 100 cells. We used a uniform initial condition $\psi = -10^4$ m because the soil was initially equilibrated with the laboratory atmosphere. We set the lower boundary condition to $\psi_{lb} = -10^{-4}$ m, while we used a zero water flux condition for the upper boundary condition because we only used the soil moisture data until the water reached the top boundary. Although the top boundary was open to the atmosphere, we assumed that the evaporation from the top boundary was negligible because the soil was equilibrated with the laboratory air. A constant time stepping $\Delta t = 1$ min was used. We used the soil moisture data collected at eight depths ($z = -0.085, -0.075, -0.065, -0.055, -0.045, -0.035, -0.025, -0.015$ m) as observational data.

We compared the performance of our neural network based approach with that of the van Genuchten-Mualem model. As for the neural network approach, we set $\mu_{\theta_s} = 0.5$, $\sigma_{\theta_s} = 0.2$, $\mu_{\log_{10} K} = -13.0$, and $\sigma_{\log_{10} K} = 3.0$. The number of units in the hidden layer was set to 40, and three different random seeds for neural network initializations were used. As for the van Genuchten-Mualem model, there are six parameters: θ_s, θ_r, α, n, τ, and K. We fixed τ to $\tau = 0.5$. We used the same transformation for θ_s and K as the neural network based approach. For θ_r, α, and n, we used the following transformations

$$\theta_r = \mu_{\theta_r} + \sigma_{\theta_r} \tanh \theta_r^t, \tag{32}$$
$$\log_{10} \alpha = \mu_{\log_{10} \alpha} + \sigma_{\log_{10} \alpha} \tanh \alpha^t, \tag{33}$$
$$\log_{10} n = \mu_{\log_{10} n} + \sigma_{\log_{10} n} \tanh n^t, \tag{34}$$

where μ_{θ_r} is the mean for the θ_r parameter, $\mu_{\log_{10} \alpha}$ and $\mu_{\log_{10} n}$ are the mean for the van Genuchten parameters α and n in log scale, respectively, σ_{θ_r} is the half of the range of the θ_r parameter, $\sigma_{\log_{10} \alpha}$ and $\sigma_{\log_{10} n}$ are the half of the range for the van Genuchten parameters α and n in log scale, respectively, θ_r^t, α^t, and n^t are the transformed parameters. We used $\mu_{\theta_r} = 0.05$, $\mu_{\log_{10} \alpha} = -0.5$, $\mu_{\log_{10} n} = 0.5$, $\sigma_{\theta_r} = 0.05$, $\sigma_{\log_{10} \alpha} = 2.5$ and $\sigma_{\log_{10} n} = 0.5$. Thus, the set of parameters estimated in the inverse modeling framework with the van Genuchten-Mualem model is $\Theta = \{\theta_r^t, \theta_s^t, \alpha^t, n^t, K^t\}$. Because we do not know the measurement error of the soil moisture data, we set σ_θ^i to 0.01 in the loss function (Equation 22). We ran the Adam optimizer with the default setting 10,000 times to minimize the loss function defined for each approach. Although it is known that the van Genuchtne-Mualem model may suffer a local minimum of the loss function, we used the gradient based method for fair comparison.

Figure 9 shows the fitted numerical solutions to the soil moisture data from the upward infiltration experiment for AZ7 soil. The neural network based approach was better fitted to the experimental data than the van Genuchten-Mualem model, particularly for dry and wet regimes. The minimized loss function $\mathcal{L}$ (Equation 22) was 0.854 and 2.95 for the neural network approach and the van Genuchten-Mualem model, respectively. This result demonstrates the strength of the neural network based approach to fit to the data compared to parametric models with few numbers of free parameters.

Figure 10 shows the estimated constitutive relations from the upward infiltration experiment for AZ7 soil. As for the water retention curve (Figure 10 (**a**)), the neural net-

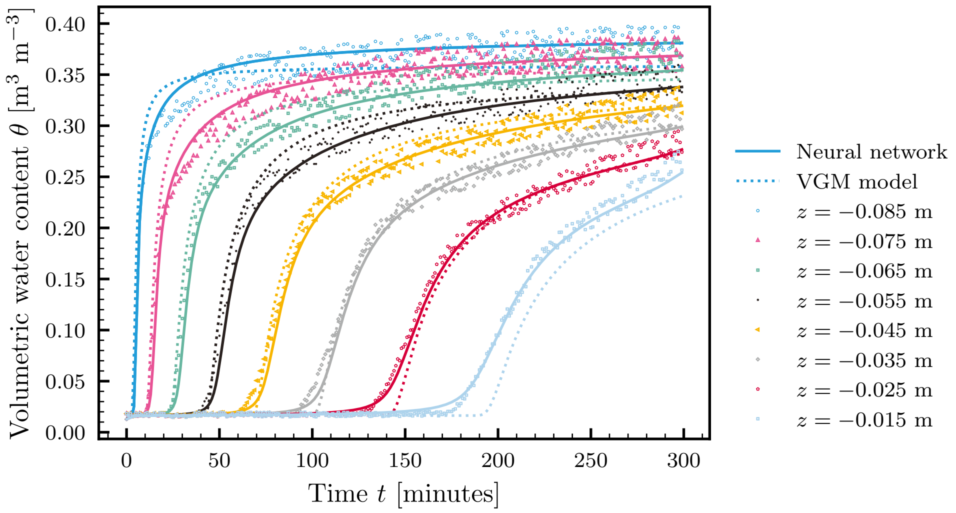

Figure 9. Fitted numerical solutions to the upward infiltration experimental data for AZ7 soil. In the numerical solutions, the constitutive relations are represented by physically constrained neural networks (Neural network) or the van Genuchten-Mualem model (VGM model).

