Deep neural operators for inversion and surrogate modeling in unsaturated flows

Hamza Kamil[a,b], Azzeddine Soulaïmani[a], Abdelaziz Beljadid[b,c]

[a] École de Technologie Supérieure, Canada.

[b] Mohammed VI Polytechnic University, Morocco.

[c] University of Ottawa, Canada.

Digital Twins in Engineering & Artificial Intelligence and Computational Methods in Applied Science,
DTE - AICOMAS 2025

February 17-21, 2025, Paris, France

Outline

Water infiltration in unsaturated soils

Water infiltration is the movement of water from the surface into the soil, driven by gravity and capillary forces.

Key factors :

- **Soil properties :** Texture, structure, and porosity control water absorption.
- **Surface conditions :** Vegetation and compaction affect infiltration rates.
- **External factors :** Rainfall and irrigation determine infiltration volume and speed.

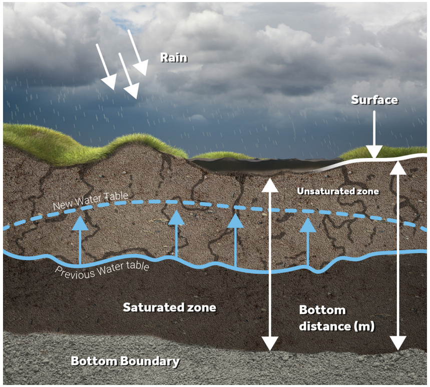


Figure – An example of an infiltration model (source : Wikipedia).

Modeling of Water infiltration

Infiltration models : Various models are available to simulate water infiltration, depending on the scale and purpose of the study. These include :

- **Zero-dimensional "bucket" models :** Simple representations of infiltration processes.
- **Richards equation (RICHARDS, 1931) :** A comprehensive partial differential equation (PDE) that models the dynamics of water content in soils.

Focus of this work : We adopt the Richards equation as the foundational model for simulating water infiltration in soils.

Richards equation 1931

$$\begin{cases} \dfrac{\partial \theta_w(\Psi)}{\partial t} = \nabla \cdot \underbrace{\Big[K(\Psi)\nabla(\Psi + z)\Big]}_{-\mathbf{q}}, \\ \text{Subject to suitable initial and boundary conditions.} \end{cases} \tag{1}$$

Notation	Meaning
Ψ	Pressure head.
(t, z)	Time-space coordinates.
K	Unsaturated hydraulic conductivity.
θ_w	Soil water content.
$\mathbf{q}$	Water flux.

VGM model

WRCs and HCFs

$$\theta_w = \begin{cases} \theta_r + (\theta_s - \theta_r)\left(1 + (\alpha_v|\Psi|)^{n_v}\right)^{-m_v}, & \text{if } \Psi \le 0, \\ \theta_s, & \text{if } \Psi > 0, \end{cases} \tag{2a}$$

$$K = \begin{cases} K_s \dfrac{\left[1 - (\alpha_v|\Psi|)^{n_v-1}\left(1 + (\alpha_v|\Psi|)^{n_v}\right)^{-m_v}\right]^2}{\left(1 + (\alpha_v|\Psi|)^{n_v}\right)^{m_v/2}}, & \text{if } \Psi \le 0, \\ K_s, & \text{if } \Psi > 0, \end{cases} \tag{2b}$$

$$m_v = 1 - \frac{1}{n_v}, \quad n_v > 1, \tag{2c}$$

N.B In the VGM model, high nonlinearity arises between pressure head and water content in the saturated zone. It is characterized by rapid saturation and hydraulic conductivity changes.

VGM model

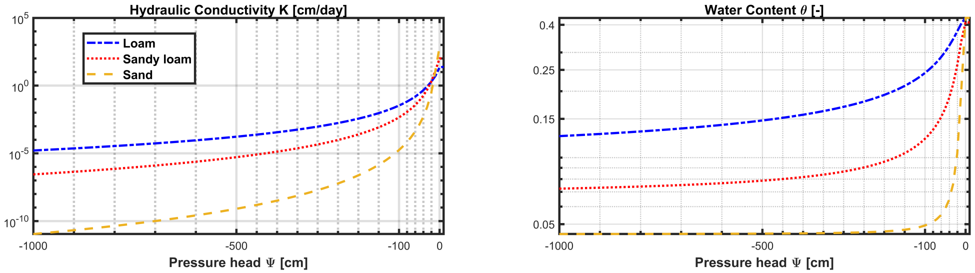


Figure – Constitutive relationships for three soil types (loam, sandy loam, and sandy soil) generated using the VGM model.

