



University
Mohammed VI
Polytechnic



ÉCOLE DE
TECHNOLOGIE
SUPÉRIEURE
Le génie pour l'industrie
Université du Québec

**University Mohammed VI Polytechnic (UM6P)- Center for Doctoral Studies
École de Technologie Supérieure du QUÉBEC (ÉTS)**

Thesis submitted for the Degree of
Doctor of Philosophy

Specialty
**Applied Mathematics - UM6P
Engineering - ÉTS**

Presented by

Hamza Kamil

Supervised by

Abdelaziz Beljadid (UM6P), Azzeddine Soulaïmani (ÉTS)

**Physics-informed neural networks for modeling infiltration and
solute transport in porous media**

Defended on 03/18/2026, with the presence of the following assessment committee:

- Pr. François Duhaime	École de Technologie Supérieure	President
- Pr. Serge M. Prudhomme	Polytechnique Montréal, Canada	Referee
- Pr. Ivan Depina	Norwegian University of Science and Technology	Referee
- Pr. Mohamed Youssfi	ENSET Mohammedia	Referee
- Pr. Giuseppe Di Labbio	École de Technologie Supérieure	Examiner
- Pr. Bouchaib Radi	Faculté des Sciences et Techniques de Settat	Examiner
- Pr. Ahmed Taik	Faculté des Sciences et Techniques de Mohammedia	Examiner
- Pr. Abdelaziz Beljadid	University Mohammed VI Polytechnic	Thesis Director
- Pr. Azzeddine Soulaïmani	École de Technologie Supérieure	Thesis Director

ABSTRACT

This thesis addresses forward and inverse modeling of water infiltration and solute transport in porous media by developing efficient physics-constrained computational numerical models. For the mathematical description of water infiltration, two formulations are considered: the standard Richards equation, which predicts stable and monotonic wetting fronts, and the extended BCJ–R equation, which accounts for unstable infiltration phenomena such as saturation overshoot and gravity-driven fingered flow. To model solute transport, the infiltration models are coupled with advection–dispersion–reaction equations that capture the dynamics of single- and multi-species solute transport. The transport framework incorporates biogeochemical processes including first-order decay of nitrogen compounds through nitrification and denitrification, reversible adsorption of solutes onto soil particles, and transformation pathways such as volatilization, mineralization, and microbial degradation.

In the first part of this thesis, a numerical model based on a semi-implicit second-order mixed finite element method is developed to simulate coupled water flow and solute transport in unsaturated porous media. Water infiltration is considered under a stable front regime using the Richards equation, which is coupled with an advection–dispersion–reaction system to represent fertilizer transport. The developed numerical model avoids nonlinear iterative solvers for the Richards equation, thereby mitigating convergence issues associated with highly nonlinear soil capillary models. Numerical experiments are conducted to solve the coupled system under scenarios including single- and multi-species nitrogen transport, pore-water electrical conductivity, and nitrate migration. Validation against analytical solutions, established numerical codes, and experimental datasets confirms the model’s accuracy in reproducing water infiltration dynamics and solute transport processes. This high-fidelity solver serves both as a benchmark and as a data source for the physics-informed machine learning models developed in subsequent chapters.

In the second part, a physics-informed neural network (PINN) is developed for forward modeling of coupled water flow and solute transport. The model requires only the initial and boundary conditions, while the governing equations are embedded directly into the training process. The PINN’s performance is evaluated against the high-fidelity finite element solver from the first part, showing good agreement and confirming the potential of PINNs for forward modeling of water–solute transport in porous media.

In the third part, the objective is to identify the most effective PINN strategy for modeling water infiltration and single- and multispecies solute transport in porous media. A comprehensive comparative study of different PINN formulations is conducted, evaluating their performance in forward simulations of coupled flow–transport processes

as well as in inverse problems for estimating unknown model parameters. Among the approaches examined, the sequential PINN strategy, where the governing equations for water flow and solute transport are trained separately, emerges as the most effective. Its superiority is demonstrated through extensive validation against synthetic datasets and experimental benchmarks, establishing its robustness for predictive simulations and parameter identification in unsaturated porous media.

In the fourth part, the sequential PINN framework is extended with transfer learning to address the complexity of multispecies solute transport in unsaturated soils. The parameters learned during the training of the Richards equation are used to initialize the solute transport PINNs, exploiting structural similarities between the governing PDEs, both of which involve advection–diffusion-type operators. This strategy accelerates convergence, enhances training stability, and improves predictive accuracy compared to randomly initialized models, as demonstrated in forward and inverse simulations across one- and two-dimensional soil configurations.

In the fifth part, the sequential PINN paradigm is generalized to arbitrary one-way coupled systems, where each equation depends on the outputs of preceding ones in a fixed causal order. New training schemes are introduced to exploit these directional dependencies, mitigating convergence issues that arise when equations are trained simultaneously. The framework shows clear advantages over standard PINNs, achieving good accuracy and efficiency in modeling the coupled dynamics of water flow and multispecies solute transport in porous media.

In the sixth part, the challenging problem of unstable infiltration in unsaturated soils is addressed by numerically solving the fourth-order BCJ–R equation. It was found that the PINN model for the Richards equation is not suitable for this case; therefore, a new solver is developed with tailored training strategies and modified neural architectures to manage the multiscale behavior and stiffness inherent to the BCJ–R equation. The resulting PINN model accurately reproduces key features such as saturation overshoot, shows good agreement with experimental data, and remains consistent with thermodynamic principles, as verified through entropy dissipation analysis in closed systems.

Finally, the last part of this thesis investigates multitask learning for infiltration in unsaturated flow through the development of a fast and accurate surrogate model. A deep neural operator is introduced and trained in a meshless, data-free, and physics-informed setting to learn the mapping from varying initial and boundary conditions to the corresponding soil moisture distributions. To enhance extrapolation capability, a fine-tuning stage is incorporated using either physical constraints or available soil moisture data. The model is validated against the high-fidelity finite element solver, achieving comparable accuracy at a fraction of the computational cost during inference. This framework establishes the neural operator as an efficient and reliable real-time surrogate for optimizing irrigation strategies.

RÉSUMÉ

Cette thèse traite de la modélisation directe et inverse de l'infiltration d'eau et du transport de solutés dans les milieux poreux par le développement de modèles numériques computationnels efficaces, contraints par la physique. Pour la description mathématique de l'infiltration d'eau, deux formulations sont considérées : l'équation de Richards standard, qui prédit des fronts d'humectation stables et monotones, et l'équation étendue BCJ-R, qui rend compte des phénomènes d'infiltration instable tels que le dépassement de saturation et les écoulements préférentiels à doigts gravitaires. Pour modéliser le transport de solutés, les modèles d'infiltration sont couplés à des équations d'advection–dispersion–réaction qui capturent la dynamique du transport de solutés mono- et multi-espèces. Le cadre de transport intègre des processus biogéochimiques, notamment la dégradation d'ordre un des composés azotés par nitrification et dénitrification, l'adsorption réversible des solutés sur les particules du sol, ainsi que des voies de transformation telles que la volatilisation, la minéralisation et la dégradation microbienne.

Dans la première partie de cette thèse, un modèle numérique basé sur une méthode des éléments finis mixtes semi-implicite d'ordre deux est développé pour simuler l'écoulement de l'eau et le transport de solutés dans les milieux poreux non saturés. L'infiltration de l'eau est considérée dans un régime de front stable à l'aide de l'équation de Richards, couplée à un système d'advection–dispersion–réaction pour représenter le transport des engrais. Le modèle numérique développé évite le recours à des solveurs itératifs non linéaires pour l'équation de Richards, ce qui permet de limiter les problèmes de convergence liés aux modèles capillaires du sol fortement non linéaires. Des expériences numériques sont menées pour résoudre le système couplé dans divers scénarios, incluant le transport mono- et multi-espèces de l'azote, la conductivité électrique de l'eau interstitielle et la migration des nitrates. La validation par comparaison avec des solutions analytiques, des codes numériques établis et des données expérimentales confirme la précision du modèle dans la reproduction des dynamiques d'infiltration de l'eau et des processus de transport des solutés. Ce solveur haute fidélité sert à la fois de référence et de source de données pour les modèles d'apprentissage automatique informés par la physique développés dans les chapitres suivants.

Dans la deuxième partie, un réseau de neurones informé par la physique (PINN) est construit pour modéliser la dynamique couplée de l'écoulement de l'eau et du transport de solutés. La méthode intègre directement les équations gouvernantes dans le processus d'apprentissage et ne requiert que les conditions initiales et aux limites. Ses performances sont évaluées par comparaison avec le solveur aux éléments finis haute fidélité présenté précédemment, montrant un accord étroit et confirmant le potentiel des PINNs pour la modélisation directe du transport eau–solutés dans les milieux poreux.

Dans la troisième partie, l’objectif est d’identifier la stratégie PINN la plus efficace pour la modélisation de l’infiltration de l’eau et du transport des solutés mono- et multispecies dans les milieux poreux. Une étude comparative approfondie de différentes formulations de PINN est menée, évaluant leurs performances à la fois dans les simulations directes des processus couplés écoulement–transport et dans les problèmes inverses visant l’estimation de paramètres inconnus. Parmi les approches examinées, la stratégie séquentielle PINN, dans laquelle les équations gouvernantes de l’écoulement et du transport sont entraînées séparément, s’avère la plus efficace. Sa supériorité est démontrée par une validation approfondie sur des jeux de données synthétiques et des bancs d’essai expérimentaux, confirmant sa robustesse pour les simulations prédictives et l’identification de paramètres en milieux poreux non saturés.

Dans la quatrième partie, le cadre séquentiel PINN est enrichi par l’apprentissage par transfert afin de traiter la complexité du transport multispecies dans les sols non saturés. Les paramètres appris lors de l’entraînement de l’équation de Richards sont utilisés pour initialiser les PINN de transport des solutés, en exploitant les similarités structurelles entre les équations gouvernantes, qui comportent toutes deux des opérateurs de type advection–diffusion. Cette stratégie accélère la convergence, améliore la stabilité de l’entraînement et accroît la précision prédictive par rapport aux modèles initialisés aléatoirement, comme démontré dans les simulations directes et inverses en une et deux dimensions.

Dans la cinquième partie, le paradigme PINN séquentiel est généralisé à des systèmes couplés unidirectionnels arbitraires, où chaque équation dépend des sorties des équations précédentes dans un ordre causal fixé. De nouveaux schémas d’entraînement sont introduits pour exploiter ces dépendances directionnelles, atténuant les problèmes de convergence rencontrés lorsque les équations sont entraînées simultanément. Le cadre présente des avantages clairs par rapport aux PINN standards, atteignant une bonne précision et efficacité pour la modélisation des dynamiques couplées écoulement–transport multispecies dans les milieux poreux.

Dans la sixième partie, le problème difficile de l’infiltration instable dans les sols non saturés est abordé en résolvant numériquement l’équation de BCJ–R d’ordre quatre. Il a été constaté que le modèle PINN conçu pour l’équation de Richards n’est pas adapté à ce cas ; par conséquent, un nouveau solveur est développé, intégrant des stratégies d’entraînement adaptées et des architectures neuronales modifiées, afin de gérer la nature multiscalaire et la raideur propres à l’équation de BCJ–R. Le modèle obtenu reproduit avec précision des phénomènes clés tels que le dépassement de saturation, montre un bon accord avec les données expérimentales et reste cohérent avec les principes thermodynamiques, comme vérifié par une analyse de dissipation d’entropie en systèmes fermés.

Enfin, la dernière partie de cette thèse étudie l’apprentissage multitâche pour l’infiltration

en régime non saturé à travers le développement d'un modèle de substitution rapide et précis. Un opérateur neuronal profond est proposé et entraîné dans un cadre sans maillage, sans données et informé par la physique, afin d'apprendre la relation entre des conditions initiales et aux limites variables et les distributions correspondantes d'humidité du sol. Pour améliorer la capacité d'extrapolation, une étape d'ajustement fin est intégrée en utilisant soit des contraintes physiques, soit des données disponibles d'humidité du sol. Le modèle est validé par comparaison avec le solveur aux éléments finis haute fidélité, atteignant une précision comparable tout en réduisant considérablement le coût de calcul en phase d'inférence. Ce cadre établit l'opérateur neuronal comme un substitut efficace et fiable en temps réel pour l'optimisation des stratégies d'irrigation.

ACKNOWLEDGMENT

First, I would like to express my deepest gratitude to my supervisors, Professor Abdelaziz Beljadid and Professor Azzeddine Soulaïmani, for their invaluable guidance, unwavering support, and constructive feedback throughout this research. Their expertise and encouragement have been instrumental in shaping the scientific content and the overall direction of this thesis project. I am also sincerely thankful to University Mohammed VI Polytechnic (UM6P, Benguerir, Morocco) and École de Technologie Supérieure (ÉTS, Montréal, Canada) for providing a collaborative and intellectually stimulating academic environment through their joint cotutelle program. The support from both institutions has been essential to the success of my doctoral journey. I extend my appreciation to the Digital Research Alliance of Canada and the Simlab high-performance computing platform at UM6P for the generous allocation of computational resources, which enabled the extensive simulations and experiments conducted in this thesis project. Finally, I gratefully acknowledge the encouragement and support of my colleagues, friends, and family, whose belief in me has been a constant source of motivation throughout my educational life.

Funding for this research was provided by UM6P/OCP Group of Morocco (UM6P-OCP-2022), the APRD research program from Moroccan Ministry of Higher Education, Scientific Research and Innovation and OCP Foundation (25219) and Natural Sciences and Engineering Research Council of Canada (RGPIN-2021-02693 & RGPIN-2022-5220).

CONTENTS

ABSTRACT	ii
RÉSUMÉ	iv
ACKNOWLEDGMENT	vii
CONTENTS	viii
LIST OF TABLES	xiii
LIST OF FIGURES	xv
CHAPTER 1 Introduction	1
1.1 Background and motivation	1
1.1.1 Water scarcity, fertilizer use, and environmental challenges	1
1.1.2 Mathematical modeling of irrigation and fertigation strategies . .	2
1.1.3 Modeling water and solute transport in porous media	3
1.1.4 Thesis motivation	5
1.2 Mathematical models	7
1.2.1 Water infiltration	7
1.2.2 Solute transport	8
1.3 Numerical models	8
1.3.1 Standard numerical methods	8
1.3.2 Standard machine learning	11
1.3.3 Scientific machine learning	13
1.4 Methodology	19
1.4.1 Finite element method	19
1.4.2 Physics-informed neural networks	20
1.4.3 Neural operators	22
1.5 Thesis outcome	24
1.5.1 Contributions	24
1.5.2 Thesis outline	27
CHAPTER 2 Semi-implicit schemes for modeling water flow and solute transport in unsaturated soils	28
2.1 Introduction	28
2.2 Governing equations	31
2.2.1 Water flow	31
2.2.2 Solute transport	33

2.2.3	Initial and boundary conditions	35
2.3	Weak formulations	35
2.4	Numerical methods	36
2.4.1	Time stepping schemes for the Richards equation	37
2.4.2	Time stepping schemes for the solute transport equation	42
2.5	Numerical experiments	44
2.5.1	Water infiltration	45
2.5.2	Water flow and solute transport	55
2.6	Conclusion	66
CHAPTER 3	Physics-informed neural networks vs finite element method for modeling coupled water and solute flow in unsaturated soils	72
3.1	Introduction	72
3.2	The physical models	73
3.2.1	Water flow	73
3.2.2	Solute transport	74
3.3	Methods	75
3.3.1	Finite element method: FEM	75
3.3.2	Physics-informed neural networks: PINNs	75
3.4	Numerical experiments	76
3.5	Conclusion	78
CHAPTER 4	A comparative study of physics-informed neural network strategies for modeling water and nitrogen transport in unsaturated soils	80
4.1	Introduction	80
4.2	The mathematical model	83
4.2.1	Water flow model	83
4.2.2	Solute transport model	84
4.2.3	Initial and boundary conditions	86
4.3	Methods	87
4.3.1	Feedforward neural networks: FNNs	87
4.3.2	Physics-informed neural networks	87
4.3.3	Single PINN solver	90
4.3.4	Multiphysics-informed neural networks: MPINN solver	90
4.3.5	Sequential physics-informed neural network: S-PINN solver	91
4.3.6	Algorithm implementation	93
4.4	Numerical experiments	93
4.4.1	Test case A: 1D coupled water flow and solute transport	94
4.4.2	Test case B: Nitrogen species transport	97
4.4.3	Test case C: 2D water flow and solute transport under uniform boundary conditions	105

4.4.4	Test case D: 2D nitrate transport under nonuniform boundary conditions	106
4.5	Limitations	108
4.6	Conclusion	110
CHAPTER 5 A transfer learning physics-informed deep learning framework for modeling multiple solute dynamics in unsaturated soils		
5.1	Introduction	119
5.2	The mathematical models	122
5.2.1	Water flow model	122
5.2.2	Solute transport model	123
5.2.3	Nitrogen species transport model	124
5.2.4	Initial and boundary conditions	126
5.3	Studied cases	126
5.4	Methods	128
5.4.1	Physics-Informed Neural Network: PINN	128
5.4.2	Improved Physics-Informed Neural Network	130
5.4.3	Transfer learning Physics-Informed Neural Network: TL-PINN	132
5.4.4	Transfer learning with Mixed-PINN	135
5.4.5	Algorithm implementation	136
5.5	Results and discussions	139
5.5.1	A comparative analysis of transfer learning PINNs and standard PINNs	139
5.5.2	Random initializer PINN vs transfer learning PINN	140
5.5.3	Investigating the influence of FNN architecture and residual points number on TL-PINN solver performance	144
5.5.4	Impact of the freezing technique on the transfer learning PINN	144
5.5.5	Training stability in the transfer learning PINN	145
5.5.6	2D simulations using the transfer learning PINN solver	148
5.5.7	Inverse problem with unknown initial and boundary conditions	149
5.5.8	Simulations using the transfer learning mixed-PINN	151
5.6	Limitations and future objectives	156
5.7	Conclusion	157
CHAPTER 6 A comprehensive analysis of physics-informed neural networks for solving one-way coupled problems		
6.1	Coupled systems	161
6.2	Motivation	161
6.2.1	One-way causality violation	163
6.2.2	Neural tangent kernel analysis	163
6.2.3	Loss landscape analysis	165

6.2.4	Ill-conditioning problem	166
6.2.5	Gradient conflict problem	167
6.3	Related works	168
6.4	Physics-informed neural networks	169
6.4.1	PINNs for single-physics problems	169
6.4.2	PINNs for coupled systems	170
6.4.3	PINNs for one-way coupled systems	172
6.5	Numerical examples	177
6.5.1	Thermo-mechanics coupling problem	178
6.5.2	Electro-thermal coupling problem	180
6.5.3	Infiltration-transport in unsaturated soils	181
6.5.4	Infiltration and multi-species solute transport problem	183
6.6	Conclusion	186
CHAPTER 7 Physics-informed neural networks for solving the BCJ-R equation:		
	Modeling saturation overshoot in porous media	195
7.1	Introduction	195
7.2	Mathematical model	197
7.2.1	Governing equations	197
7.2.2	Constitutive functions	197
7.2.3	Initial and boundary conditions	198
7.2.4	Entropy function	199
7.3	Methods	200
7.3.1	Overview of physics-informed neural networks	200
7.3.2	Enhancements in PINNs	201
7.3.3	PINNs for the BCJ-R equation	204
7.3.4	Implementation	206
7.4	Results	206
7.4.1	Saturation overshoot simulation	207
7.4.2	Entropy test analysis	208
7.4.3	Comparison with experimental data	208
7.5	Conclusion	211
CHAPTER 8 Physics-informed neural operators for efficient modeling of infiltration in porous media		
		219
8.1	Introduction	219
8.2	The mathematical model	223
8.3	Methods	226
8.3.1	DeepONet for the Richards equation	226
8.4	Results	236

8.4.1	Scenario (a): Mapping initial soil moisture to the corresponding pressure head profile	237
8.4.2	Scenario (b): Mapping water inlet boundary conditions to the corresponding pressure head profiles	244
8.5	Discussion	248
8.6	Conclusion	252
CHAPTER 9	Conclusion	257
PUBLICATIONS		260
CONFERENCES & WORKSHOPS		261
REFERENCES		263

LIST OF TABLES

2.1	A summary of the proposed semi-implicit schemes. The table provides a reference to the numerical treatment of each term in the Richards equation.	42
2.2	Soil hydraulic functions for different models.	46
2.3	The computed pressure head and saturation L^2 -errors for the semi-implicit (S, Ψ) -schemes at time $T = 10$ days.	51
2.4	The computed pressure head and effective saturation L^2 -errors for the semi-implicit (S, Ψ) -schemes in function of the soil parameter $\tilde{\alpha}$ at time $T = 10$ days with $\Delta t = 5 \times 10^{-3}$ day and 50×50 elements in the first test scenario.	52
2.5	Parameters for the soil textures in the curvilinearly layered soil test case.	54
2.6	Order of convergence for the semi-implicit (S, Ψ) -schemes in the curvilinearly layered soil test case.	55
3.1	The obtained results for both PINN and FEM in solving the one- and two-dimensional infiltration-solute model.	77
4.1	Summary of numerical tests in this paper.	94
4.2	Parameter configuration for training the PINN solvers across all examined test cases. IBC stands for initial and boundary conditions.	97
4.3	Test case A - Relative L^2 -error for the PINN solvers with and without adaptive activation (AA). For the S-PINN solver, $\cdot/\cdot$ represents the total loss of the PINN water solver followed by the total loss of the PINN solute solver.	97
4.4	Test case B - Inverse problem: Predicted value of the nitrification rate μ_1 in day^{-1} using three different PINN solvers with adaptive activation. Note that $\mu_1 = 0.12 \text{ day}^{-1}$	104
4.5	Comparison of training and prediction CPU time for different solvers	117
5.1	Relative L_2 -error for the inverse problem with different magnitude of noisy data used for the training.	154
6.1	Some examples of one-way coupled systems from the literature.	162
6.2	Summary of PINN solver, showing causality, causality type, weight type, training interval of equation i , and loss function.	177
6.3	Inverse parameter estimation of α using PINN solvers in a 1D thermo-mechanical system. The statistics (mean $\pm$ std) are derived from five independent trials. Ground truth is $\alpha = 1$	179