work model estimated higher volumetric water content θ given the same water potential ψ and the slope of the water retention curve was more gradual than the van Genuchten-Mualem model, which led to the gradual increase in the volumetric water content gradually for dry and wet regimes for the neural network model. We plotted the experimental data on the drying branch of the water retention curve for AZ7 soil measured by Tempe cells (Soilmoisture Equipment Corp., USA) and WP4-T Dewpoint Potentiameter (METER Group, Inc., USA). While the inverse modeling from the upward infiltration data extracts the wetting branch of the water retention curve, the estimated water retention curve by the neural network approach matched the experimental water retention data well from saturated to medium moisture conditions. On the other hand, there is a discrepancy between the estimated and measured water retention curve for dry conditions. This is due to the limited amount of information content in the soil moisture data for dry conditions relative to the number of neural network parameters. While there is plenty of data at near-zero water content, there is almost no change in the data, and thus, the information content is low. We might be able to deal with such low information content for dry conditions by adding more physical constraints on the dry end, such as the minimum water potential and the slope of the water retention curve (Tokunaga, 2009). In terms of the unsaturated permeability function, Figure 10 (**b**) shows higher unsaturated permeability for dry conditions for the neural network model. This is reasonable because water molecules are held by soil mineral surfaces as thin films and continue to flow for dry conditions (Tuller et al., 1999).

To further clarify the ability of neural networks to learn the constitutive relations, we plotted the relation between the relative permeability and the saturation defined by $\frac{\theta}{\theta_s}$ in Figure 10 (**c**). The van Genuchten-Mualem model in the plot corresponds to Mualem's bundle tube model (Equation 10). The result shows that the neural network model qualitatively agrees with the van Genuchten-Mualem model for intermediate moisture conditions, suggesting that the neural network learned Mualem's bundle tube model. On the other hand, in the dry and the wet regimes, we observed a discrepancy between the two models, which demonstrates that Mualem's bundle tube model is not adequate to describe the soil moisture dynamics for wet and dry conditions.

4 Discussion

Our work takes a similar approach to that of Bitterlich et al. (2004) and Iden and Durner (2007), where they proposed free-form approaches to estimate the constitutive relations of the Richardson-Richards equation (i.e., soil hydraulic functions). Compared to their studies, however, our inverse modeling framework is scalable and extensible. First, our inverse modeling framework can be extended to more complex problems (i.e., multi-physics problems) with scalable development costs. Bitterlich et al. (2004) used adjoint methods to compute the gradient of a loss function to solve their constrained optimization problem. While adjoint methods give exact derivatives as in automatic differentiation, the implementation of adjoint methods is problem-dependent, extremely tedious, and error-prone for complex problems. This is especially true for the Richardson-Richards equation, a time-dependent nonlinear partial differential equation with nonlinear constitutive relations (Bandai, 2022). Although the automatic implementation of adjoint methods exists (Mitusch et al., 2019), its application to complex problems is not straightforward and has not yet been achieved. Our framework uses reverse-mode automatic differentiation with implicit differentiation, which is problem-independent and thus can be extensible to more complex problems. Furthermore, the previous free-form approaches (Bitterlich et al., 2004; Iden & Durner, 2007) require specific optimization algorithms to satisfy the monotonicity constraint for the soil hydraulic functions. On the other hand, our monotonic neural network approach automatically satisfies the monotonicity constraint and can use any optimization algorithms available through machine learning libraries, thus reducing the development costs. Second, our framework is scalable with respect to the number of parameters. In vadose zone studies, including that of Iden and Durner (2007), global optimization methods were preferred because it was claimed that gradient-based local minimization algorithms suffer from a bad local minimum in objective functions (Vrugt et al., 2008). While global optimization methods are powerful for small scale problems (e.g., up to 50 parameters), they are not scalable with respect to the number of parameters. In contrast, our inverse modeling framework uses a gradient-based optimization algorithm with reverse-mode automatic differentiation, which is scalable and thus can incorporate highly parameterized functions, such as neural networks. It appears that local optimization algorithms do not suffer from a bad local minimum when soil hydraulic functions are parameterized by neural networks, as we demonstrated in Section 3.4. While it is widely believed that over-parameterized neural networks do not suffer from bad local minimum (Belkin, 2021), it is necessary to investigate whether such arguments are true for our application.