VGM model

Notation	Meaning
θ_s	Saturated volumetric water content.
θ_r	Residual volumetric water content.
K_s	Saturated hydraulic conductivity.
α_v, n_v, m_v	van Genuchten empirical parameters.

A homogeneous soil based on the VGM model is ultimately defined by the vector :

$$\boldsymbol{\lambda} = [\theta_s, \theta_r, K_s, \alpha_v, n_v, m_v].$$

These parameters collectively describe the soil's water retention and hydraulic conductivity properties.

Objective

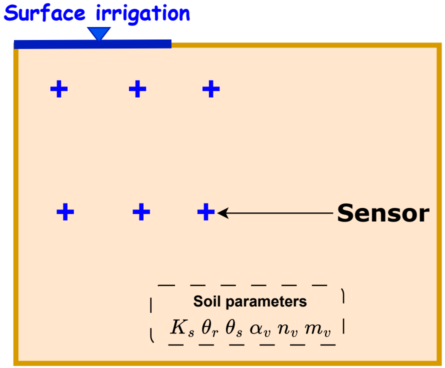
Objective : Develop an efficient and accurate framework to model water infiltration in unsaturated soils by addressing the challenges of inverse modeling through the use of deep learning and optimization techniques.

Specific goals :

- Use neural operators, specifically the DeepONet model (Lu et al., 2021), as a surrogate to reduce the computational burden of solving the Richards equation.
- Accurately infer unknown soil parameters from limited and uncertain data using advanced optimization strategies.

Methodology : Experimental setup

We consider a homogeneous soil equipped with multiple sensors installed at various depths and locations. An irrigation system is used to inject water, and the water content is measured by the sensors at different time steps.



Methodology

Our objective is to develop an efficient inversion surrogate model to address the challenges of soil parameter calibration.

Steps

- **Data generation :** Generate high-fidelity soil moisture data for a variety of possible soil configurations.
- **Surrogate model training :** Train a data-driven DeepONet model to map soil parameters to soil moisture distribution.
- **Inverse modeling :** Perform soil parameter calibration using the trained surrogate model.

Each step will be detailed in the following slides.

Methodology : Data generation

The first step involves generating high-fidelity simulations, using an in-house FEM Richard's solver (Kamil et al., 2024), for a variety of possible infiltration scenarios. Each scenario is defined by a unique vector of soil parameters λ.

Process

- Multiple soil parameters are generated within their corresponding physical bounds to represent diverse soil configurations :

$$\lambda \in [\lambda_{\text{min}}, \lambda_{\text{max}}], \quad \forall \lambda \in \Lambda. \tag{3}$$

- For a given soil parameter λ, the Richards equation is solved to obtain the full water content distribution in time and space.

This step ensures the generation of a comprehensive dataset required for training the surrogate model.

Methodology : Data generation

For each soil parameter λ, we derive the corresponding time-space soil water content profile $\theta_w(\lambda)$. This relationship is defined using the following representation operator :

$$\begin{aligned} \mathcal{O} : \quad & \Lambda \to \mathcal{S}, \\ & \lambda \mapsto \theta_w(\lambda), \end{aligned} \tag{4}$$

where Λ represents the soil parameter space, and $\mathcal{S}$ denotes the corresponding water content space.
The next step in the methodology involves approximating this operator.

At this stage, we aim to approximate the operator $\mathcal{O}$ using a deep learning model. Specifically, we adopt the concept of the DeepONet. The following section provides an illustration of this concept.

Methodology : DeepONet

A DeepONet $G_{\mathcal{W}}$ is a deep learning algorithm parameterized by $\mathcal{W}$, designed to approximate the operator $\mathcal{O}$. The DeepONet framework consists of two neural networks :

- **Branch network** : Encodes the soil parameters λ.
- **Trunk network** : Encodes the time-space coordinates y).

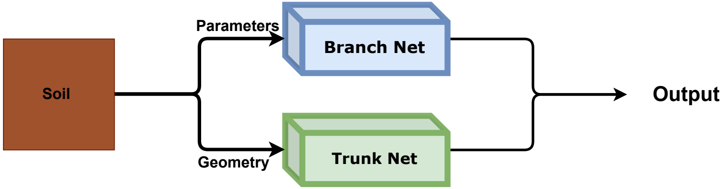


Figure – Illustration of the DeepONet model.