6.4	Practical guideline for selecting a PINN solver based on problem characteristics.	188
7.1	PINN configurations for the studied numerical tests (see Sections 7.4.1, 7.4.2, and 7.4.3).	215
7.2	Ablation study of design choices on model performance and runtime for the MLP architecture over 50k iterations using Adam optimiser. Abbreviations: F.F. = Fourier Features, W.F. = Weight Factorization, G.N. = Gradient Norm, T.C. = Time Causality, T.M. = Time Marching, L_2 = Relative Error, B.M. = Best Model (iteration).	216
7.3	Ablation study of design choices on model performance and runtime for the MLP architecture over 50k iterations using SOAP optimiser.	217
8.1	Parameters of three types of soil used for the Mualem-van Genuchten model.	226
8.2	Scenario (a) – The initial soil moisture θ_0 [m^3/m^3] for each type of soil and geometry. For scenario (a) – parametric initial conditions, we specify the range (min–max) within which we randomly generate the soil moisture values.	238
8.3	Scenario (a)-Average and standard deviation of the relative L^2 -error. $\cdot \cdot$ indicates training test errors.	241
8.4	Scenario (b)-Average and standard deviation of the relative L^2 -error for scenario (b). $\cdot \cdot$ indicates train test errors.	246
8.5	DeepONet training setup for scenario (a).	254
8.6	DeepONet training setup for Scenario (b). Architecture, optimizer, and training settings are identical across all scenarios: a 6-layer $\times$ 100-neuron network, 172,200 trainable parameters, Adam optimizer, 200,000 epochs, batch size 10,000, and loss weights $\alpha_{\text{IC}} = 1$, $\alpha_{\text{BC}} = 10$	254

LIST OF FIGURES

1.1	General workflow of a PINN solver. Collocation points for the PDE residual, boundary and initial conditions, and optional data are supplied to a neural network that represents the solution. Automatic differentiation provides spatial and temporal derivatives to form residual and constraint terms. A composite physics-informed loss is minimized with standard optimizers until convergence, yielding a trained surrogate for forward prediction and, when applicable, estimated parameters for inverse problems. . . .	22
1.2	A schematic representation of the DeepONet framework. The architecture consists of a branch network that processes the input function from the parameter space $\mathcal{P}$, and a trunk network that handles the coordinates of the computational domain Ω . The outputs of the two networks are combined to approximate the target operator G . DeepONet can be trained using a data-driven loss, a physics-informed loss, or a combination of both, optimized using for example gradient-based optimizers.	25
2.1	Examples of the absolute value of the Leverett function for various capillary pressure models.	33
2.2	Flow chart of the coupled numerical model of water flow and solute transport.	44
2.3	2D homogeneous unsaturated soil $[0, l] \times [0, L]$	46
2.4	Boundary conditions employed in both two-dimensional infiltration test cases.	47
2.5	Pressure head function applied at the soil's upper boundary in both test cases.	48
2.6	Contours of the saturation at time $T = 10$ days obtained from four semi-implicit (S, Ψ) -schemes and the exact solution with $\Delta t = 10^{-2}$ day and 25×25 elements.	49
2.7	Analysis of the pressure head and the saturation L^2 -error with respect to the free parameters (α, β) with $\Delta t = 10^{-2}$ day and 25×25 elements in the first test scenario.	50
2.8	Evaluation of the modified L^2 -error in the log-log scale of the numerical schemes as a function of the time step Δt in the first test scenario.	50
2.9	Time-evolution and the relative error of the total mass in the computational domain: Exact and numerical schemes with $\Delta t = 10^{-2}$ day and 25×25 elements.	52

2.10	(Left) Evolution of saturation over time. (Middle) Evolution of pressure head over time. (Right) Modified L^2 -error as a function of time step Δt . Simulations were performed using a time step of $\Delta t = 0.01$ second and a mesh size of $h = 0.1$ cm.	54
2.11	2D curvilinearly layered soil: $[0, 100 \text{ cm}]^2$ - initial and boundary conditions.	55
2.12	Saturation profile evolution using the SBDF scheme at various times.	56
2.13	Pressure head and saturation L^2 -errors in function of the mesh size h for the studied semi-implicit (S, Ψ) -schemes in the curvilinearly layered soil test case.	56
2.14	Horizontal infiltration test domain.	58
2.15	Saturation (left) and normalized concentration profiles (right) for different time steps Δt . The final simulation time is $T = 1$ and the mesh size is $h = 0.01$. The normalized concentration is given by: $c = (c - c_r)/(c_s - c_r)$	59
2.16	Saturation and normalized concentration profiles for different α and β values. The final simulation time is $T = 1$ and the mesh size is $h = 0.01$	59
2.17	Concentration profiles for the SBDF scheme and HYDRUS-1D at $T = 1$ day (top) and cross-section at $x = 0.5$ m for pressure head, water content, and solute concentration at $T = 10$ days (down).	61
2.18	Time-evolution of the total mass of water and solute obtained by the SBDF scheme and HYDRUS-1D.	61
2.19	Diagram illustrating the first-order decay chain of nitrogen species (left) and nitrogen species concentrations at time $T = 1$ day (right).	63
2.20	Water flow and solute transport 2D domain for pore-water EC test (left) and for nitrate test (right).	64
2.21	(a) Pressure head and (b) Pore-water EC profiles at various times obtained using the SBDF scheme.	65
2.22	(a) Pressure head and (b) Pore-water EC vertical profiles ($x = 0$ m) as functions of soil depth at various times: SBDF scheme results and comparison with Haruzi and Moreno data (Haruzi and Moreno, 2023c).	65
2.23	Cross-sections of water content (top) and nitrate concentration (down) at $x = 2.5$ cm and $x = 12.5$ cm after 8 hours of application.	67
2.24	Schematic description of nitrate fertigation strategies.	68
2.25	Cross-sections of nitrate concentration for each fertigation strategy at $x = 2.5$ cm (left) and $x = 12.5$ cm (right).	68

2.26	Water content at various times (top) and nitrate concentration at 6 hours for each fertigation strategy (down) using the SBDF scheme.	68
3.1	Diagram illustrating the PINN solver for the coupled water flow and solute transport model.	76
3.2	2D numerical test- Snapshots of the obtained results from PINN and cross-sections at $x = 0$ of the pressure head and solute concentration obtained using PINN and FEM solvers.	78
4.1	Adaptive activation function. Hyperbolic tangent (a) and sigmoid (b) activation functions for various values of $s\alpha$	88
4.2	Example of 2D time-space collocation points (a) and 3D time-space collocation points (b).	89
4.3	Schematic representation of the proposed PINN solvers for the coupled water flow and solute transport model. For notations, refer to Section 5.4.	92
4.4	Test case A – Predictions of matric head Ψ and solute concentration c in the time-space domain $[0, 1] \times [0, 1]$ using MPINN with adaptive activation, along with the absolute error between the predicted and true profiles. Bottom: Predictions from MPINN of Ψ , c , and $\mathbf{q}$ at $T = 1$ day.	98
4.5	Test case A- Evolution of loss functions for the PINN solvers during Adam epochs.	99
4.6	Test case A- The predicted matric head, solute concentration, and water flux obtained from various PINN solvers at time $T = 1$ day. The results are compared with those obtained from HYDRUS-1D.	99
4.7	Test case A- Water and solute mass evolution as a function of time obtained from different PINN solvers compared with HYDRUS-1D.	100
4.8	Test case A- Evolution of the total loss function during training and testing for the S-PINN solver with adaptive activation during Adam epochs.	101
4.9	Test case B- Illustration of the nitrogen species first-order decay chain.	101
4.10	Test case B- Forward problem: Predicted matric head and nitrogen species concentrations obtained from various PINN solvers at time $T = 1$ day.	102
4.11	Test case B - Forward problem: Predicted matric head and nitrogen species concentrations using the MPINN solver with adaptive activation at different Adam epochs, at time $T = 1$ day.	103

4.12	Test case B- Inverse problem: Predicted matric head and ammonium concentration obtained using the S-PINN solver at time $T = 1$ day.	105
4.13	Water flow and solute transport 2D domain for test case C (left), and for test case D (right).	106
4.14	Test case C – Results obtained using the S-PINN solver with adaptive activation and the FEM. The first and second rows show the 2D temporal evolution of the matric head and solute concentration at snapshots taken at 0.25, 0.5, 0.75, and 1 day. The third row presents cross-sectional views of the matric head and solute concentration at different time points.	107
4.15	Test case D - Predicted 2D profiles of water content (top) and nitrate concentration (bottom) at various time snapshots using the S-PINN solver with adaptive activation	108
4.16	Cross-sections of the water content at coordinates $x = 2.5$ cm, $z = 2.5$ cm (left), and $x = 12.5$ cm, $z = 12.5$ cm (right), as predicted using the S-PINN solver with adaptive activation. The predicted results are compared with experimental data sourced from (Li <i>et al.</i> , 2005).	109
4.17	Impact of hyperparameters on accuracy: Relative L^2-error for different hyperparameter settings. The left panel shows the analysis of neural network architecture, while the right panel illustrates the influence of loss term weights on accuracy. The parameter α_{up} represents the weight for the soil surface boundary condition, where water infiltrates the soil domain with an inflow q_0 (see Test Case A). Results are reported for the S-PINN solver with adaptive activation.	115
4.18	(Left) Comparison of total loss for S-PINN solver: Adam optimizer (blue) vs. L-BFGS (red). (Right) Breakdown of the residual, initial condition, and boundary losses using the Adam optimizer. The upper loss refers to the loss at the soil surface, while the lower loss corresponds to the loss at the soil bottom boundary.	116
4.19	Comparison of S-PINN and DeepONet performance: (a) DeepONet loss function evolution over Adam epochs; (b) Computational cost comparison (in seconds) between S-PINN (including training and prediction) and DeepONet predictions, based on the number of PDEs to be solved (number of initial conditions).	117
5.1	Representation of the first-order decay chain of nitrogen species.	125
5.2	Configuration of 1D and 2D numerical experiments for water flow and nitrogen species transport.	127

5.3	Diagram illustrating a standard PINN solver for the coupled water flow and nitrogen species transport model.	130
5.4	Activation functions: Hyperbolic tangent (left) and sigmoid (right) with varied $s\alpha$ values.	131
5.5	The overall flowchart of the transfer learning PINN solver for modeling water flow and multiple species transport in unsaturated soils.	135
5.6	Various sampling methods for residual collocation points within the time-space domain $[0, 1] \times [-1, 0]$	138
5.7	Profiles (top) and errors (down) of pressure head and nitrogen species concentrations obtained using TL-PINNs in the time-space domain $[0, 1 \text{ day}] \times [0, 1 \text{ m}]$	140
5.8	Pressure head, water flux and nitrogen species concentrations obtained using TL-PINNs and standard PINNs at time $T = 1 \text{ day}$	141
5.9	Total loss function of the standard PINN and the TL-PINN solver during Adam iterations.	142
5.10	Relative L^2 -error of TL-PINN, RD-PINN, and RD1-PINN for each nitrogen species concentration.	143
5.11	The total loss function values for c_1 and c_2 for TL-PINN and RD-PINN solvers. Ammonium and nitrite concentrations are denoted as c_1 and c_2 respectively.	143
5.12	The relative L^2 -error as a function of the number of residual collocation points for each PDE solution (Ψ, c_i)	145
5.13	The c_1 , c_2 , and c_3 total loss function evolution as a function of the Adams iterations for different numbers of residual collocation points.	146
5.14	The relative L^2 -error with varying hidden layers and neurons for each PDE solution (Ψ, c_i) in the TL-PINN solver.	147
5.15	Influence of freezing hyper-parameters k_{c_1} , k_{c_2} , and k_{c_3} on L_2 -error for ammonium (c_1), nitrite (c_2), and nitrate (c_3).	148
5.16	Evolution of total loss across Adam iterations for each PINN solver.	149
5.17	2D profiles of water content, ammonium, nitrite, and nitrate in the time-space domain $\{T = 1\} \times [0, 1]^2$	150
5.18	The two studied scenarios in the inverse problem.	152
5.19	Inverse problem-scenario 1: Profiles (top), errors (down) of pressure head and nitrogen species concentrations obtained using TL-PINN in the time-space domain $[0, 1] \times [0, 1]$	152
5.20	Inverse problem test: Profiles of pressure head and water flux obtained using a PINN and a standard FNN at $T = 1 \text{ Day}$ after 2000 Adams epochs.	153

5.21	Layered soil test: Profiles (top) and cross-sections (bottom) of water content, water flux, and ammonium concentration obtained using the transfer learning mixed-PINNs in the time-space domain $[0, 2 \text{ days}] \times [-1, 0 \text{ m}]$. The results are compared with HYDRUS, represented by red dashed lines. Panel (b) depicts scenario 1 using the soil parameters detailed in (5.23), while panel (c) presents the second scenario in which the discontinuity between the two layers is reduced by adjusting the saturated hydraulic conductivity as follows: $K_s^{\text{layer } 2} = 2 \times K_s^{\text{layer } 1}$	155
6.1	Illustration of one-way causality violation. Comparison between a standard PINN (a) and a one-way causal PINN (b) in predicting coupled water flow and solute transport, using finite element solutions as reference. Each subplot shows the solution profiles of PDE 1 (left) and PDE 2 (right) across Adam iterations. The standard PINN (a) fails to respect the causal structure of the system, leading to incorrect learning due to spectral bias. In contrast, the causal PINN (b) focuses training on PDE 1 first, preserving the one-way dependency.	164
6.2	Neural tangent kernel analysis. This figure illustrates the issues of spectral bias, unbalanced back-propagated gradients, and differing convergence rates across PDEs. (Left) Eigenvalue spectra of the NTK for each PDE. (Middle) Distribution of back-propagated gradients associated with each PDE loss term. (Right) Convergence rates for each PDE. For further explanation of these concepts, see Appendix 6.6.	165
6.3	Loss landscape analysis. This figure visualizes the local geometry of the loss landscape when training a PINN on a coupled system of two PDEs versus a single PDE, using a two-dimensional projection method around a local minimum (Im <i>et al.</i> , 2016). Left: Projected loss landscape for a PINN trained on the coupled water-solute transport problem in unsaturated soils. Right: Projected loss landscape for a PINN trained only on the water infiltration PDE.	166
6.4	PDE weights as a function of the training step for three PINN variants applied to a one-way coupled system of three equations. Each equation is trained for 100 iterations. The standard PINN uses constant (unity) weights, while S-PINN and BC-PINN employ iteration-dependent weights. See the corresponding sections for details on each solver.	174

6.5	Smooth vs. Discontinuous weights: Comparison of sequential PINN PDE weights. The red dashed lines show the discontinuous weights in S-PINN, while the smooth curves represent 2S-PINN. From left to right: weights for equation 1, a middle equation (equation 2 to $N - 1$), and equation N	176
6.6	Illustration of a system of three one-way coupled PDEs, where each equation is approximated by a neural network with parameters θ_i . The figure shows, for each solver, whether the parameters are trainable or frozen during training.	178
6.7	Thermo-mechanics coupling problem. Evolution of relative L^2 -errors for each PINN solver, bottom: Final relative L^2 -error values. The left panels display results for ODE 1, while the right panels show results for ODE 2. For each PINN solver, we report the minimum, maximum, and mean values across ten independent runs.	180
6.8	Electro-thermal coupling problem. Top: Evolution of relative L^2 -errors for each PINN solver, bottom: Final relative L^2 -error values. The left panels display results for PDE 1, while the right panels show results for PDE 2. For each PINN solver, we report the minimum, maximum, and mean values across five independent runs.	182
6.9	Infiltration-transport in unsaturated soils. Loss and PDE weight evolution for different PINN solvers. The results are presented by showing all five runs and the mean values for the losses of PDE 1, PDE 2, and the total loss.	184
6.10	Infiltration-transport in unsaturated soils. Top: Evolution of relative L^2 -errors for each PINN solver, bottom: Final relative L^2 -error values. The left panels display results for PDE 1, while the right panels show results for PDE 2. For each PINN solver, we report the minimum, maximum, and mean values across five independent runs.	185
6.11	Infiltration and multispecies solute transport problem. Left: Linear-scale comparison of relative L^2 -errors for the standard PINN and S-PINN solvers, with and without architectural and sampling enhancements. Right: Log-scale representation of the same data highlighting relative performance differences across PDEs 1–4.	186
6.12	Infiltration and multispecies solute transport problem. Evolution of the total loss and relative L^2 -error over five independent runs. The plot shows the mean values for PDEs 1 to 4.	187
6.13	Infiltration and multi-species solute transport problem. Comparison of PINN solvers in terms of L^2 -error.	188

6.14	Computational efficiency comparison for the infiltration and multispecies solute transport problem. The bar chart illustrates the CPU time (in seconds) required for various PINN solvers.	189
7.1	Constitutive functions of the BCJ-R equation (7.7). (a) relative permeability $k_r(S)$, (b) Leverett J -function $J(S)$, and (c) capillary pinning function $\kappa(S)$. The following parameter values are used to evaluate these functions: $\phi = 0.3$, $K_s = 0.4$, $h_{\text{cap}} = 0.02$, $a = 3.0$, $v_e = 1.0$, $\alpha = 10.0$, $\beta = 40.0$, and $\delta = 0.01$	198
7.2	Schematic representation of the PINN framework for solving the BCJ-R equation of unstable infiltration in porous media.	206
7.3	Saturation profiles for different initial conditions. Each panel shows the FEM reference, PINN prediction, and absolute error in the time-space domain, along with cross-sectional comparisons at selected times.	209
7.4	Entropy test analysis. (a) Saturation profiles during water redistribution in a closed 1D soil system. (b) Evolution of the difference entropy, $\mathcal{E}(t) - \mathcal{E}(0)$, which increases monotonically over time.	210
7.5	Model parameter fitting and validation: (a) Parameter optimization results showing the fit between the van Genuchten model and the generalized Brooks-Corey model. The optimization yields $h_{\text{cap}} = 4.8$ cm, $\alpha = 4.4$, and $\beta = 100$. (b) PINN model validation against DiCarlo experimental data (DiCarlo, 2004). Comparison of predicted saturation profiles with laboratory measurements under four different water flux inlets.	212
7.6	Comparison of PINN model performance using different architectures (MLP, Modified MLP (Wang <i>et al.</i> , 2021a), and PirateNet (Wang <i>et al.</i> , 2024a)) for the BCJ-R equation. Metrics shown include the L_2 -error, total loss, and CPU time, with all features enabled and the SOAP optimizer used across all experiments.	217
7.7	Weak and strong scaling efficiencies of the data-parallel training strategy. We use an MLP architecture with all features enabled and measure the execution time over 10,000 Adam iterations. For weak scaling, the number of residual points increases proportionally with the number of GPUs ($n \times 4096$), while for strong scaling, the total number of residual points is fixed at 8192.	218

8.1	Constitutive relationships for three soil types (loam, sandy loam, and sandy soil) generated using the Mualem-van Genuchten model, displayed on a log-log scale. The left panel shows the unsaturated hydraulic conductivity (K) as a function of pressure head (Ψ), while the right panel presents the water content (θ) as a function of pressure head (Ψ). Soil parameters used in the model are provided in Table 8.1.	226
8.2	Two-dimensional soil geometry configurations used in this study. The parameters are: $L_x = L_z = 1$ m for geometry 1, and $L_x = L_z = 0.5$ m, $r_0 = 5$ cm for geometry 2. The final simulation times are $T = 24$ hours for geometry 1 and $T = 6$ hours for geometry 2. The soil geometries are represented as rectangular domains, with boundary types indicated using specific colors.	228
8.3	The flowchart of the DeepONet for modeling water flow in unsaturated soils.	230
8.4	(a) Visualization of collocation points for DeepONet. (b) Unstructured mesh employed in the FEM reference solution. Both correspond to Geometry 2.	237
8.5	Scenario (a)-non-parametric /geometry 1. Top: 2D snapshots from DeepONet and FEM at different times for sandy and sandy loam soils. Bottom: Cross-sections at $x = 0.5$ m at different times for loam, sandy loam, and sandy soils.	239
8.6	Scenario (a)-non-parametric /geometry 2. Top: 2D snapshots from DeepONet and FEM at different times for sandy soil and sandy loam soil. Bottom: Cross-sections at $x = 0.5$ m at different times for loam, sandy loam, and sandy soils.	240
8.7	Scenario (a)-non-parametric : The relative L^2 -error of DeepONet for each soil type and geometry. We plot both the accuracy before and after fine-tuning the pre-trained DeepONet with physics. . .	241
8.8	Scenario (a)-non-parametric : Comparison of soil moisture cross-sections from DeepONet (with and without fine-tuning) and the FEM reference solution. The cross-section is taken from Geometry 1 at position $x = 0.5$ m, observed at the final simulation time of 1 day.	242
8.9	Scenario (a)-parametric: 2D snapshots from DeepONet and FEM at the final simulation time for loam, sandy loam, and sandy soils. The figures include two samples from the training data and two samples from the test data. Each panel shows the initial condition at the bottom, followed by the predictions from the FEM and DeepONet models.	243

8.10	Scenario (a)-parametric: Results from DeepONet in an extrapolation scenario for loam soil in geometry 1, with and without fine-tuning. The initial soil moisture varies linearly from the surface to the bottom (left). Soil moisture profiles from FEM at $T = 1$ day (middle). Cross-sections from FEM and DeepONet with and without fine-tuning (right).	244
8.11	Scenario (b)-time-independent water flux: 2D snapshots from DeepONet and the FEM reference solution at the final simulation time for loamy soil. Each snapshot includes one surface water flux from the training data and one from the test data for each geometry. The solid line represents DeepONet, while the red circles represent the FEM solution in the cross-sections.	246
8.12	Scenario (b)-pulse duration: 2D snapshots along with cross-sections from DeepONet and FEM at the final simulation times for geometry 1 for the loamy soil. The plots include one sample from the training data and two samples of test data.	247
8.13	Scenario (b)-drying-wetting cycle with different fluxes: 2D snapshots from DeepONet and FEM at the final time of simulation for loamy soil and geometry 1, including two surface water fluxes from training data and two surface water fluxes from test data.	249
8.14	Scenario (b)-drying-wetting cycle with varying fluxes: Temporal variation of soil moisture and water flux at observation depths of $z = 0$ cm (observation points $\mathbf{x}_1$) and $z = 5$ cm (observation points $\mathbf{x}_2$) for training, test, and extrapolation surface water flux scenarios.	250
8.15	The two scenarios explored in this study: four snapshots for scenario (a) and three snapshots for scenario (b) used for training the DeepONet model.	255
8.16	The total loss of DeepONet for four parametrizations used in both scenarios.	256
8.17	CPU time comparison of FEM and DeepONet solutions for Richards equation across varying initial soil moisture conditions.	256

CHAPTER 1 Introduction

1.1 Background and motivation

1.1.1 Water scarcity, fertilizer use, and environmental challenges

Agriculture is a fundamental sector on which human life depends, as it plays a crucial role in ensuring sufficient food supply for the rapidly growing global population. However, it is also the most water-dependent sector, relying heavily on freshwater resources. This dependence makes agricultural sustainability particularly vulnerable to water scarcity and climate variability. According to the Food and Agriculture Organization (FAO, 2021), nearly 70% of global freshwater withdrawals are dedicated to agricultural activities, while only about 10% is allocated to domestic use and the remainder to industry (Morianou *et al.*, 2023). In the agricultural sector, water is essential for a wide range of activities, including crop irrigation, livestock care, food processing, and the maintenance of soil quality.

The increasing global demand for food has intensified the water requirements of modern agricultural practices, exacerbating pressure on already scarce water resources. Water scarcity is particularly severe in arid and semi-arid regions, where frequent droughts and low rainfall lead to extremely limited water availability. Qadem *et al.* (2025) reported that drought conditions in the semi-arid areas of Morocco have intensified in the last few decades, with a clear increase in the frequency and severity of dry periods and a statistically significant decrease in precipitation between 1979 and 2015. This highlights the ongoing need to improve the efficiency and sustainability of water use, especially in the agricultural sector.