Next, we discuss how our work is related to other physics-informed machine learning and differentiable hybrid modeling approaches (Karniadakis et al., 2021; Shen et al., 2023). In these approaches, machine learning models (i.e., neural networks) are combined with physics-based models through differentiable modeling platforms, such as TensorFlow (Abadi et al., 2015), PyTorch (Paszke et al., 2019), and JAX (Bradbury et al., 2018), all of which support automatic differentiation (Baydin et al., 2018). Physics-based models are embedded in different ways, including soft or hard constraints in a loss function (Bandai & Ghezzehei, 2021; Lu et al., 2021; Wang et al., 2023) and structural priors (Mitusch et al., 2021; Feng et al., 2023; Gaskin et al., 2023). Our approach enforced the Richardson-Richards equation as a structural prior and is categorized in the latter group. Unlike enforcing physical models as a soft constraint in a loss function as in physics-informed neural network approaches (Bandai & Ghezzehei, 2021), our approach strictly enforces the physical laws in the Richardson-Richards equation, the conservation of mass (Equation 1) and the Buckingham-Darcy law (Equation 2). It is notable that we used neural networks to represent unknown functions in a physics-based model (i.e., soil hydraulic func-

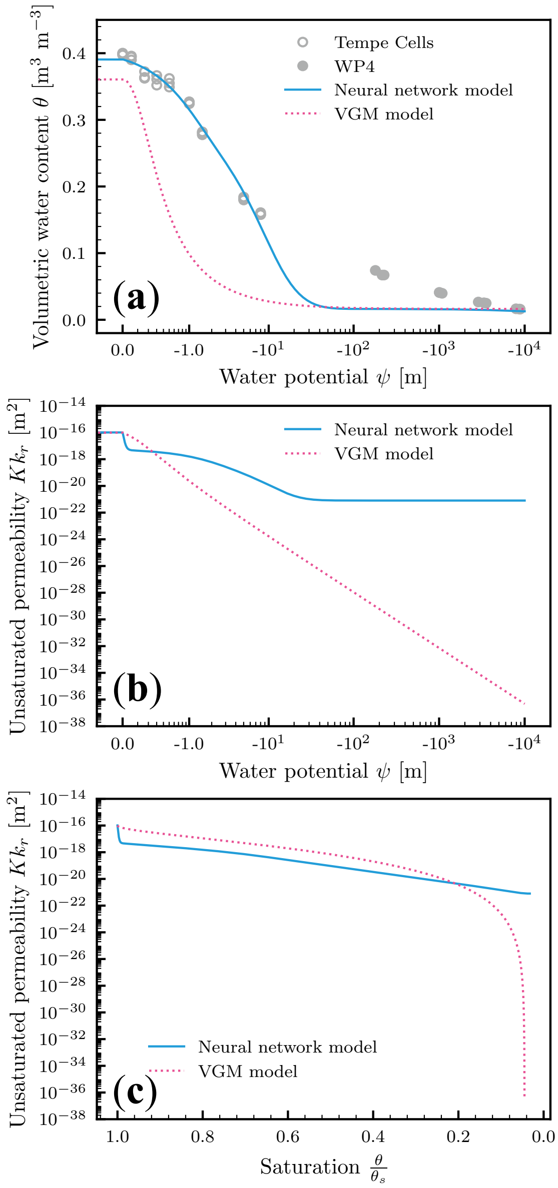

Figure 10. The estimated constitutive relations from the upward infiltration experimental data for AZ7 soil using the physically constrained neural network (Neural network model) and the van Genuchten-Mualem (VGM model). (**a**): The estimated water retention curve. The open and closed circles represent the drying branch of the water retention measurement of AZ7 soil using Tempe cells and WP-4T Dewpoint Potentiameter, respectively. (**b**): The estimated unsaturated permeability function. (**c**): The estimated unsaturated permeability with respect to the saturation.

tions) as in Mitusch et al. (2021), while other studies used neural networks to estimate physical parameters used in embedded physical models, as in Feng et al. (2023) and Gaskin et al. (2023). The former approach can be classified as a genre of universal differential equations (Rackauckas et al., 2021) and neural differentiation equations (Kidger, 2022). While we did not use neural networks to infer physical parameters, as in Feng et al. (2023) and Gaskin et al. (2023), such approaches might be more robust than directly training the physical parameters (the transformed volumetric water content θ_s^t and permeability K^t) via gradient-based algorithms, as we did.

We must mention the limitations of the current inverse modeling framework. The fundamental limitation is the high computational cost to solve the forward problem and the optimization problem. As for the forward modeling of the Richardson-Richards equation, the computational performance might be improved with more efficient discretization methods, linear solvers with appropriate preconditioners, and nonlinear solvers (Farthing & Ogden, 2017). Also, implementing parallel computations across multiple GPU cores is a promising approach. To solve the optimization problem more efficiently, we should test second-order Newton methods (Isaac et al., 2015; Ghattas & Willcox, 2021; Bandai, 2022). Another limitation is the ability to handle complicated boundary conditions, such as ponding on the soil surface and seepage boundaries. These system-dependent boundary conditions are unique to vadose zone studies, and we need to test the differentiable modeling framework against such situations for practical applications. Finally, the current inverse modeling framework is focused on deterministic parameter estimation, and the uncertainty of the estimated parameters has not formally been investigated. For interested readers, we provide information on the uncertainty of the estimated parameters obtained by synthetic data with different noise in the supplementary material, where we demonstrated that the estimated soil hydraulic functions by the physically constrained neural networks were quite consistent regardless of different noises in the data. We note that uncertainty quantification of neural networks is challenging because the number of parameters is large, which prevents us from employing Markov Chain Monte Carlo approaches. In this context, future studies should quantify the uncertainty of parameters via scalable approaches, such as the Laplace approximation with a low-rank approximation of the Hessian using efficient Hessian-vector product computations (Ghattas & Willcox, 2021).