Methodology : DeepONet

A DeepONet model $G_{\mathcal{W}}$ operates as follows :

- **Branch Net** : Takes a soil parameter vector λ and outputs a latent representation :

$$[b_1, b_2, \cdots, b_q].$$

- **Trunk Net** : Takes the time-space coordinate y and outputs a latent representation :

$$[c_1, c_2, \cdots, c_q].$$

The final output of the DeepONet model is expressed as :

$$G_{\mathcal{W}}(\lambda)(y) = \sigma\left(\sum_{i=1}^{q} b_i c_i + c_0\right), \tag{5}$$

where :

- σ : Activation function.
- c_0 : Bias term.

Methodology : DeepONet

Training a DeepONet model can follow two main approaches :

Approach	Description
Data-driven	Utilizes high-fidelity data to train the model. This method is non-intrusive as it does not require access to the original PDE, relying solely on data obtained from high-fidelity simulations.
Physics-informed DeepONet	Incorporates the Richards equation as a physical constraint during the training process to improve the model's generalization and accuracy.

We adopted the data-driven approach, using a modified multilayer perceptron (MLP) architecture (Wang et al., 2021) for both the branch and trunk networks.

Loss function :

$$\mathcal{L}(\mathcal{W}) = \frac{1}{N_{\text{soil}} N_{\text{data}}} \sum_{i=1}^{N_{\text{soil}}} \sum_{j=1}^{N_{\text{data}}} \left| \theta_w^i(y^{(j)}) - G_{\mathcal{W}}(\lambda^{(i)})(y^{(j)}) \right|^2, \tag{6}$$

Methodology : Inverse modeling

The individual sensor Fitness (loss) $J_i(\lambda)$ quantifies the mismatch between the predicted and measured water content for sensor i. It is defined as :

$$J_i(\lambda) = \frac{1}{N_t} \sum_{j=1}^{N_t} \beta_i^{(j)} \left(G_{\mathcal{W}}(\lambda)(y_i^{(j)}) - \theta_i^{(j)} \right)^2, \tag{7}$$

where :

- $G_{\mathcal{W}}(\lambda)(y_i^{(j)})$: Predicted water content at $y_i^{(j)}$.
- $\beta_i^{(j)}$: Weighting parameter for the j-th measurement at sensor i.

Purpose : The sensor loss $J_i(\lambda)$ ensures that the optimization process prioritizes measurements according to their reliability ($\beta_i^{(j)}$).

Methodology : Inverse modeling algorithm

Algorithm 1 Inverse modeling algorithm

```
Input: DeepONet model $G_{\mathcal{W}}$, measured data $\{y_i^{(j)}, \theta_i^{(j)}\}$, optimizer, convergence criteria, and initial parameters $\lambda_{\text{init}}$ (if required).
Output: Calibrated soil parameters $\lambda^*$.
Step 1 : Initialize soil parameters : Set $\lambda \leftarrow \lambda_{\text{init}}$ (if initialization is required).
while Convergence criteria are not met do
    Step 2 : Forward pass : Use the DeepONet $G_{\mathcal{W}}$ with the current parameters $\lambda$ to predict water content at the sensor locations :
        $$G_{\mathcal{W}}(\lambda)(y_i^{(j)}) \quad \forall i, j.$$
    Step 3 : Compute the loss $J(\lambda)$.
end
```

Methodology : Inverse modeling algorithm (continued)

Algorithm 2 Inverse modeling algorithm (continued).

```
while Convergence criteria are not met (continued) do
    Step 4 : Update parameters : Use the optimizer to update parameters $\lambda$.
end
Step 5 : Output calibrated parameters : Return $\lambda^*$, the optimized soil parameters.
Step 6 : FEM Analysis : Use the calibrated parameters $\lambda^*$ with a finite element method (FEM) to analyze the water content profile in the soil for different irrigation systems.
```

Methodology : Inverse modeling (Optimization)

We adopt a two-step optimization to minimize the J-function :

Steps	Description
Step 1 : Genetic algorithm	Provides a robust initialization for the parameters. This algorithm does not require gradient information or an initial guess, making it well-suited for non-convex and multimodal problems.
Step 2 : BFGS optimizer	Refines the solution obtained from the genetic algorithm. The BFGS (Broyden-Fletcher-Goldfarb-Shanno) method can efficiently converges to an optimal solution.