To improve crop yields and meet the demands of a rapidly growing global population, researchers and practitioners have increasingly focused on optimizing the combined use of water and fertilizers in agriculture. For healthy growth and maximum productivity, plants require adequate water and timely access to essential nutrients (Brendel, 2021). Irrigation refers to the process of supplying water to crops in a controlled manner, while fertigation denotes the application of fertilizers through irrigation systems to deliver nutrients directly to the root zone. Fertilizers, usually applied in liquid or powder form, provide essential elements such as nitrogen, phosphorus, potassium, sulfur, and other micronutrients. However, maximizing the efficiency of both water and fertilizer use remains a major challenge, as significant losses occur through evaporation, surface runoff, and deep percolation. These constraints highlight the urgent need for advanced irrigation and fertigation strategies that deliver water and nutrients precisely where and when crops require them. Without such control, excessive fertilizer applications not only reduce resource efficiency but also pose serious environmental and health risks, including soil ecosystem degradation and contamination of water resources. One major challenge is the efficient delivery of fertilizers without harming the surrounding

soil environment, including microbial communities such as bacteria and fungi, or contaminating groundwater. Among all nutrients, nitrate (NO_3^-) is of particular concern because it is a highly mobile anion that leaches easily beyond the root zone (Muhammad *et al.*, 2022). Its low retention in soil allows it to travel with percolating water, making it one of the most leachable and groundwater-sensitive nutrients. As a result, nitrate often accumulates in groundwater, sometimes exceeding the safe concentration limits for drinking water, which are set at 10 mg/L as nitrogen or 45 mg/L as NO_3^- (Zhou *et al.*, 2015). This poses a serious concern, especially since groundwater supplies nearly half of all drinking water worldwide (Zhou *et al.*, 2015).

1.1.2 Mathematical modeling of irrigation and fertigation strategies

To enhance the efficiency of resource use, various irrigation and fertigation strategies have been developed and evaluated. These include surface irrigation, flood irrigation, subsurface irrigation, and micro-irrigation techniques such as drip and sprinkler systems. Such systems are designed to minimize water and fertilizer losses by delivering inputs directly to the plant roots, thereby increasing water and nutrient use efficiency and reducing leaching into deeper soil layers (Gärdenäs *et al.*, 2005). Precision irrigation methods are especially valuable in water-scarce regions, where sustainable agricultural practices are crucial. In general, determining the optimal irrigation and fertigation strategy for a specific crop and region requires extensive field and experimental studies. Many studies have been conducted to assess the effectiveness of different irrigation and fertigation systems under varying soil, climate, and crop conditions (see e.g., Gärdenäs *et al.* (2005); Liu *et al.* (2019); Poch-Massegú *et al.* (2014); Rajput and Patel (2006); Russo and Kurtzman (2019)). However, a key limitation of these approaches is the time, cost, and labor involved in conducting large-scale field experiments. Testing every possible combination of irrigation schedule, nutrient dose, and system layout is often impractical due to resource constraints and the urgent need to implement irrigation systems during growing seasons.

Mathematical models provide an efficient and robust framework for analyzing water and nutrient behavior in porous media. Unlike empirical approaches, these models are grounded in physical laws and can represent the coupled processes of flow, transport, and reactions within soils. They are capable of predicting the spatial distribution and the temporal evolution of water saturation as well as solute concentrations throughout the soil profile. This predictive ability allows researchers and practitioners to assess how different irrigation and fertigation practices influence water availability, nutrient uptake, and potential losses. By simulating different management scenarios, these models can be used to identify efficient irrigation and fertigation strategies that improve the use of water and nutrients while reducing environmental risks. Such simulations reduce the reliance on costly and time-consuming field and laboratory experiments, offering a more flexible framework for decision-making across diverse crops, soil types, and climatic conditions.

In this thesis project, we focus on continuum mathematical models that describe key processes such as water infiltration and fertilizer transport in porous media. These processes are typically governed by partial differential equations (PDEs) with well-defined initial and boundary conditions, forming the foundation for predictive modeling. By accurately representing the underlying physical and chemical dynamics in soil systems, PDE-based models offer a robust framework for evaluating and optimizing irrigation and fertigation strategies across diverse agricultural scenarios. In the forward modeling setting, where initial/boundary conditions and physical parameters (e.g., soil properties such as porosity, bulk density, hydraulic conductivity functions, and capillary pressure models, as well as solute parameters such as diffusion coefficients and kinetic reaction terms) are known, the models can directly predict quantities of interest, such as soil water saturation, pressure head, and solute concentrations for different chemical species. In inverse problems, limited and noisy observational data are used to infer a suitable set of unknown parameters that yield model outputs consistent with the observations. These calibrated models can then be validated and employed for other predictive tasks.

1.1.3 Modeling water and solute transport in porous media

Understanding how water and dissolved substances move through porous media is essential for designing efficient irrigation and fertigation strategies. The subsurface zone where water saturation is less than the soil porosity, located between the soil surface and the groundwater table, includes the plant root zone and serves as the primary domain of interest in agricultural water management. Accurately modeling the flow of water and the transport of fertilizers within this zone is crucial for ensuring that crops receive sufficient nutrients while minimizing resource losses and environmental contamination.

1.1.3.1 Water flow models

Many mathematical models have been developed to describe water infiltration in porous media. One of the most widely used is the Richards equation ([Richards, 1931](#)), which accounts for water movement driven by both gravitational and capillary forces. This equation captures the nonlinear relationship between soil moisture content and hydraulic conductivity, making it well-suited for modeling unsaturated flow conditions. In soil hydrology and agro-environmental modeling, the Richards equation is often coupled with the advection–dispersion–reaction (ADR) equations ([Bear, 2013](#)) to simulate the transport of dissolved fertilizers and other solutes within the root zone. This coupled Richards–ADR system is inherently complex and can be extended to incorporate a wide range of physical and biochemical processes, including root water and nutrient uptake, hysteresis effects, nutrient adsorption and degradation, preferential flow pathways, and multiphase flow dynamics ([Šimůnek *et al.*, 2005](#)).

To model water infiltration in unsaturated porous media, several continuum models

have been developed in addition to the classical Richards equation to account for preferential flow phenomena such as unstable infiltration, saturation overshoot, and gravity-driven fingering. These processes have been extensively documented in experimental studies (e.g., DiCarlo (2004); Glass *et al.* (1989)). To better capture these complex flow patterns, several extensions and modifications of the Richards equation have been proposed (see, e.g., DiCarlo (2013); Xiong (2014); Cueto-Felgueroso and Juanes (2009); Eliassi and Glass (2002); Beljadid *et al.* (2020)). These continuum models aim to improve the representation of non-equilibrium effects, phase separation, and dynamic capillary behavior in porous media. Beljadid, Cueto-Felgueroso, and Juanes (Beljadid *et al.*, 2020) introduced a continuum model to describe unstable infiltration in porous media. They formulated a new PDE, referred to as the BCJ-R equation (see Equation 18 in (Beljadid *et al.*, 2020)), which extends the classical Richards equation by incorporating an entropy function. This formulation is grounded in second gradient theory and is derived from a free energy functional that symmetrically expands the capillary energy density term present in the traditional Richards equation.

1.1.3.2 Solute transport models

For solute transport in porous media, various physical and chemical processes can be explicitly modeled within the ADR framework. For instance, sorption of solutes to soil particles is often described using nonlinear isotherms such as those of Langmuir or Freundlich type (Wang *et al.*, 2017), which are widely used to represent phosphorus retention. Phosphorus immobilization can also be modeled through additional reaction terms accounting for microbial uptake and precipitation with soil minerals. Moreover, root water and nutrient uptake can be incorporated using sink terms based on plant stress response models, such as the Feddes model (Feddes *et al.*, 1976), which links uptake rates to soil moisture conditions.

Several mathematical models have been developed to extend the standard equilibrium ADR equation in order to account for nonequilibrium physical and chemical processes (Šimůnek *et al.*, 2005; Šimůnek and van Genuchten, 2008). Nonequilibrium physical models assume the existence of two distinct liquid regions: a mobile zone, where water and solutes move, and an immobile zone, where water remains stagnant. Solute exchange between these zones is modeled using formulations such as the mobile-immobile model, dual-porosity model, or dual-permeability model. On the other hand, nonequilibrium chemical models account for kinetically controlled reactions, such as time-dependent sorption and desorption processes, rather than assuming instantaneous equilibrium. These models are particularly relevant for reactive solute transport involving complex geochemical interactions. For a comprehensive overview of these models and their implementation, we refer the reader to (Šimůnek and van Genuchten, 2008).

To design effective fertigation strategies, it is essential to simultaneously track multiple interacting solute species. For example, in the case of NPK fertilizers (e.g.,

urea–nitrogen, phosphorus, and potassium), each solute is typically described by its own ADR equation. Urea undergoes urease-catalyzed hydrolysis to ammonium, which is further transformed through nitrification ($\text{CO}(\text{NH}_2)_2 \rightarrow \text{NH}_4^+ \rightarrow \text{NO}_2^- \rightarrow \text{NO}_3^-$) and, under anaerobic conditions, through denitrification into gaseous nitrogen products. These transformations are commonly modeled using zero or first-order kinetics with rate modifiers dependent on soil water content, temperature, and pH (Šimůnek *et al.*, 2005).

1.1.3.3 Modeling challenges and computational complexity

Modeling water infiltration and the transport of multiple solutes using mathematical models remains a highly challenging task due to several factors. From a numerical standpoint, one of the main difficulties arises from the infiltration process itself, which is governed by a highly nonlinear and degenerate PDE. This often leads to the formation of sharp saturation fronts, particularly in coarse-textured soils where gravitational forces dominate over capillary forces. A further challenge lies in the multiphysics nature of the problem: infiltration dynamics must be coupled with solute transport equations to accurately predict fertilizer concentrations within the soil–plant continuum. The complexity grows significantly when multiple interacting solutes are considered. In addition, determining optimal irrigation and fertigation strategies requires running a large number of simulations under varying soil properties, initial conditions, and boundary conditions to capture their impact on water and nutrient dynamics. This makes the process computationally intensive and time-consuming. As a result, the development of efficient and practical algorithms for solving such models remains an active and demanding area of research.

1.1.4 Thesis motivation

The primary motivation of this thesis is to develop efficient and accurate predictive algorithms for modeling water infiltration and the transport of single- and multi-species solutes in porous media. The problem is inherently challenging because infiltration processes are governed by highly nonlinear PDEs. In particular, the Richards equation captures unsaturated flow, while the more advanced BCJ–R model addresses unstable infiltration phenomena such as fingering and saturation overshoot. Beyond water flow, realistic scenarios require the coupling of these infiltration models with advection–dispersion–reaction systems that describe the transport, transformation, and degradation of multiple interacting solutes. Inverse modeling adds another layer of complexity: algorithms must infer soil hydraulic and solute parameters from sparse and noisy data while maintaining predictive reliability. Moreover, effective irrigation and fertigation management requires repeated simulations under varying soil properties, boundary conditions, and application strategies, which makes computational efficiency a critical requirement. Traditional numerical solvers become prohibitively

expensive under such demands. The overarching goal of this thesis is to close these gaps by integrating finite element methods with physics-informed machine learning surrogates and operator learning techniques. The objective is to create scalable and data-efficient algorithms capable of handling forward and inverse problems in water flow and solute transport, enabling robust predictions across a wide range of soil, crop, and environmental conditions.

Modeling challenges. The development of efficient predictive models for the studied problem is challenged by the physical complexity of the governing equations and the numerical difficulties involved in their solution. The main challenges include:

1. **Richards equation.** The Richards equation is a highly nonlinear PDE, and its solution often develops steep gradients near the wetting front. In gravity-dominated regimes, the problem becomes advection-dominated, leading to numerical instability. Standard numerical schemes typically rely on iterative solvers, which may suffer from convergence issues. In heterogeneous porous media, such as layered soils, discontinuities in hydraulic properties result in sharp transitions in water saturation at layer interfaces. Accurately capturing these transitions requires specialized numerical treatments, as naive discretizations may produce spurious oscillations or loss of accuracy.
2. **BCJ-R model.** The BCJ-R model introduces a high-order nonlinear PDE to describe complex infiltration behaviors such as saturation overshoot and fingering flow. Solving this equation requires high-order spatial and temporal discretization schemes to accurately resolve both low- and high-frequency solution components. This increases the computational burden and places strict requirements on solver stability and resolution.
3. **Multi-species solute transport:** Coupled ADR equations for multi-species solute transport often exhibit stiff behavior due to varying transport and reaction timescales. Numerical errors in one equation can propagate through the coupled system, compromising overall stability and accuracy. The multiphysics nature of these systems introduces self-scaled transport regimes for each species, governed by distinct diffusion, advection, and reaction dynamics. As a result, the predictive algorithm must be capable of handling disparate scales and interactions simultaneously to ensure accurate and stable solutions.
4. **Inverse problems.** Estimating unknown parameters (e.g., soil hydraulic or transport properties) from limited and noisy measurement data leads to ill-posed inverse problems. Robust surrogate models must generalize well under data scarcity while maintaining predictive accuracy and physical consistency.
5. **Computational cost in multi-query settings.** For tasks involving repeated simulations under varying input conditions, such as optimal irrigation design or

uncertainty quantification, traditional numerical solvers are often computationally prohibitive. This motivates the development of efficient surrogate models that can provide rapid, accurate predictions across a wide range of scenarios to support real-time and multi-task decision-making.

1.2 Mathematical models

In the following, we present the mathematical models that govern water infiltration and solute transport in porous media. Water flow in unsaturated soils is primarily described by the Richards equation, which accounts for both capillary and gravitational forces. To capture unstable infiltration phenomena, such as saturation overshoot or gravity-driven fingering, we adopt the BCJ-R equation proposed by Beljadid *et al.* (2020), which extends the classical Richards formulation. We refer to

For solute transport, we use the ADR equations (Bear, 2013), which describe the movement and transformation of dissolved substances in the soil. All models are formulated with appropriate initial and boundary conditions to reflect site-specific flow and transport scenarios.

For these formulations, we denote by Ω the computational domain representing the porous medium, t the time coordinate, and $\mathbf{x} = (x, z)$ the spatial coordinates, where x is the horizontal direction and z is the vertical direction oriented upward. The time horizon for infiltration and transport is denoted by $I = (0, T]$, where T is the final simulation time. In forward modeling, initial and boundary conditions as well as model parameters are assumed to be known, whereas in inverse settings certain parameters or boundary conditions must be inferred from available observational data.

1.2.1 Water infiltration

1.2.1.1 Local model: Stable wetting front

The Richards equation (Richards, 1931) is a time-dependent nonlinear advection-diffusion PDE that is used to model water infiltration into unsaturated porous media. It is derived by combining the mass conservation principle and Darcy-Buckingham's law (Buckingham, 1907). The expression of the Richards equation is as follows:

$$\begin{cases} \frac{\partial \theta}{\partial t} + \nabla \cdot \mathbf{q} = 0, \\ \mathbf{q} = -K \nabla (\Psi + z), \end{cases} \quad (1.1)$$

where Ψ is the matric head, $\mathbf{q}$ is the water flux, $K = K_s k_r$ is the unsaturated hydraulic conductivity, K_s is the saturated hydraulic conductivity, k_r is the water relative permeability, and θ is the volumetric water content.

To fully define the Richards equation, it is necessary to specify the soil hydraulic properties in closed form: the water retention curve, which relates the volumetric water content θ to the pressure head Ψ , and the hydraulic conductivity function, which

describes the dependence of unsaturated hydraulic conductivity on Ψ . Subsequent chapters will provide a detailed discussion of the Richards equation.

1.2.1.2 Nonlocal model: Unstable wetting front

To simulate unstable infiltration phenomena in porous media, we adopt the BCJ-R equation proposed by Beljadid, Cueto-Felgueroso, and Juanes (Beljadid *et al.*, 2020). This PDE is endowed with an entropy function to ensure thermodynamic consistency and captures key instability-driven phenomena such as saturation overshoot in 1D and fingered flow in higher dimensions. The governing equation is a high-order PDE given by (Beljadid *et al.*, 2020):

$$\phi \frac{\partial S}{\partial t} + \nabla \cdot \left[K(S) \nabla \left(z + \psi(S) + \sqrt{\kappa(S)} \nabla \cdot \left(\sqrt{\kappa(S)} \nabla S \right) \right) \right] = 0, \quad (1.2)$$

where S is the water saturation, ϕ is the medium porosity, K represents the unsaturated hydraulic conductivity, ψ is the water suction head, and κ represents the capillary pinning function. More details on the BCJ-R equation will be presented in Chapter 7.

1.2.2 Solute transport

In unsaturated porous media, the coupled water flow and solute transport system is generally represented by the ADR equation coupled with an infiltration model. The governing ADR equation takes the following form (Bear, 2013; Šimůnek *et al.*, 2005):

$$\frac{\partial(\theta c)}{\partial t} + \frac{\partial(\rho s)}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c - c \mathbf{q}), \quad (1.3)$$

where c is the solute concentration in the liquid phase, s is the concentration of solute adsorbed to the solid matrix, ρ denotes the soil bulk density, and $\mathbf{D}$ is the dispersion tensor, which accounts for both molecular diffusion and hydrodynamic dispersion. The terms θ and $\mathbf{q}$ represent the volumetric water content and Darcy water flux, respectively, as computed from the Richards equation (5.1).

More details on the solute transport model will be provided in subsequent chapters.

1.3 Numerical models

1.3.1 Standard numerical methods

Numerical methods are the most common approach for approximating the solution of well-posed differential equations that arise in modeling physical systems. In the context of subsurface hydrology, a wide range of numerical techniques have been developed to simulate water infiltration and solute transport in porous media. In the following sections, we present a literature review of the key numerical methods used to solve these problems, with a focus on their application to modeling water infiltration and fertilizer movement in the root zone as well as their limitations.

1.3.1.1 Literature review

Numerical modeling of the Richards equation has been the subject of extensive research for decades, primarily due to its inherent complexity and the fact that exact analytical solutions are only available for simplified scenarios, not for realistic field conditions. It is mathematically characterized as a degenerate, nonlinear, elliptic–parabolic PDE of advection–diffusion type. It transitions between parabolic and degenerate elliptic behavior, depending on soil moisture conditions. This dual nature arises because the equation degenerates (loses its parabolic character) when the soil becomes fully saturated (Farthing and Ogden, 2017). This complexity presents significant challenges for numerical approximation, especially in soils where the relationships between matric potential, water saturation, and hydraulic conductivity exhibit strong nonlinearity. Consequently, the development of accurate and stable numerical schemes is crucial for accurately simulating water flow in porous media using the Richards equation.

Many numerical methods have been proposed to solve the Richards equation, with the aim of developing efficient time-stepping schemes and accurate spatial discretizations of the underlying PDE. Spatial approximations encompass a variety of techniques, including finite difference methods (Chávez-Negrete *et al.*, 2018; Celia *et al.*, 1990a; Fengnan *et al.*, 2020), finite element methods (Celia *et al.*, 1990a; Keita *et al.*, 2021b; Bergamaschi and Putti, 1999a; Kamil *et al.*, 2024a; Fahs *et al.*, 2009; Toutlini *et al.*, 2025a, 2024; List and Radu, 2016), finite volume methods (Manzini and Ferraris, 2004; Casulli and Zanolli, 2010), meshless techniques (Boujoudar *et al.*, 2021a,b; Stevens and Power, 2011), and spectral methods (Difonzo and Pellegrino, 2024), among others. Each approach offers specific advantages and faces certain limitations regarding accuracy, stability, and computational efficiency, with performance often dependent on the characteristics of the soil and the imposed boundary conditions. Regarding temporal discretization, the most commonly used approach is the first-order implicit (backward) Euler scheme, due to its stability and robustness for stiff, nonlinear problems. However, the use of higher-order time integration methods remains limited in practice (Kamil *et al.*, 2024b; Keita *et al.*, 2021b; Toutlini *et al.*, 2025a), despite their potential to improve temporal accuracy and efficiency.

Several studies have addressed the challenge of solving the coupled system of equations governing water flow and solute transport by adopting either deterministic approaches (Antonopoulos, 1993; Chen *et al.*, 2019a; Haruzi and Moreno, 2023c; Illiano *et al.*, 2021; Kaluarachchi and Parker, 1988; Kamil *et al.*, 2024c; Selim and Iskandar, 1981; Srivastava and Yeh, 1992; Kamil *et al.*, 2024e) or stochastic formulations (Russo, 2012b; Russo *et al.*, 1998, 2001). To solve these coupled systems, implicit time-stepping schemes are often preferred due to their stability and ability to handle stiff nonlinearities, especially under saturated–unsaturated transitions. Various implicit schemes have been proposed and analyzed in the literature (Diaw *et al.*, 2001; Radu *et al.*, 2008; Radu and Pop, 2010; Radu *et al.*, 2010, 2006). For instance, Diaw *et al.* (2001) proposed a stan-

dard finite element formulation with primary variable switching to solve the Richards equation, along with operator splitting and a discontinuous finite element method for modeling non-sorptive solute transport. [Mitchell and Mayer \(1998\)](#) addressed the coupled problem using a standard finite element method for water flow, combined with an Eulerian–Lagrangian scheme to solve the solute transport equation. The influential work of Radu et al. ([Radu et al., 2008](#); [Radu and Pop, 2010](#); [Radu et al., 2010, 2006](#)) further advanced this area by providing theoretical analysis and numerical schemes based on the implicit mixed finite element method. Their work includes the use of Newton linearization techniques and various numerical treatments for advection and diffusion terms.

Many commercial and open-source software tools have been developed based on traditional numerical methods to simulate water flow and solute transport in unsaturated soils. Among them, HYDRUS ([Šimůnek et al., 2005](#)) is one of the most widely used and established modeling platforms. HYDRUS is open-source for one-dimensional simulations and commercially available for two- and three-dimensional cases. It is built on the finite element method for spatial discretization and typically uses a first-order implicit Euler scheme for time integration. HYDRUS is capable of modeling a broad range of processes relevant to soil physics and vadose zone hydrology. These include variably saturated water flow, heat transfer, and the transport of single or multiple solute species. The software supports complex chemical reactions, such as sorption, decay, and transformation of nitrogen species, making it suitable for agricultural and environmental applications. In addition, HYDRUS includes modules for root water uptake, plant–soil interactions, and colloid-facilitated transport, which make it highly versatile for simulating realistic soil–plant–atmosphere systems. The application of HYDRUS to real-world case studies has been widely explored by many researchers. For recent developments and updates in HYDRUS, we refer the reader to [Šimůnek et al. \(2016\)](#), and for a comprehensive overview of its applications in the field of vadose zone modeling, we recommend the literature review by [Morianou et al. \(2023\)](#).