We note future opportunities of the inverse modeling framework. We envision that our inverse modeling framework can be extended to more complex problems. For example, it would be interesting to apply the current framework to lysimeter data to recover surface water flux and estimate the constitutive relations for water and heat transport problems. Also, it might be possible to extract the constitutive relations from satellite-based soil moisture products using the current framework. Finally, we mention how to advance our understanding of the constitutive relations from the learned neural networks. One possibility is to use symbolic regression to extract parametric equations from the trained neural networks (Kidger, 2022). Another possible approach is to compare the trained neural networks with the constitutive relations predicted from pore-scale physical models (Tuller et al., 1999; Tuller & Or, 2001). In both cases, our top-down data-driven neural network approach provides valuable information on the constitutive relations of the Richardson-Richards equation on various scales of interest.

5 Conclusions

We developed a new free-form approach to extract the constitutive relations of the Richardson-Richards equation from soil moisture data via inverse modeling. We built the inverse modeling framework on a machine learning framework called JAX. We tested the inverse modeling framework against synthetic noisy data and soil moisture data from an upward infiltration experiment. The main conclusions of the study are as follows:

1. JAX on a GPU was competitively fast to solve forward problems.
2. Implicit differentiation through the Newton solver enabled a scalable algorithm with respect to the number of neural network parameters.
3. Synthetic numerical experiments demonstrated the robustness of the framework against the noise in the data, the initialization of the neural networks, and the amount of available data.
4. We demonstrated that the neural networks successfully extracted more information on the constitutive relations from the upward infiltration experimental data compared to the commonly used parametric models.

We also discussed the perspectives of the differentiable modeling approach employed in the study. We envision that this framework can be extended to other multi-physics problems in vadose zone hydrology.

Appendix A Numerical method

We solved the Richardson-Richards equation with the initial and boundary conditions by a finite volume method with the Backward Euler method. For the one-dimensional problem, the spatial domain was divided into N_s cells, where a cell C_i for $i = 1, ..., N_s$ corresponds to $z \in [z_{i-1/2}, z_{i+1/2}]$ with the grid space $\Delta z_i = z_{i+1/2} - z_{i-1/2}$. To accommodate various boundary conditions, we placed the ghost cell next to each spatial boundary, $C_0 := \{z \in [-Z - \Delta z_1, -Z]\}$ and $C_{N_s+1} := \{z \in [0, \Delta z_{N_s}]\}$. The soil hydraulic properties of the ghost cells are assumed to be the same those of the cell next to the ghost cells. Over each cell, we assumed that the water potential is constant, and the volumetric water content and relative permeability are computed by given water retention curve and relative permeability function from the cell-centered water potential ψ_i for the cell C_i for $i = 0, ..., N_s + 1$.

We integrated the mass balance equation (Equation 1) over a cell C_i for $i = 1, ..., N_s$ and obtained

$$\int_{C_i} \frac{\partial \theta}{\partial t}\, dz = q(z_{i-1/2}, t) - q(z_{i+1/2}, t). \tag{A1}$$

The temporal derivative is approximated by the Backward Euler method. For that purpose, we introduce the times t^n for $n = 0, 1, ..., N_t$, such that

$$0 = t^0 < t^1 \cdots < t^n < \cdots < t^{N_t} = T, \tag{A2}$$

and the corresponding time steps $\Delta t^n = t^n - t^{n-1}$ for $n = 1, ..., N_t$. We represent the solution at the time $t = t^n$ by a bold symbol with the superscript n, such as $\boldsymbol{\psi}^n = [\psi_0^n, \psi_1^n, ..., \psi_{N_s}^n, \psi_{N_s+1}^n]^T$ and $\boldsymbol{\theta}^n = [\theta_0^n, \theta_1^n, ..., \theta_{N_s}^n, \theta_{N_s+1}^n]^T$. The initial condition was interpolated to obtain $\boldsymbol{\psi}^0$. For the time $t = t^n$ with $n = 1, ..., N_t$, we obtain the following non-linear equation for $i = 1, ..., N_s$:

$$F_i^n := (\theta_i^n - \theta_i^{n-1}) - \frac{\Delta t^n}{\Delta z_i} \left(q(z_{i-1/2}, t^n) - q(z_{i+1/2}, t^n) \right). \tag{A3}$$