Results : Problem description

We consider a 2D homogeneous soil domain with :

- Length : 1 m, Depth : 1.5 m.
- Initial water content : 0.2.
- We consider a loamy soil with exact parameters :

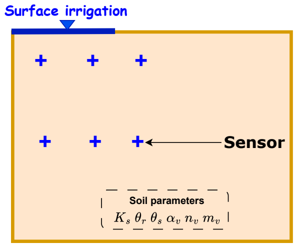
$$\lambda_{\text{loam}} = [\theta_s, \theta_r, K_s, \alpha_v, n_v] = [0.43, 0.078, 0.25, 3.6, 1.56]$$

Boundary conditions :

- Inlet flux : -1 m/day over a 7 cm section on the left.
- Remaining boundaries : No flux.

We have six sensors installed at positions :

- Depths : 5 cm and 15 cm.
- Horizontal positions : 5 cm, 15 cm, and 20 cm.



Results : Data Generation

We generated a set of 500 soil parameter vectors. For each parameter vector, we performed finite element simulations for a 3-hour infiltration event. Water content was recorded at the sensor locations every 1 minute. The training data are organized in the following form :

$$\mathcal{D} = \left(\lambda_s, \left(t_i, x_i, z_i, \theta_w^i(\lambda_s) \right)_{i=1}^{N_s} \right)_{s=1}^{N_p} \quad (8)$$

Where :

- λ_s : Soil parameter set for the s-th simulation.
- t_i : Time step i.
- (x_i, z_i) : Spatial location.
- $\theta_w^i(\lambda_s)$: Water content at the corresponding time and location.
- N_s : Number of data records summed up over all 6 sensors.
- N_p : Number of parameter sets.

Parameter	Range
θ_s	0.35 – 0.50
θ_r	0.05 – 0.15
K_s	0.1 – 1.0
α_v	1.5 – 5.0
n_v	1.2 – 2.0

Results : Surrogate Model Training

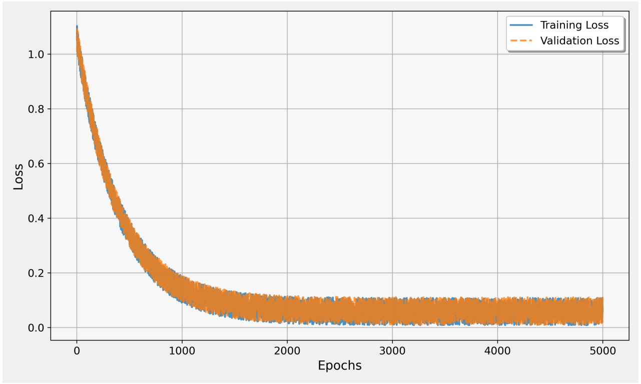


Figure – Training and validation loss curves for the for the best run.

Metric	Value
Training MSE	5.2×10^{-4}
Validation MSE	6.8×10^{-4}
Training MAE	1.1×10^{-2}
Validation MAE	1.4×10^{-2}
Training rel. L_2	3.4×10^{-2}
Validation rel. L_2	4.1×10^{-2}

Table – Training and validation metrics for the best run.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i^{\text{pred}} - y_i^{\text{true}}\right)^2, \; \text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i^{\text{pred}} - y_i^{\text{true}}\right|, \; \text{Rel } L_2 = \frac{\|y^{\text{pred}} - y^{\text{true}}\|_2}{\|y^{\text{true}}\|_2}$$

Results : Inverse modeling

We compare the DeepONet approach with two alternative methods :

MLP-approach

- Replaces DeepONet with a standard Multi-Layer Perceptron (MLP) as the surrogate model.

PINN-approach

- Trains a physics-informed neural network (PINN) (Raissi et al., 2019).
- Minimizes the data mismatch in $\mathcal{D}_m$ and adds a regularization term based on the Richards equation.
- Simultaneously learns the solution of the Richards equation and infers soil parameters.
- Optimization is performed using the Adam optimizer.