1.3.1.2 Limitations

For forward modeling of coupled water flow and solute transport in unsaturated soils, traditional numerical methods remain the primary choice due to their maturity and decades of development. However, these methods become increasingly intractable when applied to inverse problems, such as the estimation of unknown soil parameters. In such cases, parameters must be calibrated using in-situ measurements of the unsaturated hydraulic properties of soil ([Šimůnek and Van Genuchten, 1996](#)). A common approach involves coupling the numerical solver with an optimization algorithm in a feedback loop to iteratively adjust model parameters until a good fit is achieved. However, this process typically requires running the numerical solver many times, which becomes computationally expensive, particularly when dealing with highly nonlinear soil capillary pressure models and a large number of parameters to estimate in inverse problems.

In such scenarios, convergence is often poor, and the method may even become unstable or diverge entirely (Šimůnek and Van Genuchten, 1996). Moreover, traditional numerical solvers are generally task-specific. Any modification to model parameters, such as changes in initial or boundary conditions, soil types, requires rerunning the entire solver. This lack of flexibility limits their applicability in multi-task learning settings, where multiple scenarios (e.g., different irrigation or fertigation strategies) must be evaluated rapidly. Another significant limitation is the computational cost associated with running large-scale or high-resolution simulations, especially in higher-dimensional problems. The so-called curse of dimensionality makes it increasingly difficult to scale traditional methods efficiently, particularly when fast feedback is needed for real-time decision-making in agricultural water management.

Traditional methods also face significant challenges related to scalability. As the dimensionality of the problem increases due to finer spatial discretizations, longer simulation periods, or the inclusion of additional coupled solute species, the computational cost grows rapidly, and numerical stability may deteriorate, leading to reduced accuracy. This curse of dimensionality makes it difficult to apply conventional solvers to large-scale or high-resolution problems within reasonable computational limits. Furthermore, traditional numerical solvers have limited capability to incorporate real-world data. Integrating real-time sensor measurements or remote sensing observations typically requires dedicated data assimilation frameworks, which are often complex to develop and computationally expensive to execute.

These challenges highlight the urgent need for alternative surrogate models that are fast and accurate, particularly for real-time decision-making in irrigation and fertigation management.

1.3.2 Standard machine learning

The widespread adoption of deep learning (DL) algorithms for function approximation stems from the fact that artificial neural networks are powerful parametric models capable of approximating any function to arbitrary accuracy under suitable assumptions (Hornik, 1991; Llanas *et al.*, 2008). These DL algorithms differ fundamentally from traditional numerical methods in that they learn the solution through an optimization process, rather than relying on predefined discretization rules. Neural networks act as black-box learners that automatically adapt to a given task, making them highly suitable for a wide range of engineering applications.

Standard machine learning (ML) methods aim to learn patterns from data without incorporating prior knowledge of the underlying physical laws. ML models include supervised learning techniques such as regression and classification, using algorithms such as support vector machines, decision trees, ensemble methods, and neural networks. There is also unsupervised learning, which covers tasks such as dimensionality reduction, clustering, and anomaly detection. In addition, reinforcement learning has been explored in sequential decision-making problems, where an agent learns optimal

strategies through interaction with the environment. In the context of hydrological modeling, ML has been applied to tasks such as rainfall-runoff prediction, ground-water level forecasting, and streamflow estimation. The main idea is to learn the input-output relationship directly from observational data, which can be effective in data-rich scenarios.

1.3.2.1 Literature review

Machine learning techniques have a long history in hydrological modeling and continue to play an important role in advancing data-driven predictive modeling. One of the first applications involves the development of data-driven pedotransfer functions to infer soil hydraulic properties, such as water retention curves and hydraulic conductivity, from measured soil attributes such as texture, bulk density, and organic matter content (Li *et al.*, 2024b; Pham *et al.*, 2023). Beyond soil property estimation, machine learning models have also been used to capture the temporal dynamics of soil moisture under varying water-stress conditions (Karandish and Šimůnek, 2016), to estimate volumetric soil moisture content from field measurements or remote sensing data (Ahmad *et al.*, 2010; Moghadas and Badorreck, 2019; Aboutalebi *et al.*, 2019), and to retrieve combined water and salt contents directly from field-scale observations (Wan *et al.*, 2023). Additional studies have explored the simultaneous prediction of multiple soil attributes, including nitrogen, and organic carbon, using multispectral imaging and proximal sensing technologies (Morellos *et al.*, 2016).

Smart agriculture refers to the integration of intelligent, self-learning systems into agricultural practices to enable efficient crop management, optimized resource usage, and the automation of processes such as irrigation. These technologies leverage real-time data collected from various sensors, monitoring soil moisture, weather conditions, crop health, and more, to support informed decision-making and adaptive control strategies. The core objective of smart irrigation is to utilize sensor-derived intelligence to optimize water usage, enhance crop yields, and minimize environmental impact. By dynamically adjusting irrigation schedules based on actual field conditions rather than static timetables, such systems contribute to sustainability and operational efficiency. A comprehensive review by Vallejo-Gomez *et al.* (2023) highlights recent developments in this area, covering the role of machine learning models in improving irrigation decisions, system architectures, and deployment challenges.

1.3.2.2 Limitations

Standard machine learning (ML) models often fail to generalize beyond the training domain and may produce physically inconsistent results, particularly in systems with complex spatiotemporal dynamics. These models typically require large, diverse, and high-quality datasets, which are rarely available in hydrological applications due to data scarcity and measurement noise, especially in subsurface environments. Karandish and

[Šimůnek \(2016\)](#) quantitatively compared a physics-based numerical model (the finite element code HYDRUS) with various data-driven machine learning models (multiple linear regression, adaptive neuro-fuzzy inference systems, and support vector machines) for simulating time series of soil water content under water stress conditions. Their results showed that HYDRUS outperformed the purely data-driven models, which also performed poorly for soil water content forecasting.

This naturally raises the question: *What if machine learning models were designed to be physically consistent?* Addressing this question motivates the use of scientific machine learning methods, which embed physical knowledge into the learning process through physics-based constraints, thereby enabling more robust and data-efficient modeling of real-world systems.

1.3.3 Scientific machine learning

Scientific machine learning (SciML) is an emerging interdisciplinary field at the intersection of scientific computing and machine learning. The central objective of SciML is to integrate physical laws with data-driven algorithms to construct accurate and generalizable surrogate models. Related research areas include neural networks for differential equations and AI for science (AI4Science). A comprehensive survey can be found in ([Huang et al., 2025](#)).

1.3.3.1 Physics-informed neural networks

The concept of approximating the solutions of differential equations using neural networks dates back to the early 1990s ([Lagaris et al., 1998](#)). The fundamental idea is to represent the solution with a neural network and train it to satisfy the governing equations up to a prescribed tolerance. Different strategies have been proposed for this purpose. One approach discretizes the differential equation and employs the neural network to approximate the solution at selected spatial or temporal points. A more flexible alternative directly incorporates the governing equations into the loss function, with automatic differentiation used to evaluate the required derivatives. Early attempts were limited by weak computational resources, the difficulty of building deep neural networks, and the inability to automatically compute partial derivatives of functions. The arrival of GPU computing, pioneered by NVIDIA, together with modern machine learning libraries, has made the training of deep networks a practical reality ([Huang et al., 2025](#)). This development laid the foundation for physics-informed neural networks (PINNs, [Raissi et al. \(2019\)](#)), which have since gained significant attention across diverse scientific and engineering applications. PINNs leverage the power of modern deep learning frameworks (e.g., TensorFlow, PyTorch, JAX), automatic differentiation tools, and GPU acceleration to solve differential equations without relying on traditional discretization techniques. One of the key advantages of PINNs is their flexibility: they can be used in forward problems, where the solution is learned given

known initial and boundary conditions, even without labeled data inside the computational domain. At the same time, they can be seamlessly applied to inverse problems by simply including a data mismatch term in the loss function to infer unknown equation parameters or functions. This dual capability allows a single PINN framework to solve both types of problems using the same underlying architecture, making it an efficient tool in scientific computing. PINNs are particularly well-suited for hydrological modeling, primarily because observational data are often sparse, noisy, and limited to the near-surface soil layers (Bandai and Ghezzehei, 2022). In such cases, purely data-driven models struggle to generalize, while PINNs can leverage the underlying physical laws to compensate for the lack of measurements. Moreover, PINNs have demonstrated robustness to noisy data (Yang *et al.*, 2021), making them effective in scenarios where measurement errors are inevitable. This is especially relevant in field applications, where sensor noise, environmental variability, and instrument limitations introduce uncertainty into the data. By enforcing the governing PDEs as soft constraints, PINNs can filter out noise and produce physically consistent solutions even when data quality is low.

1.3.3.2 Literature review

The application of PINNs in hydrological and transport modeling has gained significant traction in recent years. PINNs have been successfully applied to both forward and inverse problems in surface water flow modeling. For example, several studies have used the shallow water equations as physical constraints to train PINN models for simulating surface hydrodynamics under varying conditions (Nazari *et al.*, 2022; Ohara *et al.*, 2024; Donnelly *et al.*, 2024; Dazzi, 2024). In the vadose zone, PINNs have also demonstrated strong potential. Haruzi and Moreno (2023c) applied a PINN to inverse modeling of soil moisture and pore-water electrical conductivity in unsaturated soils. Their model was trained using time-lapse geoelectrical data and matric head measurements with unknown initial conditions. The predictions achieved by the PINN were in close agreement with HYDRUS simulations, showcasing its capability as an efficient inverse modeling solver. The work by Bandai and Ghezzehei (2021, 2022) explored forward and inverse modeling of 1D water infiltration in homogeneous and layered soils. They enhanced the standard PINN framework by incorporating adaptive activation functions to improve learning efficiency in highly nonlinear regimes governed by the Richards equation. Similarly, Wang *et al.* (2023b) developed a multiphysics PINN to solve coupled water flow and heat transfer in unsaturated soils. Their model integrated adaptive loss weighting and sequential training to improve convergence across complex coupled processes. To address soil heterogeneity and boundary uncertainties in the infiltration process, Gong and Zha (2025) employed PINNs combined with the Karhunen–Loève expansion, demonstrating reliable predictive capability for modeling stochastic unsaturated flow processes. Other applications of PINNs in hydrology include inverse estimation of soil parameters. For instance, Vemuri *et al.* (2024) used

water potential measurements to estimate the van Genuchten soil parameters, while Tartakovsky *et al.* (2020) demonstrated the use of PINNs for estimating spatially varying hydraulic conductivity under both saturated and unsaturated conditions. In a similar vein, Chen *et al.* (2023b) introduced an adaptive loss weighting strategy to improve the accuracy of PINNs in solving the Richards equation and validated its performance across several benchmark problems. A hybrid approach was proposed by Depina *et al.* (2022), who developed a two-loop framework where the PINN served as the forward model, and an external optimization algorithm was used to estimate soil parameters. Their method showed promising results in laboratory experiments. In Li *et al.* (2024a), the authors propose a long short-term memory PINN to predict the stress-strain responses and hydraulic properties of unsaturated soils by incorporating the soil-water retention curve into training.

On the other hand, Almajid and Abu-Al-Saud (2022) investigated the use of PINNs for modeling fluid flow in porous media using the Buckley–Leverett equation as a constraint, analyzing how the inclusion of observational data during training influenced prediction accuracy and generalization. Faroughi *et al.* (2023) applied a PINN framework with periodic activation functions to model solute transport in porous media, demonstrating the effectiveness of spectral features in capturing complex transport behavior. In a related study, He *et al.* (2020) assessed the performance of a PINN model for estimating hydraulic conductivity, hydraulic head, and solute concentration fields from sparse observational data. Their results demonstrated that PINNs can effectively solve inverse problems, achieving accurate reconstructions even under data-scarce conditions. Modeling flow in heterogeneous porous media remains a significant challenge for PINN models. To address this, Lehmann *et al.* (2023) proposed a PINN framework in which the residual is formulated using the mixed formulation of the governing equations. This approach enables more effective handling of heterogeneity in simulating fluid flow through complex porous structures. Similarly, Zhang *et al.* (2023) developed a data-free physics-informed convolutional neural network for two-phase flow in heterogeneous reservoirs. Their method integrates a finite volume discretization of the PDE residual into the loss function, leading to accurate predictions in both homogeneous and heterogeneous settings. Further advancing this direction, Yan *et al.* (2024) proposed a data-free PINN for predicting two-phase flow in 2D and 3D heterogeneous and fractured porous media. Their model successfully simulates immiscible displacement processes in fractured reservoirs and demonstrates robustness across complex scenarios, including variable injection–production schemes and gravity-driven segregation. These studies collectively underscore the growing capability of physics-informed machine learning models to address real-world porous media problems with high degrees of heterogeneity and limited data availability.

Data discovery is another research direction related to PINNs, in which deep learning algorithms are trained to identify the terms of the underlying differential equations and to approximate their coefficients using only scarce and noisy data. Such approaches

have been used to recover missing terms in PDEs or even reconstruct full governing equations (Schaeffer, 2017; Chen *et al.*, 2021b). Applications of this methodology to hydrological and transport models remain limited. For instance, Zhan *et al.* (2024) proposed a physics-guided neural network for discovering governing equations, embedding conservation laws and dimensional analysis to ensure physical consistency, and validated the method on Darcy’s law and the advection–diffusion equation under noisy and incomplete data. Sahimi (2024) explored data-driven discovery of transport equations in heterogeneous porous media. Xu *et al.* (2023b) applied sparse regression and neural operators to identify reactive transport dynamics, while Suh *et al.* (2024) demonstrated a symbolic regression approach for subsurface hydrology to derive an interpretable water retention model from limited and noisy pore-scale data. Song *et al.* (2023) developed a physics-regularized deep learning framework to reconstruct the unsaturated flow equation from scarce and noisy soil moisture data.

Theoretical investigations of PINNs have emerged as an active area of research, aimed at deepening our understanding of their approximation capabilities, convergence properties, and training dynamics. For instance, Mishra and Molinaro (2023) derive bounds on the generalization error of PINNs in terms of the training error and the number of collocation points, providing insights into how sample complexity affects performance. Shin *et al.* (2020) establish rigorous convergence results for PINNs applied to linear second-order elliptic and parabolic PDEs, offering theoretical guarantees under certain regularity assumptions. In the context of inverse problems, Mishra and Molinaro (2020) analyze the stability and generalization properties of PINNs, showing how error bounds depend on data noise and model complexity. Furthermore, Wang *et al.* (2022d) study the training dynamics of PINNs through the lens of the Neural Tangent Kernel (NTK) in the infinite-width limit. They show that the NTK of PINNs converges to a deterministic kernel that remains fixed during training, providing a theoretical explanation for the observed training behavior in practice. These studies highlight the growing body of theoretical work aimed at closing the gap between empirical success and mathematical understanding.

1.3.3.3 Limitations

In general, training PINNs to solve a differential equation is not as straightforward as might be expected, since convergence is not guaranteed and several inherent challenges arise. PINNs often struggle with differential equations that involve multi-scale dynamics, steep gradients, multiphysics interactions, chaotic behavior, or higher-order derivatives. These issues have been widely documented in the literature (see e.g., Wang *et al.* (2023a); Krishnapriyan *et al.* (2021)). As a result, specialized training strategies are often required, depending on the structure and complexity of the underlying PDE, to ensure accurate solutions and stable training. Another significant limitation of PINNs is their computational cost. Training can be time-consuming, ranging from several minutes to multiple hours or even days, depending on the scale and complexity

of the problem. In contrast, traditional numerical methods can often solve the same forward problem in a matter of seconds (Chuang and Barba, 2022). Consequently, while PINNs may not yet be competitive with classical methods for forward simulations, they are particularly powerful for inverse problems, where traditional approaches often face severe limitations in parameter estimation and data assimilation.

The vanilla PINN (Raissi *et al.*, 2019) faces several practical limitations that can significantly hinder its performance across a wide range of applications. One well-known issue is the spectral bias of neural networks, which refers to their inherent tendency to learn low-frequency components of a function more easily than high-frequency ones (Rahaman *et al.*, 2019). This makes it particularly difficult for PINNs to accurately capture sharp gradients or high-frequency oscillations, which are common in solutions of PDEs. A simple yet effective remedy is to transform the input space using Fourier feature mappings, i.e., by applying sinusoidal transformations to the inputs, so that both low- and high-frequency components of the solution are represented on equal footing from the beginning of training. Another major challenge is the multi-objective nature of the PINN loss function. The network is simultaneously trained to satisfy initial and boundary conditions as well as minimize the PDE residual, which can lead to gradient imbalance. When one component of the loss dominates the training, the model may become biased toward that component and neglect others, resulting in suboptimal or physically inconsistent solutions. To address this issue, several adaptive weighting strategies have been proposed that dynamically adjust the relative contributions of different loss terms during training. For example, Wang *et al.* (2021a) proposed a method based on gradient normalization that ensures all loss components contribute equally to the optimization process. Beyond these issues, several other technical challenges have been identified in the PINN literature. These include weight initialization pathologies that impair convergence (Wang *et al.*, 2024a), long-time integration errors that accumulate over extended time horizons (Wang *et al.*, 2023a; Krishnapriyan *et al.*, 2021), difficulties in solving higher-order PDEs due to the complexity of computing high-order derivatives accurately and stably (Wight and Zhao, 2020; Matthey and Ghosh, 2022), and challenges associated with multiphysics coupling, where different physical processes (e.g., water flow and solute transport) interact across varying temporal and spatial scales (Kamil *et al.*, 2024d).

Collectively, these limitations highlight the need for careful architectural choices, adaptive training strategies, and domain-specific enhancements to make PINNs truly effective for complex, real-world applications in scientific computing. However, it is essential to keep in mind that there is no universal PINN model that works for all types of PDEs (Monaco and Apletti, 2023), a reality aligned with the “no free lunch” theorem in optimization and learning theory. Consequently, developing a robust and efficient PINN model often requires extensive experimentation, sensitivity analysis, and rigorous benchmarking to tailor the approach to the particular characteristics of the PDE under study.

1.3.3.4 Neural operators

Parametric differential equations, often referred to as multi-task learning in the context of machine learning, arise when one aims to address multiple scenarios, such as varying soil properties, boundary conditions, or source terms. In such cases, the governing equations must be solved for each parameter configuration. Traditional numerical solvers require the model to be rerun for every new setting, which becomes computationally expensive and inefficient. This challenge is particularly significant in applications involving optimization, real-time control, or uncertainty quantification, where a large number of forward model evaluations are necessary. As a result, traditional methods are not well-suited for multi-task or many-query scenarios. To address this limitation, operator learning has emerged as a promising approach. The core idea is rooted in the theory that neural networks can approximate not just functions, but operators, that is, mappings between infinite-dimensional function spaces. Under suitable assumptions, neural networks can be trained to approximate the operator that maps input functions, such as PDE parameters or boundary conditions, to output functions, such as the corresponding PDE solution (Chen and Chen, 1993, 1995). Once trained, the operator can be evaluated quickly, making it attractive for real-time or large-scale applications. This idea has led to the development of several notable architectures. Deep Operator Networks (DeepONet, Lu *et al.* (2021a)) learn the nonlinear operator using a branch-trunk neural network structure. Fourier Neural Operators (FNO, Li *et al.* (2020)) leverage spectral convolution in Fourier space to capture global dependencies efficiently. Other extensions include the Laplace Neural Operator (LNO, Cao *et al.* (2024)), Convolutional Neural Operators (CNO, Raonic *et al.* (2023)), and Spectral Neural Operators (SNO, Fanaskov and Oseledets (2023)), each offering different perspectives on how best to encode the operator structure. To further improve performance in settings with limited data when physical constraints are available, recent work has extended operator learning frameworks by incorporating physics-based knowledge, similar to the PINNs approach. This includes Physics-Informed Neural Operators (PINO, Wang *et al.* (2021b)), Variationally Mimetic Operator Networks (VarMiON, Patel *et al.* (2024)), Mesh-Informed Neural Operators (MINO, Franco *et al.* (2023)), Variational Physics-Informed Neural Operator (VINO, Eshaghi *et al.* (2025)), and many others. These physics-aware models aim to combine the data efficiency and flexibility of operator learning with the physical reliability offered by governing equations, making them well-suited for forward and inverse problems in complex systems.

1.3.3.5 Literature review

In recent years, operator learning has emerged as a promising direction for addressing multi-task learning problems in hydrology, where traditional solvers become computationally expensive. For instance, Pang (2024) developed an FNO for efficient modeling of river hydrodynamics. Taccari *et al.* (2024) proposed a DeepONet architecture to

predict hydraulic head distributions in confined, heterogeneous aquifers. [Nunez *et al.* \(2025\)](#) applied an FNO to simulate pressure and CO_2 phase saturation distributions in carbon storage reservoirs. Similarly, [Choubineh *et al.* \(2023\)](#) used an FNO to predict single-phase fluid flow in porous reservoirs, while [Huang *et al.* \(2024\)](#) proposed a convolutional DeepONet for modeling reactive transport processes in porous media. In another study, [Haghighat *et al.* \(2025\)](#) developed a data-driven, transform-based neural operator to efficiently model contaminant transport in micro-cracked porous media. On the other hand, [Diab *et al.* \(2024\)](#) introduced a data-free DeepONet framework for modeling multiphase transport in porous media, where the Buckley–Leverett equation serves as the governing physical constraint. Their work highlights the promise of operator learning for physics-based modeling in the absence of observational data. More recently, [Ye *et al.* \(2025\)](#) introduced a hybrid data-driven and physics-informed DeepONet framework that surpasses HYDRUS in modeling soil moisture dynamics within one-dimensional soil columns.

Theoretical studies on neural operators have also emerged as an active and important research direction, providing insights into their approximation capabilities, generalization behavior, and convergence properties. For instance, [Lanthaler *et al.* \(2022\)](#) establish estimates on the approximation and generalization errors of DeepONets. They further extend the universal approximation theorem to measurable mappings in non-compact spaces, thereby broadening the theoretical foundation of operator learning. Similarly, [Kovachki *et al.* \(2021\)](#) demonstrate that FNOs are universal approximators for continuous operators between infinite-dimensional Banach spaces, providing strong guarantees for their applicability to a wide range of PDE-based tasks. In another line of work, [Bhattacharya *et al.* \(2020\)](#) analyze a class of neural operator architectures inspired by model reduction techniques and prove their convergence under suitable assumptions. These results collectively advance our theoretical understanding of neural operators and lay the groundwork for their application in high-dimensional and parameterized PDE problems.

1.4 Methodology

In the following sections, we present the methodology adopted in this thesis. We first describe the finite element method, which serves as a high-fidelity numerical simulator for generating reference solutions used in the training and evaluation of PINN and neural operator models. We then introduce physics-informed neural networks and neural operators for modeling water infiltration and single- or multi-species solute transport in porous media.

1.4.1 Finite element method

The finite element method (FEM) is a powerful and widely used numerical technique for solving differential equations, particularly in problems characterized by complex

geometries, heterogeneous material properties, and non-trivial boundary conditions. The FEM is based on reformulating the strong form of a PDE into its variational form, typically defined over a suitable Sobolev space. This formulation ensures the existence of a well-posed problem even for solutions with limited regularity. FEM then restricts the infinite-dimensional solution space to a finite-dimensional subspace, spanned by locally supported basis functions, often polynomials defined on a mesh partitioning the domain. The solution is approximated as a linear combination of these basis functions, and the resulting discrete problem is solved by enforcing the weak form to hold for all test functions in the chosen subspace.