Here, the water flux at the interface is evaluated as follows:

$$q(z_{i-1/2}, t^n) = -\frac{\hat{K}_{i-1/2} \hat{k}_r^n(z_{i-1/2}, t^n) \rho g}{\mu} \left(\frac{h_i - h_{i-1}}{z_i - z_{i-1}} \right), \tag{A4}$$

where the permeability at the internal interface is computed by an inverse distance-weighted harmonic mean:

$$\hat{K}_{i-1/2} = \frac{K_{i-1} K_i (z_i - z_{i-1})}{K_{i-1} \Delta z_i / 2 + K_i \Delta z_{i-1} / 2}, \tag{A5}$$

and the relative permeability is upwinded according to

$$\hat{k}_r^n(z_{i-1/2}, t^n) = k_r^n(z_{i-1}, t^n) \quad \text{if} \quad h_{i-1}^n > h_i^n, \tag{A6}$$

$$\hat{k}_r^n(z_{i-1/2}, t^n) = k_r^n(z_i, t^n) \quad \text{if} \quad h_i^n > h_{i-1}^n. \tag{A7}$$

We obtain additional non-linear equations corresponding to the lower and upper boundary condition. For the lower boundary,

$$F_0^n := \psi_{1/2}^n - \psi_{lb}, \tag{A8}$$

and for the upper boundary,

$$F_{N_s+1}^n := q(z_{N_s+1/2}, t^n) - q_{ub}. \tag{A9}$$

The interface water potential is computed by a distance-weighted arithmetic mean:

$$\psi_{1/2}^n = \frac{\psi_0^n \Delta z_1 + \psi_1^n \Delta z_0}{z_1 - z_0}. \tag{A10}$$

We assemble the non-linear equations as $\mathbf{F}^n = [F_0^n, F_1^n, ..., F_{N_s}^n, F_{N_s+1}^n]^T$ for $n = 1, 2, ..., N_t$. The system of non-linear equations $\mathbf{F}^n$ was solved by the Newton method. Let $\boldsymbol{\psi}^{n,k}$ denote the solution at the kth Newton iteration for $k = 0, 1, 2, ...$ with $\boldsymbol{\psi}^{n,0} = \boldsymbol{\psi}^{n-1}$. For the kth Newton iteration, the Newton direction $\mathbf{d}^k$ is determined by solving the Newton system:

$$\mathbf{F}'(\boldsymbol{\psi}^{n,k})\mathbf{d}^k = -\mathbf{F}(\boldsymbol{\psi}^{n,k}), \tag{A11}$$

where the Jacobian matrix $\mathbf{F}' := \frac{\partial \mathbf{F}}{\partial \boldsymbol{\psi}}$ was computed by analytically or automatic differentiation. The Newton step size is adjusted by Armijo line search:

$$\boldsymbol{\psi}^{n,k+1} = \boldsymbol{\psi}^{n,k} + \lambda \mathbf{d}^k, \tag{A12}$$

with the initial $\lambda = 1$, and λ is reduced by a factor 0.5 when the sufficient decrease condition

$$||\mathbf{F}(\boldsymbol{\psi}^{n,k+1})|| < (1 - c\lambda)||\mathbf{F}(\boldsymbol{\psi}^{n,k})||, \tag{A13}$$

where $c = 10^{-4}$, is not met. Here, $|| \cdot ||$ represents the L^∞ norm. The Newton iteration was terminated when

$$||\mathbf{F}(\boldsymbol{\psi}^{n,k})|| \leq \tau_a, \tag{A14}$$

where $\tau_a = 10^{-8}$.

Open Research Section

The Python and Fortran codes and computational results are available on Bandai, Ghezzehei, et al. (2024). The upward infiltration experimental data is available on Bandai et al. (2023).

Acknowledgments

This work was funded by the ExaSheds project, which was supported by the U.S. Department of Energy, Office of Science, Office of Biological and Environmental Research, Earth and Environmental Systems Sciences Division, Data Management Program, under award no. DE-AC02-05CH11231. Toshiyuki Bandai was additionally supported by an EESA Early Career Development Grant 2024 from the Lawrence Berkeley National Laboratory. This research used computational resources of the National Energy Research Scientific Computing Center (NERSC), a Department Energy Office of Science User Facility. We are indebted to Dr. Morteza Sadeghi, Dr. Scott B. Jones, and Dr. Markus Tuller for sharing the experimental data used in the study. We thank Dr. Ebrahim Babaeian for reviewing the earlier version of the manuscript. We appreciate the JAX team at Google for developing and maintaining the JAX software and answering the authors' questions. We are indebted to the Associate Editor Dr. Hannes Bauser, and the two anonymous reviewers for improving this manuscript in the peer review process.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous distributed systems.* Retrieved from https://www.tensorflow.org/

Bandai, T. (2022). *Inverse modeling of soil moisture dynamics: Estimation of soil hydraulic properties and surface water flux* (Unpublished doctoral dissertation). University of California, Merced.