Results

Parameter	Exact Value	DeepONet	PINN	MLP
θ_s	0.430	0.432	0.435	0.448
θ_r	0.078	0.079	0.082	0.090
K_s (m/day)	0.250	0.252	0.259	0.275
α_v (1/m)	3.60	3.58	3.62	3.75
n_v	1.560	1.558	1.562	1.585

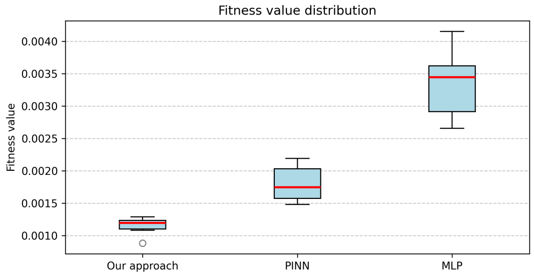
Table – Comparison of predicted parameter values.

Approach	Fitness value	Absolute error
DeepONet	1.2×10^{-3}	0.65%
PINN	1.8×10^{-3}	2.11%
MLP	3.5×10^{-3}	7.07%

Table – Fitness value and absolute error for each approach.

$$\text{Absolute error (\%)} = \frac{1}{5}\sum_{i=1}^{5}\left|\frac{\lambda_i - \lambda_i^{\text{exact}}}{P_i^{\text{exact}}}\right| \times 100$$

Results



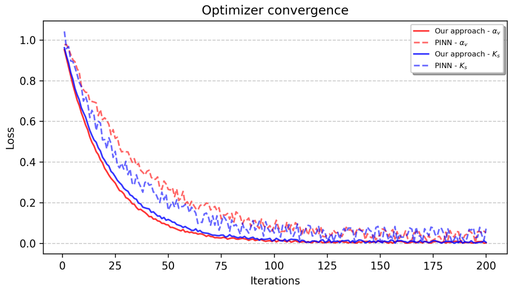


Figure – Boxplot showing the fitness value distribution for 10 runs (left) and training convergence of α_v and K_s for each approach (right).

Conclusion

- We proposed a novel approach using a DeepONet-based framework to infer soil parameters from sensor data, achieving superior accuracy compared to PINN and MLP-based methods.
- The proposed method demonstrated lower fitness values and smaller absolute relative errors across all evaluated soil parameters.
- Our approach showed robust convergence behavior during the optimization process, with consistent performance across multiple runs.
- The results highlight the effectiveness of integrating surrogate modeling and advanced optimization techniques for inverse modeling tasks. This approach can be used for other scientific/engineering problems.

Thank You!

Questions or Discussions?

Appendix : Loss parameters

The parameters $\beta_i^{(j)}$ are critical in weighting the measurements. They assign higher importance to more reliable measurements, where reliability is defined by low data uncertainty. We define $\beta_i^{(j)}$ as :

$$\beta_i^{(j)} = \tanh\left(\frac{1}{\gamma\left(\frac{y_i^{(j)} - y_i^m}{\sigma_i^m + \varepsilon}\right)^2}\right), \tag{9}$$

where :

- y_i^m : Mean of the measurements from sensor i.
- σ_i^m : Standard deviation of the measurements from sensor i.
- γ : A scaling parameter controlling sensitivity to uncertainty.
- ε : A small constant to prevent division by zero.

Appendix : Surrogate model training

Parameter/Process	Details
Data split	80% for training, 20% for validation.
Framework	DeepONet implementation using JAX library.
Parameter initialization	Xavier algorithm for weights, biases initialized to 0.
Optimizer	Adam optimizer.
Architecture	5 hidden layers with 100 neurons in each layer for both the branch and trunk networks.
Mini-batching	Mini-batches were used during training.
Early stopping	Training stopped upon reaching a predefined maximum number of epochs.

Appendix : Perspective

- **Model improvement :** Explore advanced surrogate modeling techniques, such as Physics-Informed Neural Operators (PINOs), to further enhance accuracy and efficiency.
- **Generalization :** Extend the framework to heterogeneous soils and evaluate its performance under varying environmental conditions.
- **Scalability :** Investigate the scalability of the approach for large-scale three-dimensional soil domains.
- **Real-world application :** Validate the methodology on real-world datasets, incorporating uncertainty quantification to address practical challenges.

- Richards, L. A. (1931). Capillary conduction of liquids through porous mediums. *physics, 1*(5), 318-333.
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks : A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics, 378*, 686-707.
- Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature machine intelligence, 3*(3), 218-229.
- Wang, S., Teng, Y., & Perdikaris, P. (2021). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing, 43*(5), A3055-A3081.
- Kamil, H., Beljadid, A., Soulaimani, A., & Bourgault, Y. (2024). Semi-implicit schemes for modeling water flow and solute transport in unsaturated soils. *Advances in Water Resources, 193*, 104835.