In this thesis, FEM is employed to generate high-fidelity solutions for water flow and solute transport. These solutions serve both as benchmarks for validating the PINN and neural operator models, and as data sources for their training and evaluation. Further details of the developed FEM numerical model are provided in Chapter 2.

1.4.2 Physics-informed neural networks

In this section, we introduce the PINN framework for solving differential equations. Consider the following initial-boundary value problem defined over a domain Ω :

$$\mathcal{R}(u) = u_t + \mathcal{N}(u) = 0, \quad (1.4)$$

subject to suitable initial and boundary conditions:

$$\mathcal{B}(u) = 0, \quad \text{on } \partial\Omega, \quad (1.5)$$

where $\Omega \subset \mathbb{R}^n$ denotes an open, bounded domain with a sufficiently smooth boundary $\partial\Omega$. The operator $\mathcal{R}$ represents the governing physical law, encoding the relationship between the unknown solution u , its derivatives, and any source or sink terms within Ω . Note that the domain Ω also includes the temporal dimension. The operator $\mathcal{B}$ encompasses the initial and boundary conditions.

In forward modeling, the PDE (6.4), along with its initial and boundary conditions (7.14), is assumed to be fully specified. In contrast, inverse problems involve one or more unknown coefficients or source terms in the governing equations or conditions, which must be inferred from observational data through an inversion process.

The core idea of the PINN approach is to approximate the solution of a given PDE by restricting the solution space to functions represented by neural networks. The solution u is approximated by a neural network $u(\mathbf{x}; \Theta)$, where Θ denotes the neural parameters and $\mathbf{x} \in \Omega$ represents the spatial-temporal input variables. The parameters Θ are optimized by solving a non-convex minimization problem that enforces the governing PDE (6.4) and the associated initial and boundary conditions (7.14). This is achieved through a composite physics-informed loss function that includes contributions from the PDE residual, boundary constraints, and initial conditions. Automatic differentiation

is used to compute the required spatial and temporal derivatives of the network output with respect to its input, enabling efficient and accurate evaluation of the residuals during training.

The total loss function used to train the network is formulated as:

$$\mathcal{L}(\Theta) = \alpha_r \mathcal{L}_r(\Theta) + \alpha_{bc} \mathcal{L}_{bc}(\Theta), \quad (1.6)$$

where $\mathcal{L}_r$ and $\mathcal{L}_{bc}$ are the residual and boundary losses, respectively. The weights α_r and α_{bc} control the relative contribution of each term to the total loss. These components are typically approximated via Monte Carlo sampling over collocation points:

$$\mathcal{L}_r(\Theta) = \|\mathcal{R}(u(\mathbf{x}; \Theta))\|_{2,\Omega} = \int_{\Omega} \mathcal{R}(u(\mathbf{x}; \Theta))^2 d\Omega \simeq \frac{1}{N_r} \sum_{i=1}^{N_r} |\mathcal{N}(u(\mathbf{x}_i^r; \Theta))|^2, \quad (1.7a)$$

$$\mathcal{L}_{bc}(\Theta) = \|\mathcal{B}(u(\mathbf{x}; \Theta))\|_{2,\partial\Omega} = \int_{\partial\Omega} \mathcal{B}(u(\mathbf{x}; \Theta))^2 d\Omega \simeq \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} |\mathcal{B}(u(\mathbf{x}_i^b; \Theta))|^2, \quad (1.7b)$$

where $\{\mathbf{x}_i^r\}_{i=1}^{N_r} \subset \Omega$ and $\{\mathbf{x}_i^b\}_{i=1}^{N_{bc}} \subset \partial\Omega$ denote the sets of collocation points used to evaluate the PDE residual and the initial/boundary conditions, respectively. The residual and boundary terms are typically computed using the L^2 -norm, which is the most commonly adopted choice. However, other norms or weighting strategies may also be considered, depending on the problem characteristics and the desired training behavior. A general schematic overview illustrating the main workflow of a PINN solver is presented in Figure 1.1.

PINNs are mesh-free, semi-supervised solvers capable of addressing both forward and inverse problems. Unlike traditional numerical methods, PINNs do not require explicit discretization of the spatial and temporal domains, nor do they rely on iterative schemes to handle implicit nonlinear terms. This makes them particularly well-suited for solving high-dimensional and parametric PDEs. PINNs are especially effective in inverse problems, where unknown parameters or inputs must be inferred from sparse or noisy observations. In such cases, a data-mismatch term can be naturally incorporated into the overall loss function (7.15), enabling joint training of the solution and parameters without the need for a separate optimization loop. This seamless integration of data and physical laws makes PINNs a powerful and flexible tool for model calibration, uncertainty quantification, and inverse problem solving in scientific computing.

Solving a coupled system of equations using PINNs is straightforward. The total loss function for a PINN solving N equations is defined as:

$$\mathcal{L}(\Theta) = \sum_{i=1}^N \lambda_i \mathcal{L}_i(\theta_i), \quad (1.8)$$

where $\mathcal{L}_i$ denotes the loss associated with the i -th equation, and λ_i is its correspond-

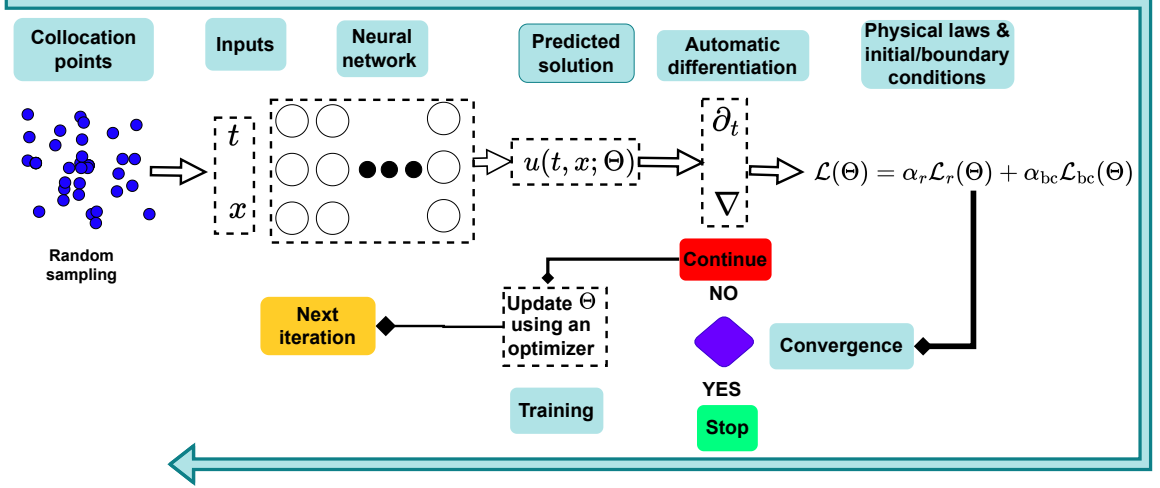


Figure 1.1 General workflow of a PINN solver. Collocation points for the PDE residual, boundary and initial conditions, and optional data are supplied to a neural network that represents the solution. Automatic differentiation provides spatial and temporal derivatives to form residual and constraint terms. A composite physics-informed loss is minimized with standard optimizers until convergence, yielding a trained surrogate for forward prediction and, when applicable, estimated parameters for inverse problems.

ing weighting coefficient. The parameter vector $\Theta = (\theta_1, \dots, \theta_N)$ includes the neural network parameters used to approximate the solution of each equation in the system.

1.4.3 Neural operators

In this section, we introduce the mathematical framework and architectural principles underlying neural operators, with a particular emphasis on Deep Operator Networks (DeepONets, [Lu et al. \(2021a\)](#)) and its physics-informed version (PI-DeepONets, [Wang et al. \(2021b\)](#)).

The goal of neural operators is to learn mappings between infinite-dimensional function spaces:

$$G: \mathcal{P} \rightarrow \mathcal{S}, \quad p \mapsto G(p) = s, \quad (1.9)$$

where $\mathcal{P}$ and $\mathcal{S}$ are Banach spaces. For instance, $\mathcal{P}$ may represent a space of input functions, such as source terms or initial and boundary conditions of a given PDE, while $\mathcal{S}$ represents the corresponding solution space. The objective is to construct a surrogate model G_Θ that approximates the true operator G . Once trained, G_Θ can predict the solution $s \in \mathcal{S}$ for any new input $p \in \mathcal{P}$ within the training distribution, eliminating the need to explicitly solve the underlying PDE. This enables fast and scalable inference across a wide range of input conditions.

The feasibility of learning such an operator using neural networks is theoretically sup-

ported by the universal approximation theorem for operators (Chen and Chen, 1995), which states that neural networks can approximate nonlinear operators under certain conditions. Building on this foundation, various neural operator architectures have been proposed to make this idea practical and computationally efficient. In what follows, we focus on the structure and learning mechanism of one such approach: DeepONets.

1.4.3.1 Deep Operator Networks: DeepONets

DeepONets (Lu *et al.*, 2021a) are a neural network architecture specifically designed for learning nonlinear operators, grounded in the universal approximation theorem for operators (Chen and Chen, 1995). A DeepONet consists of two separate neural networks:

1. **Branch network:** Takes as input the function $p \in \mathcal{P}$, typically represented by its evaluations at a finite set of sensor points $\{s_1, s_2, \dots, s_m\}$. Given an input function p , the branch network receives the vector $[p(x_1), \dots, p(x_m)]$ and outputs a feature vector $[b_1(p), \dots, b_q(p)]$.
2. **Trunk network:** Takes as input a spatial or spatio-temporal coordinate $\mathbf{x} \in \Omega \subset \mathbb{R}^d$, where the output of the operator is to be evaluated. The trunk network processes the spatial-temporal coordinates $\mathbf{x} = (t, x)$ and outputs basis functions $[\Psi_1(\mathbf{x}), \dots, \Psi_q(\mathbf{x})]$.

The final output is computed by taking the inner product of the outputs of the branch and trunk networks, allowing the model to approximate the action of the operator at any location $\mathbf{x}$:

$$G_{\Theta}(p)(\mathbf{x}) = \sum_{i=1}^q b_i(p) \Psi_i(\mathbf{x}) + c_0, \quad (1.10)$$

where Θ denotes the collective trainable parameters of both networks, and c_0 is an optional bias term.

A key advantage of DeepONets lies in their ability to generalize across function spaces: once trained on a sufficiently diverse set of input-output function pairs $(p, G_{\Theta}(p))_{p \in \mathcal{P}}$, the model can predict the solution $G_{\Theta}(p)$ for an unseen input function $p \in \mathcal{P}$ with minimal additional computation. This makes DeepONets particularly well-suited for parametric PDEs and scenarios involving multi-query evaluations, such as uncertainty quantification, optimization, and real-time control.

To ensure good generalization performance, it is critical to sample a representative set of input functions from $\mathcal{P}$ and solution evaluations over a dense grid in the output domain Ω . This allows the network to capture the functional diversity and variability inherent in the operator G .

In the next subsection, we will explore how to incorporate physical constraints, such as governing PDEs, into the DeepONet framework to enable training in data-scarce regimes, leading to the development of physics-informed DeepONets.

1.4.3.2 Physics-informed DeepONets: PI-DeepONets

DeepONet, being a purely data-driven approach, becomes inefficient in scenarios where high-fidelity real-world data are scarce, as its performance heavily relies on the availability of large and diverse training datasets. This limitation has motivated researchers (e.g., Wang *et al.* (2021b)) to integrate physical laws directly into the learning process as used in the PINN framework (Raissi *et al.*, 2019). This leads to the so-called physics-informed DeepONet (PI-DeepONet). This approach offers a promising solver to enhance model robustness and generalizability in scenarios where data availability is constrained. The loss function $\mathcal{L}$ for training the PI-DeepONet can be defined as follows:

$$\mathcal{L}(\Theta) = \lambda_{\text{data}} \mathcal{L}_{\text{data}}(\Theta) + \lambda_{\text{physics}} \mathcal{L}_{\text{physics}}(\Theta), \quad (1.11)$$

where $\mathcal{L}_{\text{physics}} = \mathcal{L}_{\text{pde}} + \mathcal{L}_{\text{ic}} + \mathcal{L}_{\text{bc}}$, λ_{data} and λ_{physics} are weighting factors associated with the data and PDE losses, respectively.

Refer to Figure 1.2 for a schematic illustration of the standard DeepONet architecture and its physics-informed extension.

1.5 Thesis outcome

1.5.1 Contributions

The primary objective of this thesis project is to develop efficient and accurate predictive algorithms for forward, inverse, and multi-task modeling of water infiltration and single- or multi-species solute transport in porous media. The proposed methods are designed to address the complex multiphysics coupling arising from water flow and multispecies fertilizer transport, as well as the stiff, highly nonlinear behavior inherent in the governing infiltration PDEs.

For forward modeling, we develop solvers capable of delivering physically consistent and accurate predictions across infiltration regimes, ranging from stable flow governed by the Richards equation to unstable infiltration described by the higher-order BCJ-R model. The solvers also address the coupling of infiltration processes with multi-species solute transport. For inverse problems, the algorithms are designed to operate effectively in data-scarce and noisy environments. They enable the estimation of unknown parameters in the PDE models, such as soil hydraulic properties or reaction coefficients, using limited field observations. In multi-query settings, such as the evaluation and optimization of different irrigation or fertigation strategies, the goal is to construct efficient surrogate models that can approximate water infiltration dynamics. These surrogate models allow for rapid inference of soil moisture distributions and can also

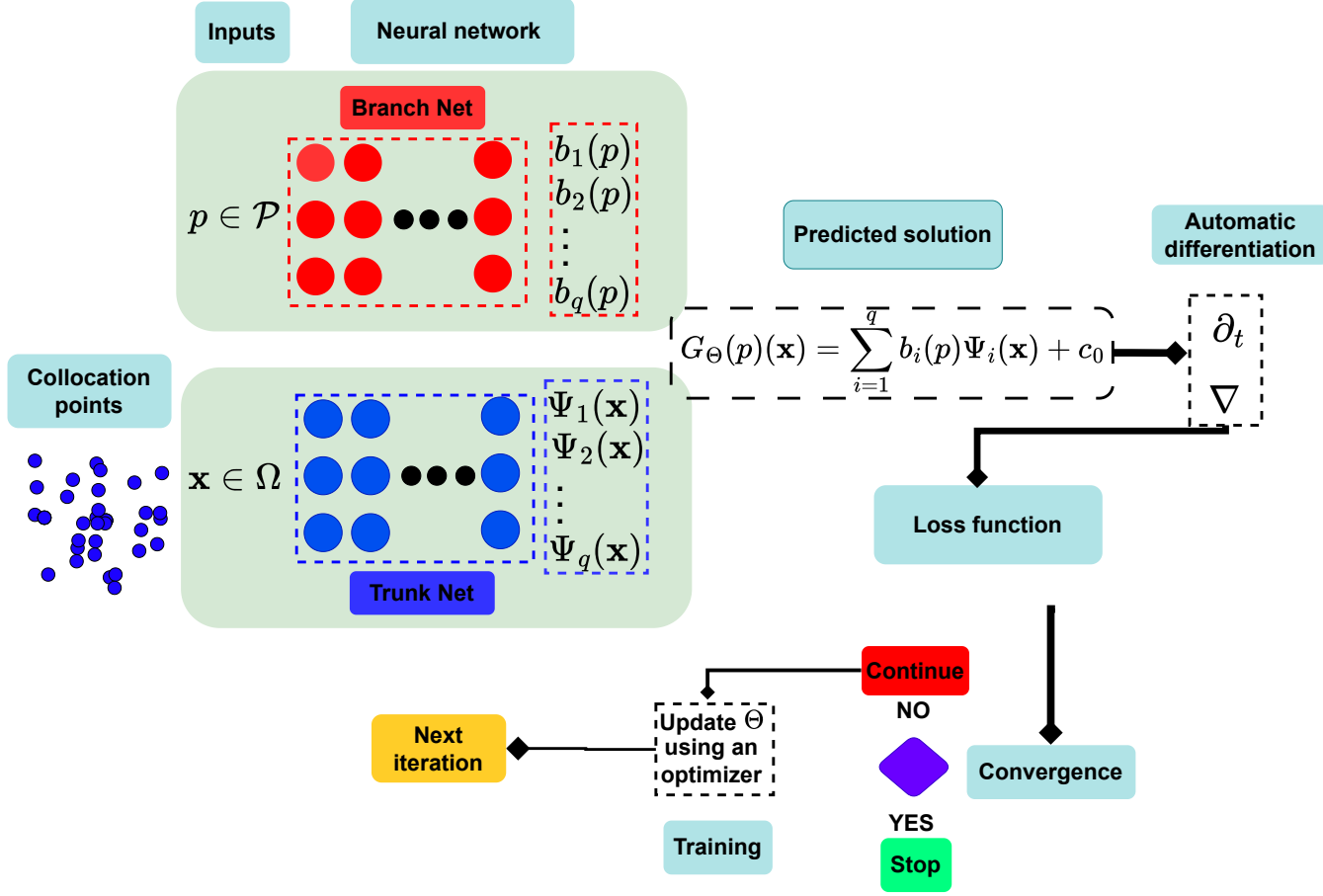


Figure 1.2 A schematic representation of the DeepONet framework. The architecture consists of a branch network that processes the input function from the parameter space $\mathcal{P}$, and a trunk network that handles the coordinates of the computational domain Ω . The outputs of the two networks are combined to approximate the target operator G . DeepONet can be trained using a data-driven loss, a physics-informed loss, or a combination of both, optimized using for example gradient-based optimizers.

be leveraged to solve inverse problems without repeatedly running computationally expensive solvers.

In the first part of this thesis, we developed an efficient mixed FEM for the coupled modeling of water flow and solute transport in unsaturated soils. The proposed numerical model is semi-implicit, which avoids the need for iterative solvers and thus mitigates convergence issues commonly encountered in solving the Richards equation. The semi-implicit mixed FEM approach is extended to handle both single and multi-species solute transport in two-dimensional flow domains. Its performance is validated against reference solutions, including analytical benchmarks, other numerical methods, and experimental datasets. These comparisons confirm the reliability and accuracy of

the proposed FEM scheme, which will also serve as a reference solution and data generator for training PINNs in forward and inverse modeling, as well as for multi-task learning scenarios for training a physics-informed DeepONet model.

In the second part, we shifted focus to the development of a PINN model for simulating the coupled flow and transport processes in unsaturated porous media. This PINN model is data-free and trained in a semi-supervised fashion, relying only on initial and boundary condition data. Its accuracy is assessed by comparing predictions against the FEM reference model. Numerical experiments demonstrate that the PINN yields reliable results and can approximate the solution with good agreement across a range of scenarios.

The third part of the thesis is dedicated to identifying the most effective PINN strategies for handling water infiltration and multi-species solute transport in unsaturated soils. This includes a systematic comparative study addressing both forward and inverse problems, with a focus on inferring unknown PDE parameters from sparse and noisy measurements. Among the models evaluated, the sequential PINN, where each PDE is trained independently, emerged as the most accurate and computationally efficient approach. The reliability of its predictions was further validated using experimental data.

In the fourth part, we extended the sequential PINN framework by incorporating a transfer learning strategy to enhance training efficiency and generalization. The idea is to progressively train the model across a sequence of PDEs, leveraging prior knowledge gained from earlier training phases. This approach significantly reduces training time and improves convergence. Extensive simulations demonstrate its effectiveness in both forward and inverse modeling tasks for water and multi-species solute transport problems.

In the fifth part of the thesis, the sequential training strategy is generalized to a broader class of one-way PINN solvers. We propose specialized PINN architectures that exploit the inherent structure of one-way coupled systems. The developed models demonstrate good performance in simulating coupled water flow and solute transport involving up to three interacting species. These architectures effectively capture the causality and directional dependencies of the physical processes, ensuring both accuracy and physical consistency.

In the sixth part, we addressed the challenging problem of modeling unstable infiltration using the high-order BCJ-R equation. Our objective is to develop an efficient PINN model capable of capturing both the low- and high-frequency characteristics of the BCJ-R dynamics, including saturation overshoot at the wetting front. We found that standard PINN architectures previously applied to the Richards equation were insufficient to capture key features of the BCJ-R equation. Therefore, we adopted several improvements, including network architecture modifications and training strategies, to accurately resolve phenomena such as saturation overshoot. The resulting PINN model not only matches experimental observations but also satisfies thermodynamic princi-

ples, such as entropy dissipation in closed systems.

Finally, in the seventh part, we developed a neural operator-based model tailored for multi-task learning of water infiltration in porous media. This model is data-free, mesh-independent, and trained in a self-supervised manner. It can solve the Richards equation across a wide range of initial and boundary conditions and provide soil moisture predictions at arbitrary spatial and temporal locations. The model successfully learns the mapping from inputs (e.g., water fluxes, initial conditions) to soil moisture distributions. To enhance its extrapolation capability beyond the training distribution, we integrate physics-based regularization and fine-tune the model using small sets of data measurements. This positions the operator-based model as an efficient real-time surrogate for high-fidelity simulations in hydrological modeling, enabling rapid predictions across varying inputs without the need to repeatedly solve the underlying PDEs.

1.5.2 Thesis outline

The rest of this thesis is organized as follows. We begin by presenting a semi-implicit mixed finite element method for modeling coupled water flow and solute transport in unsaturated soils in Chapter 2. We then introduce the use of PINNs for forward modeling of these coupled processes and evaluate their accuracy against the finite element method in Chapter 3. Next, we carry out an extensive comparative study to identify the most effective PINN strategies for both forward and inverse problems involving water infiltration and multispecies solute transport in Chapter 4. We further enhance the sequential training of PINNs by incorporating transfer learning to improve training efficiency and model accuracy in Chapter 5. Building on this, we extend the sequential PINN framework to general one-way coupled systems and propose new architectures that leverage their directional structure in Chapter ???. We then develop a PINN model tailored for simulating unstable infiltration governed by the high-order BCJ-R equation in Chapter 7, capturing key features such as saturation overshoot and ensuring consistency with thermodynamic principles. Finally, we introduce a physics-informed neural operator in Chapter 8 for multitask learning of water infiltration in porous media, enabling rapid predictions across a range of input conditions. We conclude in Chapter 9 with a summary of the main contributions and outline directions for future research.

CHAPTER 2 Semi-implicit schemes for modeling water flow and solute transport in unsaturated soils

Abstract

The coupled model of water flow and solute transport in unsaturated soils is addressed in this study. Building upon previous research findings by Keita, Beljadid, and Bourgault, we investigate a class of second-order time-stepping techniques where two free parameters are introduced, to identify the most stable and accurate scheme. The spatial discretization of the Richards equation is accomplished using the mixed finite element method. The proposed approach involves formulating noniterative schemes using an extrapolation formula and Taylor approximation in time to linearize nonlinear terms. Additionally, a specialized regularization technique is applied to ensure the convergence of the proposed numerical methods. Numerical simulations are conducted to determine the optimal scheme for solving the Richards equation, which is subsequently extended to the transport equation.