Bandai, T., & Ghezzehei, T. A. (2021). Physics-informed neural networks with monotonicity constraints for Richardson-Richards equation: Estimation of constitutive relationships and soil water flux density from volumetric water content measurements. *Water Resour. Res.*, *57*. doi: 10.1029/2020WR027642

Bandai, T., & Ghezzehei, T. A. (2022). Forward and inverse modeling of water flow in unsaturated soils with discontinuous hydraulic conductivities using physics-informed neural networks with domain decomposition. *Hydrol. Earth Syst. Sci.*, *26*(16), 4469–4495. doi: 10.5194/hess-26-4469-2022

Bandai, T., Ghezzehei, T. A., Jiang, P., Kidger, P., Chen, X., & Steefel, C. I. (2024). *Dataset for Learning constitutive relations from soil moisture data via physically constrained neural networks.* Zenodo [data set]. doi: 10.5281/zenodo.11248011

Bandai, T., Sadeghi, M., Babaeian, E., Jones, B. S., Tuller, M., & Ghezzehei, T. A. (2023). *Dataset for Estimating soil hydraulic properties from oven-dry to full saturation using inverse modeling and shortwave infrared imaging.* Zenodo [data set]. doi: 10.5281/zenodo.8303144

Bandai, T., Sadeghi, M., Babaeian, E., Jones, S. B., Tuller, M., & Ghezzehei, T. A. (2024). Estimating soil hydraulic properties from oven-dry to full saturation using shortwave infrared imaging and inverse modeling. *J. Hydrol.*, *635*(October 2023), 131132. Retrieved from https://doi.org/10.1016/j.jhydrol.2024.131132 doi: 10.1016/j.jhydrol.2024.131132

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, *18*, 1–43. doi: 10.1016/j.advwatres.2018.01.009

Belkin, M. (2021). Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. *Acta Numer.*, *30*(2021), 203–248. doi: 10.1017/S0962492921000039

Bezgin, D. A., Buhendwa, A. B., & Adams, N. A. (2023). JAX-Fluids: A fully-differentiable high-order computational fluid dynamics solver for compressible two-phase flows. *Comput. Phys. Commun.*, *282*, 108527. Retrieved from https://doi.org/10.1016/j.cpc.2022.108527 doi: 10.1016/j.cpc.2022.108527

Bitterlich, S., Durner, W., Iden, S. C., & Knabner, P. (2004). Inverse estimation of the unsaturated soil hydraulic properties from column outflow experiments using free-form parameterizations. *Vadose Zone J.*, *3*(3), 971–981. doi: 10.2113/3.3.971

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., ... Zhang, Q. (2018). *JAX: composable transformations of Python+NumPy programs.* Retrieved from http://github.com/google/jax

Brooks, R. H., & Corey, A. T. (1964). Hydraulic properties of porous media. *Hydrol. Pap. 3, Color. State Univ.*.

Buckingham, E. (1907). *Studies on the movement of soil moisture* (Vol. Bull. 38; Tech. Rep.). Washington, DC: USDA, Bureau of Soils.

Burdine, N. T. (1953). Relative permeability calculations from pore size distribution data. *Soc. Pet. Eng.*, *5*(3). doi: 10.2118/225-G

Chou, H. H., Gieshoff, C., Maly, J., & Rauhut, H. (2024). Gradient descent for deep matrix factorization: Dynamics and implicit bias towards low rank. *Appl.*

Comput. Harmon. Anal., *68*, 101595. Retrieved from https://doi.org/10.1016/j.acha.2023.101595 doi: 10.1016/j.acha.2023.101595

Coon, E. T., Berndt, M., Jan, A., Svyatsky, D., Atchley, A. L., Kikinzon, E., ... Molins, S. (2020). *Advanced Terrestrial Simulator.* U.S. Department of Energy, USA. doi: https://doi.org/10.11578/dc.20190911.1

Daniels, H., & Velikova, M. (2010). Monotone and partially monotone neural networks. *IEEE Trans. Neural Networks*, *21*(6), 906–917. doi: 10.1109/TNN.2010.2044803

DeepMind, Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., ... Viola, F. (2020). *The DeepMind JAX Ecosystem.* Retrieved from http://github.com/google-deepmind

Farthing, M. W., & Ogden, F. L. (2017). Numerical solution of Richards' equation: A review of advances and challenges. *Soil Sci. Soc. Am. J.*, *81*, 1257–1269. doi: 10.2136/sssaj2017.02.0058

Feng, D., Beck, H., Lawson, K., & Shen, C. (2023). The suitability of differentiable, physics-informed machine learning hydrologic models for ungauged regions and climate change impact assessment. *Hydrol. Earth Syst. Sci.*, *27*(12), 2357–2373. doi: 10.5194/hess-27-2357-2023

Forsyth, P. A., Wu, Y. S., & Pruess, K. (1995). Robust numerical methods for saturated-unsaturated flow with dry initial conditions in heterogeneous media. *Adv. Water Resour.*, *18*(1), 25–38. doi: 10.1016/0309-1708(95)00020-J

Frostig, R., Johnson, M. J., & Leary, C. (2018). Compiling machine learning programs via high-level tracing. In *Sysml conf. 2018.*

Gaskin, T., Pavliotis, G. A., & Girolami, M. (2023). Neural parameter calibration for large-scale multiagent models. *Proc. Natl. Acad. Sci. U. S. A.*, *120*(7), 1–10. doi: 10.1073/pnas.2216415120

Ghattas, O., & Willcox, K. (2021). Learning physics-based models from data: perspectives from inverse problems and model reduction. *Acta Numer.*, *30*, 445–554. doi: 10.1017/S0962492921000064

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proc. thirteen. int. conf. artif. intell. stat.* (pp. 249–256). Sardinia, Italy, May 13 - 15.

Griewank, A., & Walther, A. (2008). *Evaluating Derivatives: Principles and Techniques ofAlgorithmic Differentiation, second edition, Society for Industrial and Applied Math- ematics (SIAM).* (Second ed.). Society for Industrial and Applied Math- ematics (SIAM).

Häfner, D., Nuterman, R., & Jochum, M. (2021). Fast, Cheap, and Turbulent—Global Ocean Modeling With GPU Acceleration in Python. *J. Adv. Model. Earth Syst.*, *13*, e2021MS002717. doi: 10.1029/2021MS002717

Hammond, G. E., Lichtner, P. C., & Mills, R. T. (2014). Evaluating the performance of parallel subsurface simulators : An illustrative example with PFLOTRAN. *Water Resour. Res.*, *50*, 208–228. doi: 10.1002/2012WR013483

He, X. (2023). Accelerated linear algebra compiler for computationally efficient numerical models: Success and potential area of improvement. *PLoS One*, *18*(2 February), 1–38. Retrieved from http://dx.doi.org/10.1371/journal.pone.0282265 doi: 10.1371/journal.pone.0282265

Hückelheim, J., Menon, H., Moses, W., Christianson, B., Havland, P., & Hascoët, L. (2023). Understanding Automatic Differentiation Pitfalls. *arXiv*.

Iden, S. C., & Durner, W. (2007). Free-form estimation of the unsaturated soil hydraulic properties by inverse modeling using global optimization. *Water Resour. Res.*, *43*, 1–12. doi: 10.1029/2006WR005845

Isaac, T., Petra, N., Stadler, G., & Ghattas, O. (2015). Scalable and efficient algorithms for the propagation of uncertainty from data through inference to prediction for large-scale problems, with application to flow of the Antarctic ice sheet. *J. Comput. Phys.*, *296*, 348–368. Retrieved from http://dx.doi.org/

10.1016/j.jcp.2015.04.047 doi: 10.1016/j.jcp.2015.04.047

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

Kidger, P. (2022). *On Neural Differential Equations* (Doctoral dissertation, University of Oxford). Retrieved from http://arxiv.org/abs/2202.02435

Kidger, P., & Garcia, C. (2021). Equinox: neural networks in JAX via callable PyTrees and filtered transformations. In *35h conf. neural inf. process. syst. (neurips 2021).* Retrieved from http://arxiv.org/abs/2111.00254

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., & Hoyer, S. (2021). Machine Learning Computational Fluid Dynamics. *Proc. Natl. Acad. Sci.*, *118*(21), 1–8. doi: 10.1109/SAIS53221.2021.9483997

Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., & Johnson, S. G. (2021). Physics-Informed Neural Networks With Hard Constraints for Inverse Design. *SIAM J. Sci. Comput.*, *43*(6), B1105–B1132. doi: 10.1137/21M1397908

Miller, C. T., Christakos, G., Imhoff, P., McBride, J. F., & Pedit, J. A. (1998). Multiphase flow and transport modeling in heterogeneous porous media: Challenges and approaches. *Adv. Water Resour.*, *21*(2), 77–120. doi: 10.1016/S0309-1708(96)00036-X

Mitusch, S. K., Funke, S. W., & Dokken, J. S. (2019). dolfin-adjoint 2018.1: automated adjoints for FEniCS and Firedrake. *J. Open Source Sortware*, *4*(38), 1292. doi: 10.1145/2566630

Mitusch, S. K., Funke, S. W., & Kuchta, M. (2021). Hybrid FEM-NN models: Combining artificial neural networks with the finite element method. *J. Comput. Phys.*, *446*, 110651. Retrieved from https://doi.org/10.1016/j.jcp.2021.110651 doi: 10.1016/j.jcp.2021.110651

Mualem, Y. (1976). A new model for predicting the hydraulic conductivity of unsaturated porous media. *Water Resour. Res.*, *12*(3), 513–522. doi: 10.1029/WR012i003p00513

Naseri, M., Joshi, D. C., Iden, S. C., & Durner, W. (2023). Rock fragments influence the water retention and hydraulic conductivity of soils. *Vadose Zone J.*, *22*, e20243. doi: 10.1002/vzj2.20243

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *33rd conf. neural inf. process. syst.* Vancouver, Canada, December 8 - 14.

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., ... Edelman, A. (2021). Universal Differential Equations for Scientific Machine Learning. *arXiv*, 1–55. Retrieved from http://arxiv.org/abs/2001.04385

Rader, J., Lyons, T., & Kidger, P. (2023). Lineax : unified linear solves and linear least-squares in JAX and Equinox. *arXiv*, 1–8.