Numerical simulations of water flow reveal the good accuracy of three schemes, SBDF, MSBDF, and Richards-M2 for homogeneous and heterogeneous soils. Notably, the SBDF scheme stands out for its computational efficiency and stability, especially when gravity forces dominate over capillary forces. Through numerical analysis of the coupled semi-implicit schemes, our results affirm the SBDF scheme's superior robustness, establishing it as the optimal choice among the proposed numerical methods. Therefore, the SBDF scheme is employed to solve the coupled model of water flow and solute transport. We conducted various numerical experiments to solve the coupled model, addressing scenarios including single and multispecies nitrogen transport, pore water electrical conductivity, and nitrate transport. The SBDF scheme's accuracy was rigorously verified through comparisons with reference solutions and experimental data. This establishes the SBDF scheme as an efficient alternative to traditional implicit methods for modeling water flow and solute transport in unsaturated soils.

2.1 Introduction

In multiple fields such as hydrology, soil physics, and agriculture, there has been a growing emphasis on the infiltration of fluids through porous mediums, recognizing its profound importance. The prediction of soil moisture and solute concentration in soils remains a complex and challenging task. The utilization of an experimental-based approach can be demanding in terms of resources and time. Mathematical modeling offers an efficient approach for rapid prediction and visualization of the studied physical parameters, making it an excellent choice in various applications. Various infiltration models have been developed in the literature to describe water movement through soil. The most used model to describe the infiltration process in unsaturated porous media

is the Richards equation (Richards, 1931) which is derived based on a multiphase flow extension of Darcy’s law and the principle of mass conservation. Many studies were conducted to extend the Richards equation to account for gravity-driven fingers in unsaturated porous media (Eliassi and Glass, 2002; Cueto-Felgueroso and Juanes, 2009; Xiong, 2014; DiCarlo, 2013). In (Beljadid *et al.*, 2020), Beljadid, Cueto-Felgueroso and Juanes recently developed a continuum model of unstable infiltration in porous media, endowed with an entropy function.

The Richards equation is often coupled with other advection-diffusion equations, such as those governing solute transport or heat transfer. In terms of models, the Richards equation and convection-dispersion equations can simulate physical phenomena in unsaturated porous media. However, analytical solutions are generally unavailable when confronted with more realistic boundary conditions, intricate geometries, or highly nonlinear water retention curves. Consequently, numerical schemes represent a powerful tool to approximate the solutions of the coupled model of water flow and solute transport for a given problem, irrespective of its geometry or boundary conditions. A variety of numerical methods are available in the literature to solve the Richards equation: they include the standard Galerkin finite element (Arbogast *et al.*, 1993; Slodicka, 2002), mixed-hybrid finite element (Bergamaschi and Putti, 1999a), local discontinuous Galerkin (Li *et al.*, 2007) methods, finite volume (Eymard *et al.*, 1999; Machado *et al.*, 2022), finite difference (Celia *et al.*, 1990a; Fengnan *et al.*, 2020), and meshless (Boujoudar *et al.*, 2021a,b, 2023a) techniques. We refer the reader to the literature review (Zha *et al.*, 2019) for more detailed information. For the transport equation, various numerical methods are available, such as the upstream weighting finite element (Kaluarachchi and Parker, 1988), the upwind mixed hybrid finite element (Younes *et al.*, 2022), and the finite difference (Chen *et al.*, 2019a; Selim and Iskandar, 1981) schemes. Several studies have addressed the challenge of solving the coupled equations governing water flow and solute transport by employing either a deterministic approach (Antonopoulos, 1993; Chen *et al.*, 2019a; Haruzi and Moreno, 2023c; Illiano *et al.*, 2021; Kaluarachchi and Parker, 1988; Kamil *et al.*, 2024c; Selim and Iskandar, 1981; Srivastava and Yeh, 1992; Kamil *et al.*, 2024e) or a stochastic one (Russo, 2012b; Russo *et al.*, 1998, 2001). For solving this coupled system, implicit schemes can be used due to their stability and reasonable time step. Various implicit schemes have been proposed (Diaw *et al.*, 2001; Radu *et al.*, 2008; Radu and Pop, 2010; Radu *et al.*, 2010, 2006). Diaw *et al.* (Diaw *et al.*, 2001) proposed a standard finite element method with primary variable switching to solve the Richards equation, along with operator splitting and the discontinuous finite element method for the transport of non-sorptive solute. They demonstrated the efficiency of their schemes in a one-dimensional heterogeneous soil. Mitchell *et al.* (Mitchell and Mayer, 1998) solved the coupled model of water flow and solute transport by combining the standard finite element method with an Eulerian-Lagrangian method for the transport equation. The pioneering works of Radu *et al.* (Radu *et al.*, 2008; Radu and Pop, 2010; Radu *et al.*, 2010, 2006) have pro-

vided theoretical analyses based on the implicit mixed finite element method, various linearization techniques (e.g., Newton’s method), and different numerical treatments of advection and diffusion terms. Most papers dealing with numerical schemes for solving the coupled flow-transport system rely on implicit treatment of the Richards equation to compute variables of interest, such as water content and water flux. However, implicit schemes require a linearization solver, which may lead to convergence issues for certain capillary models and types of soil (Cote *et al.*, 2003). The limitations of fully implicit schemes highlight the need for further research to explore and develop more powerful and robust numerical techniques capable of accurately and efficiently modeling the coupled water flow and solute transport processes in unsaturated soils. The aim of this study is to introduce and numerically investigate semi-implicit schemes for solving the coupled flow-transport system in soils. The goal is to study the advantages of these schemes and select the potential ones which can be the subject of more theoretical analysis in future studies.

This study is a follow-up to the papers (Beljadid and Hanini, 2023; Keita *et al.*, 2021a,b,d), where semi-implicit time-stepping schemes were developed for solving nonlinear partial differential equations. Time-stepping schemes for the coupled Richards equation and solute transport model have received less attention than spatial discretization schemes. The implicit Euler temporal scheme is commonly employed due to its ability to handle highly nonlinear functions and allow for reasonably sized time steps. However, it is important to note that this method may face convergence challenges in certain cases, particularly when applied to the Richards equation (Kavetski *et al.*, 2002; Lehmann and Ackerer, 1998). Therefore, noniterative time-stepping schemes have been proposed for the Richards equation (e.g., Kavetski *et al.* (2001, 2002); Keita *et al.* (2021b); Paniconi *et al.* (1991); Shahraiyini and Ataie-Ashtiani (2012)). These schemes prove advantageous by providing stable and accurate results without encountering convergence issues, especially when dealing with the degeneracy associated with certain constitutive relationships (Kavetski *et al.*, 2002). In (Paniconi *et al.*, 1991), it was demonstrated that, when solving the Richards equation, second-order accurate non-iterative schemes can achieve computational efficiency and accuracy comparable to iterative methods such as Newton’s and Picard’s methods.

Keita, Beljadid, and Bourgault (Keita *et al.*, 2021b) have developed time-stepping schemes for the Richards equation, building upon their previous studies (Keita *et al.*, 2021a,d). They employed the second-order backward differentiation formula (BDF2) for the time derivative approximation, known for its efficiency in computing solutions for stiff problems with a reasonable time step (Beljadid *et al.*, 2014; Boujoudar *et al.*, 2024; Dharmaraja, 2007; Keita *et al.*, 2021b). Subsequently, the implicit nonlinear functions were linearized using special approximation methods, eliminating the necessity for an iterative process. This method holds the potential to enhance the accuracy and efficiency of solving the Richards equation. In their study, they analyzed various time-stepping schemes, including both fully implicit and semi-implicit approaches.

Their findings indicated that the semi-implicit (S, Ψ) -scheme exhibited superior performance compared to other schemes. Notably, the explicit treatment of the unsaturated conductivity played a pivotal role in significantly reducing the required CPU time, leading to a remarkable improvement in computational efficiency. This result emphasizes the potential of the developed semi-implicit (S, Ψ) -scheme for effectively modeling unsaturated flow through soils. The foundation of their work relied on implicit-explicit (IMEX) schemes, also known as semi-implicit schemes, which are extensively detailed in (Ascher *et al.*, 1995). IMEX schemes have proved to be versatile and effective in various applications, and researchers have developed several IMEX schemes for different types of equations, including reaction-diffusion systems (Koto, 2008) and other complex problems (DeCaria and Schneier, 2021; Ethier and Bourgault, 2008; Loy and Bourgault, 2017). The fundamental concept behind IMEX schemes is to utilize an explicit time treatment for handling nonlinear terms while employing an implicit approach for linear ones. This combination of explicit and implicit methods allows for an efficient and accurate treatment of both terms. The solute transport equation can generally involve multiple phenomena, including advection, dispersion, and reaction. In this context, IMEX schemes can allow the implementation of customized time-stepping techniques for each specific phenomenon.

The primary objective of the present paper is to explore the extension of IMEX schemes to solve the coupled model of water flow and solute transport in unsaturated soils, building upon the scheme developed in the previous study (Keita *et al.*, 2021b). A mixed finite element method is employed for the spatial discretization of the Richards equation, while the Galerkin finite element for the solute transport. Moreover, a specialized regularization technique is integrated into the proposed schemes for the Richards equation to avoid singularities, ensuring stable and accurate solutions during the numerical computations (Keita *et al.*, 2021b). We aim to present a comprehensive and efficient numerical framework for simulating the complex processes of unsaturated infiltration and transport through soils.

The paper is organized as follows. In Section 2.2, we introduce the mathematical equations governing the coupled model of water flow and solute transport. Section 2.3 presents the weak formulations of the studied problem. The time-stepping schemes for the coupled model are outlined in Section 2.4. In Section 5.5, we conduct various numerical tests to assess the proposed methods. Finally, in Section 4.6, we draw conclusions and provide future perspectives.

2.2 Governing equations

2.2.1 Water flow

The Richards equation is a commonly employed mathematical model for simulating the infiltration of water in unsaturated soils. It is derived by combining the mass

conservation principle and Darcy-Buckingham's law (Buckingham, 1907). There are three primary versions of this equation (Celia *et al.*, 1990a). In this study, the mixed form is used to ensure mass conservation (Celia *et al.*, 1990a). The equation can be expressed as follows:

$$\begin{cases} \frac{\partial \theta}{\partial t} + \nabla \cdot \mathbf{q} = f(\Psi) & \text{in } \Omega \times I, \\ \mathbf{q} = -K \nabla(\Psi + z), \end{cases} \quad (2.1)$$

where Ψ [L] is the pressure head, $\mathbf{x} = (x, z)$ denotes the spatial coordinates, and the vertical coordinate z [L] is assumed to be directed upward, $\mathbf{q}$ is the water flux, $K = K_s k_r$ [LT⁻¹] is the unsaturated hydraulic conductivity, and K_s is the saturated hydraulic conductivity [LT⁻¹], k_r is the water relative permeability, θ [-] is the volumetric water content, $f(\Psi)$ [T⁻¹] is a source or sink term, $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$, is a connected bounded open domain representing the soil and having a smooth boundary $\partial\Omega$. $I = (0, T]$, where $T > 0$ is the final simulation time. In this paper, we set $f(\Psi) = 0$.

To solve the Richards equation, explicit functions for K and θ in terms of Ψ need to be provided. In the literature, commonly used models for these functions include the ones proposed by Gardner (Gardner, 1958), Brooks-Corey (Brooks and Corey, 1966), Haverkamp (Haverkamp *et al.*, 1977), and van Genuchten-Mualem (Mualem, 1976; Van Genuchten, 1980).

In this study, we used the mixed form of the Richards equation with the main variable being the effective saturation:

$$S = \frac{\theta - \theta_r}{\theta_s - \theta_r}, \quad (2.2)$$

where θ_s and θ_r denote the saturated and residual volumetric water contents, respectively, and are assumed to depend solely on space. The S -based form of the Richards equation is given by:

$$\phi \frac{\partial S}{\partial t} = \nabla \cdot \left[K \nabla(\Psi + z) \right] \quad \text{in } \Omega \times I, \quad (2.3)$$

where $\phi = \theta_s - \theta_r$.

The relationship between the pressure head and saturation is described by Leverett's equation (Leverett, 1941):

$$\Psi(S) = h_{\text{cap}} J(S), \quad (2.4)$$

where h_{cap} represents the capillary rise function, which is assumed to depend solely on spatial coordinates, and J denotes the Leverett function. Figure 2.1 illustrates the absolute value of the Leverett function for Gardner, Brooks-Corey, Haverkamp, and van Genuchten capillary pressure models. The Leverett function exhibits degeneracy when $S = 1$ for both the Haverkamp and van Genuchten models. To ensure convergence of numerical schemes for saturation in the vicinity of one, regularization techniques for

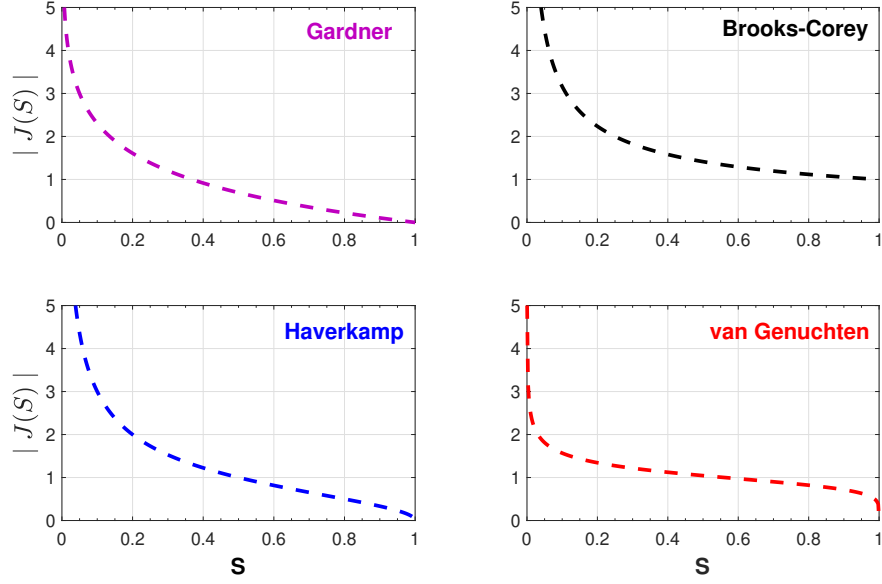


Figure 2.1 Examples of the absolute value of the Leverett function for various capillary pressure models.

the derivatives of this function are employed (Keita *et al.*, 2021b; Pop and Schweizer, 2011). These regularization methods aim to address the challenges associated with the degenerate behavior, leading to improved accuracy and stability of numerical computations, as demonstrated in (Keita *et al.*, 2021b). It is noteworthy that capillary pressure models, such as the Brooks-Corey and Gardner models, do not require the use of regularization techniques. In our study, we employed the Gardner, Haverkamp, and van Genuchten models.

2.2.2 Solute transport

The solute transport equation in unsaturated soil is intricately coupled with the Richards equation, which takes the following form (Bear, 2013; Šimůnek *et al.*, 2005):

$$\frac{\partial \theta c}{\partial t} + \frac{\partial \rho s}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c - c \mathbf{q}), \quad (2.5)$$

where c is the concentration of the solute in the liquid phase [ML^{-3}], s is the concentration of the solute in the solid phase [MM^{-1}], ρ is the soil bulk density [ML^{-3}], $\mathbf{D}$ is the dispersion tensor [L^2T^{-1}], which incorporates the effect of molecular diffusion and the hydrodynamic dispersion, θ and $\mathbf{q}$ are the water content and the water flux, respectively, given by the Richards equation (2.1).

The interaction between the solute concentration c and the adsorbed concentration s

can be expressed using the Freundlich equation ([Šimůnek et al., 2005](#)):

$$s = K_d c^p, \quad (2.6)$$

where K_d is an empirical coefficient [L^3M^{-1}], which is equal to zero if no interaction between soil particles and solute is considered, p is the Freundlich constant, which is equal to 1 for linear models (2.6).

The solute transport equation (5.3) can be transformed to the following form:

$$\frac{\partial R\theta c}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c - c \mathbf{q}), \quad (2.7)$$

where R is the retardation factor [-], which accounts for the interaction between the solute and the solid phase of the soil. It is determined by the following formula:

$$R = 1 + \rho K_d \frac{c^{p-1}}{\theta}. \quad (2.8)$$

The components of the dispersion tensor when neglecting the gaseous phase are calculated as follows ([Bear, 2013](#)):

$$\theta \mathbf{D}_{ij} = (D_T \|\mathbf{q}\| + \theta D_w \tau) \delta_{ij} + (D_L - D_T) \frac{\mathbf{q}_i \mathbf{q}_j}{\|\mathbf{q}\|}, \quad (2.9)$$

where D_w is the molecular diffusion constant [L^2T^{-1}], D_L and D_T are the longitudinal and transversal dispersivities [L]. δ_{ij} is the Kronecker symbol, τ is the tortuosity factor in the liquid phase and $\|\mathbf{q}\|$ denotes the Euclidean norm of $\mathbf{q}$ given by (2.1). The tortuosity factor is calculated by using the following formula ([Bear, 2013](#); [Millington and Quirk, 1961](#)):

$$\tau = \frac{\theta^{7/3}}{\theta_s^2}. \quad (2.10)$$

The dispersion tensor D in one-dimensional problem is a single value that combines the effects of molecular diffusion and hydrodynamic dispersion ([Bear, 2013](#)):

$$\theta \mathbf{D} = D_L |\mathbf{q}| + \theta \tau D_w. \quad (2.11)$$

For two-dimensional problems, the coefficients of tensor $\mathbf{D}$ are ([Bear, 2013](#)):

$$\begin{aligned} \theta \mathbf{D}_{xx} &= D_L \frac{\mathbf{q}_x^2}{\|\mathbf{q}\|} + D_T \frac{\mathbf{q}_z^2}{\|\mathbf{q}\|} + \theta D_w \tau, \\ \theta \mathbf{D}_{zz} &= D_L \frac{\mathbf{q}_z^2}{\|\mathbf{q}\|} + D_T \frac{\mathbf{q}_x^2}{\|\mathbf{q}\|} + \theta D_w \tau, \\ \theta \mathbf{D}_{xz} &= \theta \mathbf{D}_{zx} = (D_L - D_T) \frac{\mathbf{q}_x \mathbf{q}_z}{\|\mathbf{q}\|}, \end{aligned} \quad (2.12)$$

where $\mathbf{q}_x$ and $\mathbf{q}_z$ are the components of the water flux in x and z directions respectively, obtained from (2.1).

The equation (5.5) is classified as a linear advection-diffusion equation when $p = 1$. We note that the water content θ may depend on the concentration c of the solute as in the surfactant transport problems (Illiano *et al.*, 2021). However, this aspect falls outside the scope of the present paper.

2.2.3 Initial and boundary conditions

To close equations (2.3) and (5.5), appropriate initial and boundary conditions must be specified. In this study, we employed the following initial and boundary conditions for the Richards equation:

$$\begin{cases} S(x, 0) = S_0(x) & \text{in } \Omega, \\ S = S_D & \text{in } \Gamma_D \times I \subset \partial\Omega \times I, \\ \mathbf{q} \cdot \mathbf{n} = q_N & \text{in } \Gamma_N \times I \subset \partial\Omega \times I, \end{cases} \quad (2.13)$$

where S_0 , S_D , and q_N represent the initial saturation, Dirichlet, and Neumann boundary conditions, respectively. $\partial\Omega$ denotes the boundary of the domain Ω , while Γ_D and Γ_N refer to the Dirichlet and Neumann boundaries, respectively. The outward unit normal vector to the boundary Γ_N is represented by $\mathbf{n}$. It is worth noting that the Neumann boundary condition may be imposed on the pressure head Ψ in situations such as free drainage boundary conditions (refer to sections 2.5.2.4 and 2.5.2.5). Furthermore, we emphasize that the initial condition can be specified as a function of the water content θ , the saturation S , or the pressure head Ψ .

For the transport equation (5.5), the following initial and boundary conditions can be used:

$$\begin{cases} c(x, 0) = c_0(x) & \text{in } \Omega, \\ c = c_D & \text{in } \Gamma_D \times I \subset \partial\Omega \times I, \\ -(\theta \mathbf{D} \nabla c - \gamma c \mathbf{q}) \cdot \mathbf{n} = \bar{c}_0 q_N & \text{in } \Gamma_N \times I \subset \partial\Omega \times I, \end{cases} \quad (2.14)$$

where c_0 and c_D denote the initial and Dirichlet boundary conditions. The third equation in (5.11) represents a second-type (Neumann) boundary condition when $\gamma = 0$ and a third-type (Cauchy) boundary condition when $\gamma = 1$, respectively. The notation $\bar{c}_0$ denotes the applied solute concentration. If multiple solutes are considered, the initial and boundary conditions for each solute must be specified.

2.3 Weak formulations

In this study, we utilized a mixed Galerkin finite element method for the Richards equation and the standard Galerkin finite element method for the solute transport equation with linear basis functions.

The notations employed in this study follow the standard conventions of functional

analysis. The space $L^2(\Omega)$ represents the set of real-valued functions that are square integrable over the domain Ω and $H^1(\Omega)$ its subspace of functions with first-order weak derivatives that are also square integrable over Ω . Moreover, $H_D^1(\Omega)$ denotes a specific subset of $H^1(\Omega)$, characterized by functions that have a trace of zero on the Dirichlet boundary. Lastly, the Bochner space $L^2(I, X)$ combines functions that are square integrable over the time interval I and have values in the Banach space X .

We aim to solve the Richards equation (2.3) using a mixed weak formulation. This formulation, as derived in (Keita *et al.*, 2021b) to construct the semi-implicit (S, Ψ) -scheme, involves a combination of equations (2.3) and (2.4). The resulting expression for the mixed weak formulation is as follows.

Find $(S, \Psi) \in L^2(I, H^1(\Omega)^2)$ such that:

$$\begin{cases} S - S_D \in L^2(I, H_D^1(\Omega)), \\ \int_{\Omega} \phi \frac{\partial S}{\partial t} v_1 \, d\Omega + \int_{\Omega} [K \nabla(\Psi + z)] \cdot \nabla v_1 \, d\Omega + \int_{\Gamma_N} q_N v_1 \, d\Gamma_N = 0, \\ \forall v_1 \in H_D^1(\Omega) \text{ and a.e. (almost everywhere) in } I, \\ \int_{\Omega} \Psi v_2 \, d\Omega = \int_{\Omega} h_{\text{cap}} J(S) v_2 \, d\Omega, \quad \forall v_2 \in H_D^1(\Omega) \text{ and a.e. in } I. \end{cases} \quad (2.15)$$

In order to ensure the existence and uniqueness of the solution to problem (2.15), we assume that all theoretical hypotheses are satisfied. For a detailed analysis of the existence and uniqueness of weak solutions for the Richards equation (2.1), we refer interested readers to (Alt and Luckhaus, 1983; Marinoschi, 2006; Van Duyn and Peletier, 1982). It is worth noting, as mentioned in (Keita *et al.*, 2021b), that this formulation differs from the standard mixed finite element approach, where the variables are typically the pressure head and the water flux. In our study, we seek to find both solutions within the same space, leading to a smaller linear system that needs to be solved. The (S, q) -scheme (Keita *et al.*, 2021b) is more time-consuming and results in a larger linear system, leading to overly expensive numerical solutions. Therefore, our work serves as a continuation of the study in (Keita *et al.*, 2021b), aiming to further investigate the semi-implicit (S, Ψ) -scheme in a more general form.

A weak formulation for the solute transport equation (5.5) is as follows: Find $c \in L^2(I, H^1(\Omega))$ such that:

$$\begin{cases} c - c_D \in L^2(I, H_D^1(\Omega)), \\ \int_{\Omega} \frac{\partial R\theta c}{\partial t} w \, d\Omega + \int_{\Omega} (\theta \mathbf{D} \nabla c - c \mathbf{q}) \cdot \nabla w \, d\Omega + \int_{\Gamma_N} q_N \bar{c}_0 w \, d\Gamma_N = 0, \\ \forall w \in H_D^1(\Omega) \text{ and a.e. in } I. \end{cases} \quad (2.16)$$

2.4 Numerical methods

To discretize the time interval I , we divide it into N subintervals with a time step Δt . Within this subdivision, we define time points as $t_n = n\Delta t$, where n ranges from 0 to

N . Here, N is a strictly positive integer. Additionally, we consider the partitioning of the domain $\bar{\Omega}$ into elements denoted by Ω^e , where $\mathcal{T}_h$ represents the set of all these elements. The choice of Ω^e depends on the dimensionality of the problem: for $d = 1$, Ω^e is a segment; for $d = 2$, Ω^e can be a triangle; and for higher dimensions, Ω^e can be a closed d -simplex. The mesh size h of the partition $\mathcal{T}_h$ is defined as follows:

$$h = \max_{\mathcal{T}_h} \delta(\Omega^e), \quad (2.17)$$

where $\delta(\Omega^e)$ is the diameter of the element Ω^e .

It is assumed that $\bar{\Omega}$ is a polygonal domain. The Galerkin finite element space V_h is defined as follows:

$$V_h = \left\{ v_h \in H^1(\Omega) ; v_h|_{\Omega^e} \in \mathcal{P}_k(\Omega^e), \forall \Omega^e \in \mathcal{T}_h \right\},$$

where $\mathcal{P}_k(\Omega^e)$ represents the space of polynomials of degree k on element Ω^e . For this study, a linear polynomial space ($k = 1$) was utilized. We approximate $S(\cdot, t_n)$ by $S_h^n \in V_h$, $\Psi(\cdot, t_n)$ by $\Psi_h^n \in V_h$ and $c(\cdot, t_n)$ by $c_h^n \in V_h$, for all $n = 0, 1, \dots, N$.

2.4.1 Time stepping schemes for the Richards equation

2.4.1.1 Semi-implicit (S, Ψ) -scheme

The semi-implicit (S, Ψ) -scheme was developed by combining the mixed weak formulation (2.15) and incorporating specific numerical methods. For the time discretization, a second-order backward differentiation formula (BDF2) was employed. The unsaturated hydraulic conductivity $K(\Psi, \mathbf{x})$ was approximated using a second-order extrapolation formula, while the Leverett function $J(S)$ was approximated using a Taylor expansion with second-order truncation error. This reads as follows:

$$\begin{cases} K(\Psi_h^{n+1}) \simeq 2K(\Psi_h^n) - K(\Psi_h^{n-1}), \\ J(S_h^{n+1}) \simeq J(S_h^n) + J'(S_h^n)(S_h^{n+1} - S_h^n). \end{cases} \quad (2.18)$$

Here, we have omitted the spatial coordinates dependency in the function K for the sake of notation simplicity.

This approach based on extrapolation and Taylor expansion allows to avoid the need for iterative computations. The approximation of the J function in (2.18) requires the computation of the derivative of J . To address the degeneracy of the Leverett function mentioned before, we adopt the regularization technique used in (Keita *et al.*, 2021b). This regularization is defined as follows:

$$J'_\varepsilon(S_h) = \begin{cases} J'(S_h) & \text{in } (0, 1 - \varepsilon), \\ J'(1 - \varepsilon) & \text{in } [1 - \varepsilon, 1], \end{cases} \quad (2.19)$$

where $0 < \varepsilon \ll 1$. For notation simplicity, we will also denote J'_ε by J' . The regularization technique used in this context is designed to address the behavior of the derivative of J when the saturation level reaches $1 - \varepsilon$. Instead of allowing the derivative to rapidly increase to infinity in the vicinity of the saturated condition, the regularization approach limits the derivative to a maximal value of $J'(1 - \varepsilon)$ and keeps it constant thereafter. It was exclusively applied to the van Genuchten model, with $\varepsilon = 10^{-6}$.

The approximation (2.18) requires two initial data, namely S_h^0 and S_h^1 . S_h^0 represents the initial data S_0 given by (6.33), while S_h^1 can be determined using either an implicit scheme with backward Euler in time or a semi-implicit scheme as used in (Keita *et al.*, 2021a). This reads as follows: Find $(S_h^1, \Psi_h^1) \in V_h^2$ such that:

$$\begin{cases} S_h^0 = \Pi_h S_0, \text{ and, } \Psi_h^0 = h_{\text{cap}} J(S_h^0), \\ \int_{\Omega} \phi_h \frac{S_h^1 - S_h^0}{\Delta t} v_h \, d\Omega + \int_{\Omega} K(\Psi_h^0) \nabla(\Psi_h^1 + z) \cdot \nabla v_h \, d\Omega \\ \quad = - \int_{\Gamma_N} q_N v_h \, d\Gamma_N, \quad \forall v_h \in V_h \text{ and a.e. in } I, \\ \int_{\Omega} \Psi_h^1 w_h \, d\Omega = \int_{\Omega} h_{\text{cap}} J(S_h^0) w_h \, d\Omega, \quad \forall w_h \in V_h \text{ and a.e. in } I, \end{cases} \quad (2.20)$$

where Π_h is an interpolation or projection operator on V_h .

By incorporating (2.18) into (2.15), we obtain the semi-implicit (S, Ψ) -scheme:

$$\begin{cases} \text{Let } n = 1, \dots, N, \text{ and } S_h^{n-1}, S_h^n, \Psi_h^{n-1}, \Psi_h^n \in V_h \text{ be given. Find } (S_h^{n+1}, \Psi_h^{n+1}) \in V_h^2 \text{ such that:} \\ \int_{\Omega} \phi_h \frac{3S_h^{n+1} - 4S_h^n + S_h^{n-1}}{2\Delta t} v_h \, d\Omega + \int_{\Omega} \left[(2K(\Psi_h^n) - K(\Psi_h^{n-1})) \nabla(\Psi_h^{n+1} + z) \right] \cdot \nabla v_h \, d\Omega \\ \quad = - \int_{\Gamma_N} q_N v_h \, d\Gamma_N, \quad \forall v_h \in V_h \text{ and a.e. in } I, \\ \int_{\Omega} \Psi_h^{n+1} w_h \, d\Omega = \int_{\Omega} h_{\text{cap}} \left[J(S_h^n) + J'(S_h^n)(S_h^{n+1} - S_h^n) \right] w_h \, d\Omega, \quad \forall w_h \in V_h \text{ and a.e. in } I. \end{cases} \quad (2.21)$$

The numerical scheme (2.21) leads to a linear system that does not necessitate any iterative process. This simplicity makes it much easier to solve when compared to a fully implicit approach.

Remark 1 *The numerical results in (Keita *et al.*, 2021b) demonstrate that the numerical scheme (2.21) performs better in terms of accuracy and efficiency compared to the scheme using only saturation as the sole unknown.*

2.4.1.2 Second-order generalized semi-implicit (S, Ψ) -schemes

Inspired by second-order IMEX schemes (Ascher *et al.*, 1995) and building upon the semi-implicit (S, Ψ) -scheme (Keita *et al.*, 2021b), a generalization approach involves introducing free parameters in the extrapolation technique (2.18) applied to the unsaturated hydraulic conductivity and the Leverett function. This can be written as follows:

$$\begin{cases} K(\Psi_h^{n+\alpha}) \simeq (\alpha + 1)K(\Psi_h^n) - \alpha K(\Psi_h^{n-1}), \\ J(S_h^{n+\alpha}) \simeq (\alpha + \frac{\beta}{2})J(S_h^{n+1}) + (1 - \alpha - \beta)J(S_h^n) + \frac{\beta}{2}J(S_h^{n-1}), \end{cases} \quad (2.22)$$

where α and β are free parameters in $[0, 1]$. Once again, we employ the Taylor expansion (2.18) to linearize the term $J(S_h^{n+1})$. The detailed discussion of the numerical treatments in (2.22) is provided in Appendix 2.6.

The approximations (2.22) will be applied across all proposed schemes, with differences arising solely in the handling of the time derivative. The treatment of the time derivative is a crucial factor in obtaining accurate results for the Richards equation (Celia *et al.*, 1990a; Kavetski *et al.*, 2001; Keita *et al.*, 2021b). In the following sections, we present two distinct approaches for approximating the time derivative of saturation.

First type of generalized semi-implicit (S, Ψ) -schemes. The first approximation is obtained by centering around time $t_{n+\alpha}$ to get:

$$\left(\frac{\partial S_h}{\partial t} \right)_{n+\alpha} \simeq \frac{\left(\alpha + \frac{1}{2} \right) S_h^{n+1} - 2\alpha S_h^n + \left(\alpha - \frac{1}{2} \right) S_h^{n-1}}{\Delta t}. \quad (2.23)$$

Based on the time derivative approximation (2.23) and incorporating (2.22), the first version of the proposed generalized two free parameters semi-implicit (S, Ψ) -scheme is given as follows:

$$\left\{ \begin{array}{l} \text{Let } n = 1, \dots, N, \text{ and } S_h^{n-1}, S_h^n, \Psi_h^{n-1}, \Psi_h^n \in V_h \text{ be given. Find } (S_h^{n+1}, \Psi_h^{n+\alpha}) \in V_h^2 \text{ such that:} \\ \int_{\Omega} \phi_h \frac{\left(\alpha + \frac{1}{2} \right) S_h^{n+1} - 2\alpha S_h^n + \left(\alpha - \frac{1}{2} \right) S_h^{n-1}}{\Delta t} v_h \, d\Omega + \int_{\Omega} \left[\left((\alpha + 1)K(\Psi_h^n) - \alpha K(\Psi_h^{n-1}) \right) \right. \\ \quad \left. \times \nabla (\Psi_h^{n+\alpha} + z) \right] \cdot \nabla v_h \, d\Omega = - \int_{\Gamma_N} q_N v_h \, d\Gamma_N, \quad \forall v_h \in V_h \text{ and a.e. in } I, \\ \int_{\Omega} \Psi_h^{n+\alpha} w_h \, d\Omega = \int_{\Omega} h_{\text{cap}} \left[\left(\alpha + \frac{\beta}{2} \right) J'(S_h^n)(S_h^{n+1} - S_h^n) + \left(1 - \frac{\beta}{2} \right) J(S_h^n) + \frac{\beta}{2} J(S_h^{n-1}) \right] w_h \, d\Omega, \\ \quad \forall w_h \in V_h \text{ and a.e. in } I. \end{array} \right. \quad (2.24)$$

where β is a free parameter in $[0, 1]$. We utilized equation (2.20) to calculate S_h^1 based on S_h^0 . In the case of $(\alpha, \beta) = (1, 0)$, we obtained the original semi-implicit (S, Ψ) -scheme proposed in (Keita *et al.*, 2021b). Specifically, for certain values of α and β , the following names were assigned to the schemes:

- **SBDF** (Semi-implicit BDF2) $(\alpha, \beta) = (1, 0)$:

$$\begin{cases} \phi_h \frac{3S_h^{n+1} - 4S_h^n + S_h^{n-1}}{2\Delta t} = \nabla \cdot \left[\left(2K(\Psi_h^n) - K(\Psi_h^{n-1}) \right) \nabla (\Psi_h^{n+1} + z) \right], \\ \Psi_h^{n+1} = h_{\text{cap}} \left[J(S_h^n) + J'(S_h^n)(S_h^{n+1} - S_h^n) \right], \end{cases} \quad (2.25)$$

which is the semi-implicit (S, Ψ) -scheme.

- **MSBDF** (Modified semi-implicit BDF2) $(\alpha, \beta) = (1, 1)$:

$$\begin{cases} \phi_h \frac{3S_h^{n+1} - 4S_h^n + S_h^{n-1}}{2\Delta t} = \nabla \cdot \left[\left(2K(\Psi_h^n) - K(\Psi_h^{n-1}) \right) \nabla (\Psi_h^{n+1} + z) \right], \\ \Psi_h^{n+1} = h_{\text{cap}} \left[\frac{1}{2} (J(S_h^n) + J(S_h^{n-1})) + \frac{3}{2} J'(S_h^n) (S_h^{n+1} - S_h^n) \right]. \end{cases} \quad (2.26)$$

- **Richards-AB** $(\alpha, \beta) = \left(\frac{1}{2}, 0\right)$:

$$\begin{cases} \phi_h \frac{S_h^{n+1} - S_h^n}{\Delta t} = \nabla \cdot \left[\left(\frac{3}{2} K(\Psi_h^n) - \frac{1}{2} K(\Psi_h^{n-1}) \right) \nabla (\Psi_h^{n+1/2} + z) \right], \\ \Psi_h^{n+1/2} = h_{\text{cap}} \left[J(S_h^n) + \frac{1}{2} J'(S_h^n) (S_h^{n+1} - S_h^n) \right]. \end{cases} \quad (2.27)$$

- **Richards-MAB** $(\alpha, \beta) = \left(\frac{1}{2}, 1\right)$:

$$\begin{cases} \phi_h \frac{S_h^{n+1} - S_h^n}{\Delta t} = \nabla \cdot \left[\left(\frac{3}{2} K(\Psi_h^n) - \frac{1}{2} K(\Psi_h^{n-1}) \right) \nabla (\Psi_h^{n+1/2} + z) \right], \\ \Psi_h^{n+1/2} = h_{\text{cap}} \left[\frac{1}{2} (J(S_h^n) + J(S_h^{n-1})) + J'(S_h^n) (S_h^{n+1} - S_h^n) \right]. \end{cases} \quad (2.28)$$

The incorporation of the variable S_h^{n+1} in the second equation of the above schemes demonstrates their semi-implicit nature.

Remark 2 The “AB” in Richards-AB and Richards-MAB refers to Adams-Bashforth with “M” for modified. This is because the K function extrapolation resembles the Adams-Bashforth methods. However, the full-time discretization differs significantly from the true Adams-Bashforth scheme.

Second type of generalized semi-implicit (S, Ψ) -schemes. For the second approximation, we used the following second-order trapezoid scheme, which was adopted in (Diersch and Perrochet, 1999; Kavetski *et al.*, 2001):

$$\left(\frac{\partial S_h}{\partial t} \right)_{n+1} = \dot{S}_h^{n+1} \simeq \frac{2(S_h^{n+1} - S_h^n) - \Delta t \dot{S}_h^n}{\Delta t}, \quad (2.29)$$

where $\dot{S}_h^n = \left(\frac{\partial S_h}{\partial t} \right)_n$. Note that $\dot{S}_h^n$ is not approximated using formula (2.23). The approximation will be provided as follows. To determine the initialization for the sequence (2.29), we initially adopt the same approach as employed in (Diersch and Perrochet, 1999; Kavetski *et al.*, 2001). By incorporating the time derivative approximation (2.29)

along with the approximations (2.18) in the Richards equation at time t_{n+1} , we derive the following numerical scheme, referred to as **Richards-M1**:

$$\left\{ \begin{array}{l} \text{Let } n = 1, \dots, N, \text{ and } S_h^n, \dot{S}_h^n, \Psi_h^{n-1}, \Psi_h^n \in V_h \text{ be given. Find } (S_h^{n+1}, \Psi_h^{n+1}) \in V_h^2 \text{ such that:} \\ \int_{\Omega} \phi_h \frac{2(S_h^{n+1} - S_h^n) - \Delta t \dot{S}_h^n}{\Delta t} v_h \, d\Omega + \int_{\Omega} \left[\left(2K(\Psi_h^n) - K(\Psi_h^{n-1}) \right) \right. \\ \quad \left. \times \nabla (\Psi_h^{n+1} + z) \right] \cdot \nabla v_h \, d\Omega = - \int_{\Gamma_N} q_N v_h \, d\Gamma_N, \quad \forall v_h \in V_h \text{ and a.e. in } I, \\ \int_{\Omega} \Psi_h^{n+1} w_h \, d\Omega = \int_{\Omega} h_{\text{cap}} \left[J(S_h^n) + J'(S_h^n)(S_h^{n+1} - S_h^n) \right] w_h \, d\Omega, \\ \quad \forall w_h \in V_h \text{ and a.e. in } I. \end{array} \right. \quad (2.30)$$

An alternative approach to handle the term $\dot{S}_h^n$ in (2.29) is to employ the following Leap-frog formula:

$$\dot{S}_h^n = \frac{1}{\phi_h} C(\Psi_h^n) \dot{\Psi}_h^n \simeq \frac{1}{\phi_h} C(\Psi_h^n) \frac{\Psi_h^{n+1} - \Psi_h^{n-1}}{2\Delta t}, \quad (2.31)$$

where $C = \frac{d\theta}{d\Psi}$ is the specific water capacity. The first equation in (2.31) is derived using the chain rule as follows:

$$\phi_h \frac{\partial S_h}{\partial t} = \frac{\partial \theta_h}{\partial t} = C_h \frac{\partial \Psi_h}{\partial t}. \quad (2.32)$$

Note that we utilized the second-order accurate approximation of $\dot{\Psi}_h^n$ in (2.31) to maintain the overall second-order accuracy of the proposed scheme. However, the introduction of the nonlinear capacity term C may lead to mass balance errors (Celia *et al.*, 1990a).

Finally, substituting the approximation (2.31) into (2.29), we obtain the following expression:

$$\phi_h \left(\frac{\partial S_h}{\partial t} \right)_{n+1} \simeq \frac{4\phi_h (S_h^{n+1} - S_h^n) - C(\Psi_h^n) (\Psi_h^{n+1} - \Psi_h^{n-1})}{2\Delta t}. \quad (2.33)$$

The **Richards-M2** scheme is obtained by employing the time derivative approximation

(2.33) along with the approximations (2.18) in the Richards equation at time $n + 1$:

$$\left\{ \begin{array}{l} \text{Let } n = 1, \dots, N, \text{ and } S_h^{n-1}, S_h^n, \Psi_h^{n-1}, \Psi_h^n \in V_h \text{ be given. Find } (S_h^{n+1}, \Psi_h^{n+1}) \in V_h^2 \text{ such that:} \\ \int_{\Omega} \frac{4\phi_h(S_h^{n+1} - S_h^n) - C(\Psi_h^n)(\Psi_h^{n+1} - \Psi_h^{n-1})}{2\Delta t} v_h \, d\Omega + \int_{\Omega} \left[(2K(\Psi_h^n) - K(\Psi_h^{n-1})) \right. \\ \quad \left. \times \nabla(\Psi_h^{n+1} + z) \right] \cdot \nabla v_h \, d\Omega = - \int_{\Gamma_N} q_N v_h \, d\Gamma_N, \quad \forall v_h \in V_h \text{ and a.e. in } I, \\ \int_{\Omega} \Psi_h^{n+1} w_h \, d\Omega = \int_{\Omega} h_{\text{cap}} \left[J(S_h^n) + J'(S_h^n)(S_h^{n+1} - S_h^n) \right] w_h \, d\Omega, \\ \quad \forall w_h \in V_h \text{ and a.e. in } I. \end{array} \right. \quad (2.34)$$

Remark 3 The free parameter β can be incorporated into the second equation in the Richards-M1 and Richards-M2 schemes. In our analysis, we choose the value $\beta = 0$.

Remark 4 To mitigate the occurrence of oscillatory solutions, we employed a mass lumping technique (Celia et al., 1990a). Specifically, we opted for the row-sum technique, which involves summing all the terms in each row of the mass matrix and placing the result on the diagonal. This technique has the advantage of simplicity and effectiveness in reducing oscillations.

A summary of the proposed numerical schemes for the Richards equation is presented in Table 2.1.

2.4.2 Time stepping schemes for the solute transport equation

For the solute transport equation (5.5), we employed IMEX schemes (Ascher et al., 1995). By employing the linear Galerkin finite element method, we obtain the following semi-discrete system:

$$\frac{d(\mathbf{M}c_h)}{dt} + (\mathbf{K} + \mathbf{L})c_h = \mathbf{F}. \quad (2.35)$$

Table 2.1 A summary of the proposed semi-implicit schemes. The table provides a reference to the numerical treatment of each term in the Richards equation.