Radul, A., Paszke, A., Frostig, R., Johnson, M. J., & Maclaurin, D. (2023). You only Linearize Once: Tangents Transpose to Gradients. *Proc. ACM Program. Lang.*, *7*, 1246–1274. doi: 10.1145/3571236

Richards, L. A. (1931). Capillary conduction of liquids through porous mediums. *J. Appl. Phys.*, *1*, 318–333. doi: 10.1063/1.1745010

Richardson, L. F. (1922). *Weather prediction by numerical process.* Cambridge, United Kingdom: Cambridge University Press. doi: 10.1017/CBO9780511618291

Sadeghi, M., Jones, S. B., & Philpot, W. D. (2015). A linear physically-based model for remote sensing of soil moisture using short wave infrared bands. *Remote Sens. Environ.*, *164*, 66–76. doi: 10.1016/j.rse.2015.04.007

Sadeghi, M., Sheng, W., Babaeian, E., Tuller, M., & Jones, S. B. (2017). High-resolution shortwave infrared imaging of water infiltration into dry soil. *Vadose Zone J.*, *16*(13). doi: 10.2136/vzj2017.09.0167

Schoenholz, S. S., & Cubuk, E. D. (2020). JAX, M.D. A framework for differentiable physics. In *34th conf. neural inf. process. syst. (neurips 2020).* doi: 10.1088/1742-5468/ac3ae9

Shen, C., Appling, A. P., Gentine, P., Bandai, T., Gupta, H., Tartakovsky, A., ... Lawson, K. (2023, jul). Differentiable modelling to unify machine learning and physical models for geosciences. *Nat. Rev. Earth & Environ.*. Retrieved from https://www.nature.com/articles/s43017-023-00450-9 doi: 10.1038/s43017-023-00450-9

Šimůnek, J., van Genuchten, M., & Šejna, M. (2016). Recent developments and applications of the HYDRUS computer software packages. *Vadose Zone J.*, 1–25. doi: 10.2136/vzj2016.04.0033

Steefel, C. I., Appelo, C. A. J., Arora, B., Jacques, D., Kalbacher, T., Kolditz, O., ... Yeh, G. T. (2015). Reactive transport codes for subsurface environmental simulation. *Comput. Geosci.*, *19*, 445–478. doi: 10.1007/s10596-014-9443-x

Tokunaga, T. K. (2009). Hydraulic properties of adsorbed water films in unsaturated porous media. *Water Resour. Res.*, *45*, W06415. doi: 10.1029/2009WR007734

Tuller, M., Dani, O., & Dudley, L. M. (1999). Adsorption and capillary condensation in porous media: Liquid retention and interfacial configurations in angular pores. *Water Resour. Res.*, *35*(7), 1949–1964. doi: 10.1029/1999WR900098

Tuller, M., & Or, D. (2001). Hydraulic conductivity of variably saturated porous media: Film and corner flow in angular pore space. *Water Resour. Res.*, *37*(5), 1257–1276. doi: 10.1029/2000WR900328

van Genuchten, M. T. (1980). A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Sci. Soc. Am.*, *44*, 892–898. doi: 10.1016/j.pan.2017.07.214

Vogel, H. J., Gerke, H. H., Mietrach, R., Zahl, R., & Wöhling, T. (2023). Soil hydraulic conductivity in the state of nonequilibrium. *Vadose Zone J.*, *22*, e20238. doi: 10.1002/vzj2.20238

Vogel, T., Huang, K., Zhang, R., & van Genuchten, M. T. (1996). *Hydrus code for simulating one-dimensional water flow, solute transport, and heat movement in variably-saturated media: Version 5.0.* (Tech. Rep.). Riverside, CA, USA: U.S. Salinity Laboratory, Agricultural Research Service, USDA. doi: 10.13140/RG.2.1.3456.7525

Vogelmann, E. S., Reichert, J. M., Prevedello, J., Consensa, C. O., Oliveira, A. É., Awe, G. O., & Mataix-Solera, J. (2013). Threshold water content beyond which hydrophobic soils become hydrophilic: The role of soil texture and organic matter content. *Geoderma*, *209-210*, 177–187. Retrieved from http://dx.doi.org/10.1016/j.geoderma.2013.06.019 doi: 10.1016/j.geoderma.2013.06.019

Vrugt, J. A., Stauffer, P. H., Wöhling, T., Robinson, B. A., & Vesselinov, V. V. (2008). Inverse modeling of subsurface flow and transport properties: A review with new developments. *Vadose Zone J.*, *7*, 843–864. doi: 10.2136/vzj2007.0078

Wang, Y., Wang, W., Ma, Z., Zhao, M., Li, W., Hou, X., ... Ma, W. (2023). A Deep Learning Approach Based on Physical Constraints for Predicting Soil Moisture in Unsaturated Zones. *Water Resour. Res.*, *59*(11). doi: 10.1029/2023WR035194

Xue, T., Liao, S., Gan, Z., Park, C., Xie, X., Liu, W. K., & Cao, J. (2023). JAX-FEM: A differentiable GPU-accelerated 3D finite element solver for automatic inverse design and mechanistic data science. *Comput. Phys. Commun.*, *291*, 108802. Retrieved from https://doi.org/10.1016/j.cpc.2023.108802 doi: 10.1016/j.cpc.2023.108802