Scheme	$\partial_t S_h$	$K(\Psi_h)$	$J(\Psi_h)$	α	β	Reference
First type of generalized semi-implicit schemes						
SBDF	(2.23)	(2.22)	(2.22)	1	0	(2.25)
MSBDF	(2.23)	(2.22)	(2.22)	1	1	(2.26)
Richards-AB	(2.23)	(2.22)	(2.22)	0.5	0	(2.27)
Richards-MAB	(2.23)	(2.22)	(2.22)	0.5	1	(2.28)
Second type of generalized semi-implicit schemes						
Richards-M1	(2.29)	(2.22)	(2.22)	1	0	(2.30)
Richards-M2	(2.33)	(2.22)	(2.22)	1	0	(2.34)

The matrices involved in equation (2.35) are provided as follows:

$$\begin{cases} \mathbf{M}_{ij} = \int_{\Omega} R_h \theta_h \varphi_i \varphi_j \, d\Omega, & \mathbf{K}_{ij} = \int_{\Omega} \theta_h \mathbf{D}_h \nabla \varphi_i \cdot \nabla \varphi_j \, d\Omega, \\ \mathbf{L}_{ij} = - \int_{\Omega} [\mathbf{q}_h \cdot \nabla \varphi_i] \varphi_j \, d\Omega, & \mathbf{F}_i = - \int_{\Gamma_N} q_N \bar{c}_0 \varphi_i \, d\Gamma_N, \end{cases} \quad (2.36)$$

where $(\varphi_i)_{i=1}^n$ denotes a suitable basis function set for V_h , $\mathbf{M}$ is the mass matrix, the matrix $\mathbf{K}$ represents the diffusive part, while the matrix $\mathbf{L}$ accounts for the advective part. This decomposition enables the application of different time-stepping schemes for these parts. To mitigate numerical oscillations, a row-sum mass lumping technique is applied to the matrix $\mathbf{M}$, resulting in the updated mass matrix:

$$\mathbf{M}_{ij} = \delta_{ij} \int_{\Omega} R_h \theta_h \varphi_i \, d\Omega. \quad (2.37)$$

The proposed temporal schemes are outlined as follows: A first-order IMEX scheme is applied to equation (2.35):

$$\frac{(\mathbf{M}c_h)^{n+1} - (\mathbf{M}c_h)^n}{\Delta t} + (\mathbf{L}c_h)^n + \alpha(\mathbf{K}c_h)^{n+1} + (1 - \alpha)(\mathbf{K}c_h)^n = \mathbf{F}^{n+1}. \quad (2.38)$$

A second-order IMEX scheme is applied to equation (2.35):

$$\begin{aligned} & \frac{\left(\alpha + \frac{1}{2}\right)(\mathbf{M}c_h)^{n+1} - 2\alpha(\mathbf{M}c_h)^n + \left(\alpha - \frac{1}{2}\right)(\mathbf{M}c_h)^{n-1}}{\Delta t} + \left((\alpha + 1)(\mathbf{L}c_h)^n - \alpha(\mathbf{L}c_h)^{n-1}\right) + \\ & \left(\left(\alpha + \frac{\beta}{2}\right)(\mathbf{K}c_h)^{n+1} + (1 - \alpha - \beta)(\mathbf{K}c_h)^n + \frac{\beta}{2}(\mathbf{K}c_h)^{n-1}\right) = \alpha\mathbf{F}^{n+1} + (1 - \alpha)\mathbf{F}^n. \end{aligned} \quad (2.39)$$

The advective part, reflecting the hyperbolic nature of the solute transport equation, is discretized explicitly. Meanwhile, the diffusive part, reflecting the parabolic nature of the equation, is discretized using an implicit scheme (Ascher *et al.*, 1995). The same abbreviations utilized for the proposed schemes (2.25)-(2.28) are employed for the IMEX schemes proposed for the transport equation. As previously mentioned, our focus will be exclusively on second-order IMEX schemes. For the initialization of the second-order time-stepping schemes (2.39), the first-order IMEX scheme (2.38) was used. The solute transport equation can incorporate reaction terms, as demonstrated in the numerical test in Section 2.5.2.3. In such cases, the treatment used for the right-hand-side $\mathbf{F}$ in the scheme (2.39) will be applied.

The general algorithm of modeling water flow and solute transport in unsaturated soils is illustrated in Figure 2.2.

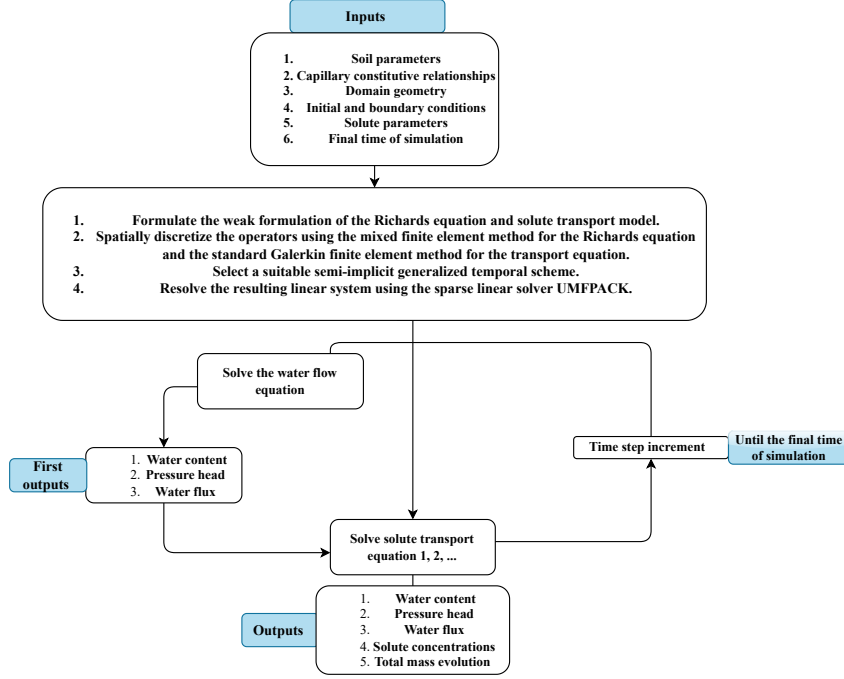


Figure 2.2 Flow chart of the coupled numerical model of water flow and solute transport.

2.5 Numerical experiments

In this section, several numerical tests are conducted to evaluate the performance of the proposed numerical schemes. Numerical experiments presented in this paper were carried out using the supercomputer simlab-cluster, supported by University Mohammed VI Polytechnic, and facilities of simlab-cluster HPC & IA platform. We employed MATLAB for visualization, and the main program was implemented using FreeFEM++ software (Hecht, 2012). The linear systems obtained by the proposed schemes were solved using the sparse linear solver UMFPACK (Davis, 2004).

To compute the error between the numerical and reference solutions, we used the L^2 and H^1 -errors at the final time of the simulation T :

$$\|u - u_h\|_{L^2(\Omega)}^2 = \int_{\Omega} |u - u_h|^2 \, d\Omega, \quad \|u - u_h\|_{H^1(\Omega)}^2 = \|u - u_h\|_{L^2(\Omega)}^2 + \|\nabla(u - u_h)\|_{L^2(\Omega)}^2, \quad (2.40)$$

where u represents the reference solution, and u_h denotes the numerical approximation. The order of convergence of a numerical method can be determined using the following formula:

$$p = \log \left(\frac{\|u - u_{\tau_1}\|}{\|u - u_{\tau_2}\|} \right) / \log(\tau_1/\tau_2), \quad (2.41)$$

where τ_i represents a mesh size or a time step value, and $\|\cdot\|$ denotes either the L^2 or the H^1 -norms.

To numerically assess the total mass of water and solute at time t , we employed the following formulas:

$$T_w M(t) = \int_{\Omega} \theta_h(x, z, t) \, d\Omega, \quad (2.42)$$

$$T_s M(t) = \int_{\Omega} \theta_h(x, z, t) c_h(x, z, t) \, d\Omega, \quad (2.43)$$

where $T_w M(t)$ and $T_s M(t)$ represent the total water mass and total solute mass, respectively at time t . These quantities will be compared to an exact or a reference solution. The relative error in the mass $R_e M(t)$ can also be used to quantify mass error, and it is defined as (Celia *et al.*, 1990a):

$$R_e M(t) = \left| \frac{\Delta T_e - \Delta T_n}{\Delta T_e} \right|, \quad (2.44)$$

where $\Delta T_i = T_i M(t) - T_i M(0)$, and $i = n, e$ represent the numerical and exact total quantities of mass in the computational domain, respectively.

2.5.1 Water infiltration

In the following subsections, we evaluate the effectiveness of the proposed numerical schemes for simulating water infiltration. Two numerical tests are considered, with the first one involving homogeneous soil, where we assess the accuracy, stability, and mass conservation of the proposed schemes to select the most accurate one. The second, presenting a more challenging scenario with heterogeneous soil, serves as confirmation for the first test. For further analysis of the effective numerical method, time order accuracy was numerically demonstrated for SBDF (Keita *et al.*, 2021b) using a manufactured solution.

2.5.1.1 2D homogeneous soil

In this section, our objective is to assess how the free parameters α and β impact the accuracy, stability, and mass conservation of the proposed schemes for the Richards equation. We conducted two numerical tests for a two-dimensional infiltration problem within a rectangular domain $[0, l] \times [0, L]$, as illustrated in Figure 2.3. These tests provide the advantage of having available analytical solutions. The Gardner constitutive model for the capillary pressure is employed in these two tests, where the respective relationships are defined in Table 2.2. The $\tilde{\alpha}$ parameter represents a soil parameter related to the pore size, specifically governing the rate at which the unsaturated hydraulic conductivity $K(\Psi)$ diminishes with changes in pressure head Ψ . In the first test case, the soil domain is subject to the following initial and boundary conditions

Table 2.2 Soil hydraulic functions for different models.

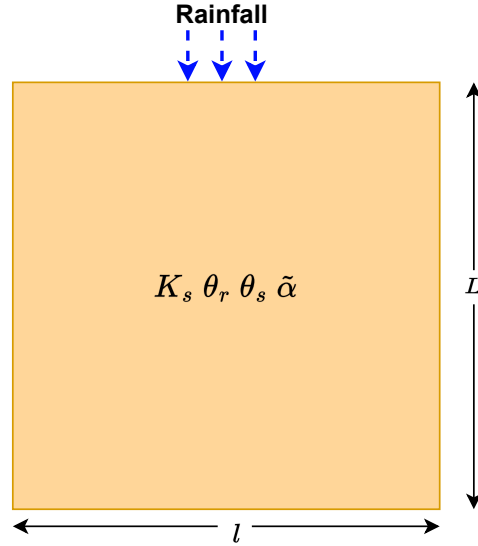
Model	$S(\Psi)$	$K(\Psi)$	$J(S)$
Gardner	$\exp(\tilde{\alpha}\Psi)$	$K_s S(\Psi)$	$\log(S)$
Haverkamp	$\frac{1}{1 + \tilde{\alpha}\Psi ^{\beta_0}}$	$K_s \frac{1}{1 + A\Psi ^\gamma}$	$-(S^{-1} - 1)^{1/\beta_0}$
van Genuchten-Mualem*	$(1 + \tilde{\alpha}\Psi ^{n_v})^{-m_v}$	$K_s S^{0.5} \left[1 - (1 - S^{1/m_v})^{m_v}\right]^2$	$-(S^{-1/m_v} - 1)^{1/n_v}$

* van Genuchten-Mualem means that the water retention $S(\Psi)$ is modeled using the van Genuchten model; and the unsaturated hydraulic conductivity is modeled by the Mualem model, where $m_v = 1 - \frac{1}{n_v}, n_v > 1$.

(see Figure 2.4):

$$\begin{cases} \Psi(x, z, 0) = \Psi_d & \text{for } (x, z) \in [0, l] \times [0, L], \\ \Psi(0, z, t) = \Psi(l, z, t) = \Psi_d & \text{for } (t, z) \in (0, T] \times [0, L], \\ \Psi(x, 0, t) = \Psi_d & \text{for } (t, x) \in (0, T] \times [0, l], \\ \Psi(x, L, t) = \frac{1}{\tilde{\alpha}} \ln \left(\xi + (1 - \xi) \sin \left(\frac{\pi x}{l} \right) \right) & \text{for } (t, x) \in (0, T] \times [0, l], \end{cases} \quad (2.45)$$

where Ψ_d is the initial pressure head, $\xi = \exp(\tilde{\alpha}\Psi_d)$. For the second test case, the soil

Figure 2.3 2D homogeneous unsaturated soil $[0, l] \times [0, L]$.

domain is subject to the following initial and boundary conditions (see Figure 2.4):

$$\begin{cases} \Psi(x, z, 0) = \Psi_d & \text{for } (x, z) \in [0, l] \times [0, L], \\ \Psi(x, 0, t) = \Psi_d & \text{for } (t, x) \in (0, T] \times [0, l], \\ q_N(0, z, t) = q_N(l, z, t) = 0 & \text{for } (t, z) \in (0, T] \times [0, L], \\ \Psi(x, L, t) = \frac{1}{\tilde{\alpha}} \ln \left[\xi + \frac{(1 - \xi)}{2} \left(1 - \cos \left(\frac{2\pi x}{l} \right) \right) \right] & \text{for } (t, x) \in (0, T] \times [0, l], \end{cases} \quad (2.46)$$

The pressure head boundary conditions implemented at the upper surface of the soil sample for the two test cases are represented in Figure 2.5. The analytical solution, denoted as Ψ_i , is provided for test cases $i = 1$ and $i = 2$ by (Tracy, 2006):

$$\begin{aligned} \Psi_i &= \frac{1}{\tilde{\alpha}} \ln(\xi + \bar{\psi}_i), \quad \text{where} \\ \bar{\psi}_1(x, z, t) &= (1 - \xi) \sin \left(\frac{\pi x}{l} \right) \exp \left(\frac{\tilde{\alpha}}{2} (L - z) \right) \left(\frac{\sinh(\beta_1 z)}{\sinh(\beta_1 L)} + \frac{2}{Ld} \sum_{j=1}^{\infty} (-1)^j \frac{\lambda_j}{\gamma_j} \sin(\lambda_j z) e^{-\gamma_j t} \right), \\ \bar{\psi}_2(x, z, t) &= \frac{(1 - \xi)}{2} \exp \left(\frac{\tilde{\alpha}}{2} (L - z) \right) \left(\frac{\sinh \left(\frac{\tilde{\alpha} z}{2} \right)}{\sinh \left(\frac{\tilde{\alpha} L}{2} \right)} - \cos \left(\frac{2\pi x}{l} \right) \frac{\sinh(\beta_2 z)}{\sinh(\beta_2 L)} + \frac{2}{Ld} v(x, z, t) \right), \\ \beta_1 &= \sqrt{\frac{\tilde{\alpha}^2}{4} + \left(\frac{\pi}{l} \right)^2}, \quad d = \frac{\tilde{\alpha}(\theta_s - \theta_r)}{K_s}, \quad \lambda_j = \frac{\pi}{L} j, \quad \gamma_j = \frac{1}{d} (\beta_1^2 + \lambda_j^2), \quad \beta_2 = \sqrt{\frac{\tilde{\alpha}^2}{4} + \left(\frac{2\pi}{l} \right)^2}, \\ v(x, z, t) &= \sum_{j=1}^{\infty} (-1)^j \lambda_j \sin(\lambda_j z) \left[\frac{e^{-\gamma_{1j} t}}{\gamma_{1j}} - \frac{e^{-\gamma_{2j} t}}{\gamma_{2j}} \cos \left(\frac{2\pi x}{l} \right) \right], \\ \gamma_{1j} &= \frac{1}{d} \left(\frac{\tilde{\alpha}^2}{4} + \lambda_j^2 \right), \quad \gamma_{2j} = \frac{1}{d} \left(\frac{\tilde{\alpha}^2}{4} + \lambda_j^2 + \left(\frac{2\pi}{l} \right)^2 \right). \end{aligned}$$

The parameters used in the simulations are as follows: $l = L = 15.24 \text{ m}$, $\Psi_d =$

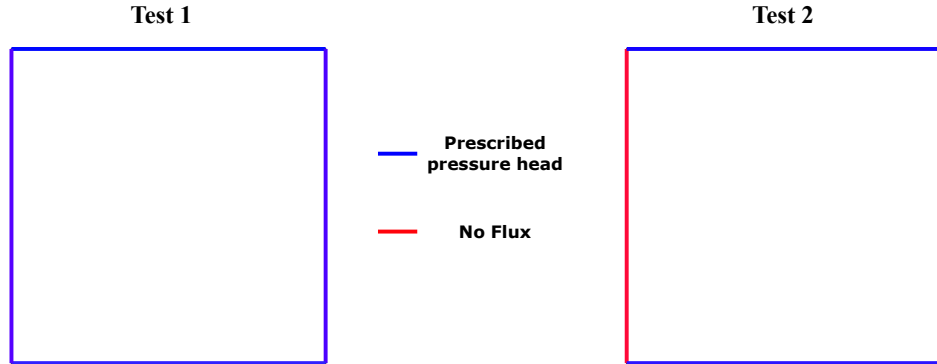


Figure 2.4 Boundary conditions employed in both two-dimensional infiltration test cases.

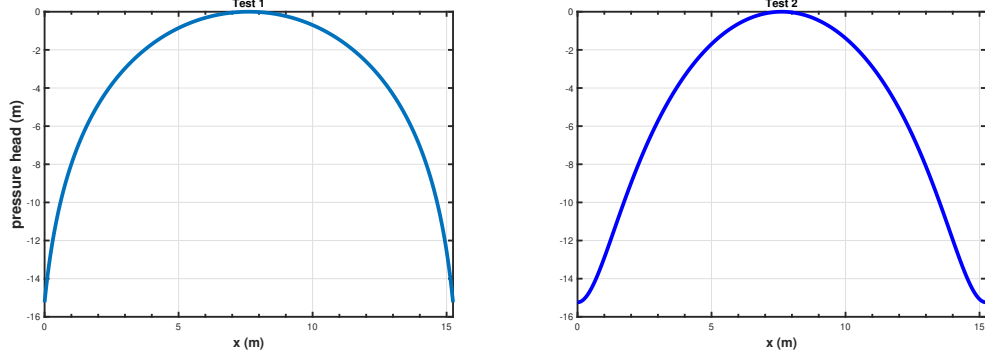


Figure 2.5 Pressure head function applied at the soil's upper boundary in both test cases.

-15.24 m, $K_s = 0.1$ m/day, $\tilde{\alpha} = 0.164$ m $^{-1}$, $\theta_s = 0.45$, $\theta_r = 0.15$. The time step is $\Delta t = 10^{-2}$ day, and the domain is discretized into a 25×25 grid.

Figure 2.6 presents the contours of the saturation profile using four semi-implicit (S, Ψ) -schemes at time $T = 10$ days. All four schemes demonstrate good agreement with the exact solution. The Richards-M1 and Richards-M2 schemes also provide accurate simulations for each test case (not shown here). To examine the impact of the free parameters (α, β) on the accuracy, we calculate the L^2 -error between the numerical and exact solutions, where the exact solution is computed using the first 200 terms of the series. Figure 2.7 illustrates the results obtained for the first test case, revealing that SBDF exhibits the lowest error. Notably, among the proposed schemes, those with $\alpha = 1$ exhibit the best accuracy. The error remains almost the same when α is fixed and β is changed. However, for a fixed value of β , the error increases significantly when α is decreased. As α approaches 0, the scheme becomes unstable, as it becomes fully explicit when $\alpha = 0$. Similar results were obtained for the second test case (not shown here). Table 2.3 presents the L^2 -error of the pressure head and the effective saturation for different semi-implicit (S, Ψ) -schemes, considering different time steps and mesh sizes. The results show that SBDF and Richards-M2 have the lowest error among the schemes, with Richards-M2 showing a particularly small error. Furthermore, MSBDF shows a comparable error to SBDF and Richards-M1, whereas both Richards-AB and Richards-MAB exhibit higher errors. As the mesh size decreases, Richards-M2 showcases higher accuracy compared to the other schemes.

Determining the temporal convergence rate of the proposed numerical schemes can be challenging due to the influence of spatial error. To address this problem, we apply the algorithm used in (Bona *et al.*, 1995) based on a modified error as explained below. A reference solution, denoted as u_{ref} , is computed using a mesh comprising 100×100 elements and a small time step $\Delta t_{\text{ref}} = 1 \times 10^{-4}$ day. The reference time step is selected to be sufficiently large to facilitate a comprehensive temporal convergence analysis while maintaining numerical stability. The reference solution differs from the exact solution

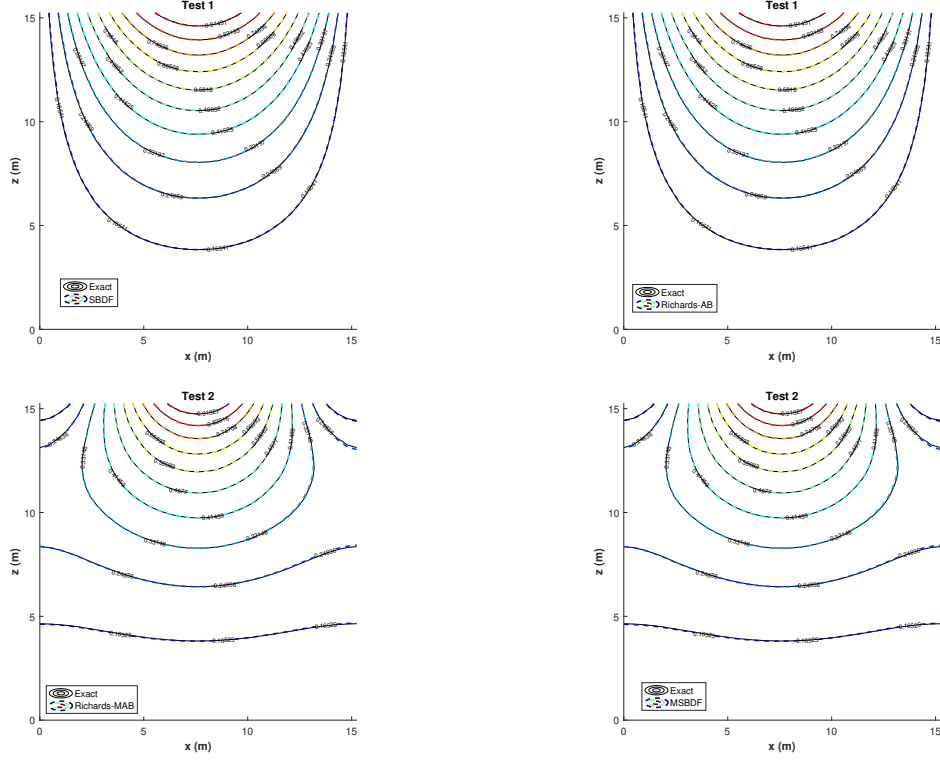


Figure 2.6 Contours of the saturation at time $T = 10$ days obtained from four semi-implicit (S, Ψ) -schemes and the exact solution with $\Delta t = 10^{-2}$ day and 25×25 elements.

u_{exact} primarily due to errors arising from spatial discretization. Additionally, for each time step Δt , we compute the associated numerical solution $u_{\Delta t}$ using the same mesh size as used for u_{ref} (100×100 elements). Finally, the modified L^2 -error is expressed as follows (Beljadid *et al.*, 2013; Bona *et al.*, 1995):

$$E_{\Delta t}^* = \frac{\|u_{\text{ref}} - u_{\Delta t}\|_{L^2(\Omega)}^2}{\|u_{\text{exact}}\|_{L^2(\Omega)}^2}. \quad (2.47)$$

By considering the numerator of (2.47), the spatial discretization error becomes negligible, as both the reference and numerical solutions $u_{\Delta t}$ are computed using the same mesh, leaving the temporal discretization as the dominant source of error.

Figure 2.8 presents the temporal convergence rates of each numerical schemes, determined using formula (2.47) across different time steps: $\Delta t = 4 \times 10^{-2}$, 2×10^{-2} , 1×10^{-2} , 5×10^{-3} . For both test cases, an approximate convergence order of 2 is observed for all schemes at the larger time steps $\Delta t = 4 \times 10^{-2}$ and 2×10^{-2} . However, at smaller time steps $\Delta t = 1 \times 10^{-2}$ and 5×10^{-3} , Richards-M2 shows an approximate convergence order of 1.68 for both Ψ_h and S_h . SBDF, MSBDF, and Richards-M1 ex-

hibit convergence orders of 1.63 for Ψ_h and 1.5 for S_h . In contrast, Richards-AB and Richards-MAB display slightly lower convergence orders of around 1.3 for Ψ_h and 1.25 for S_h . The reduction in convergence order at smaller time steps can be attributed to the increased influence of discretization errors in the computational domain. Finally, we conclude that the proposed semi-implicit techniques do not impact the second-order accuracy of the methods based on BDF2 and the trapezoid techniques.

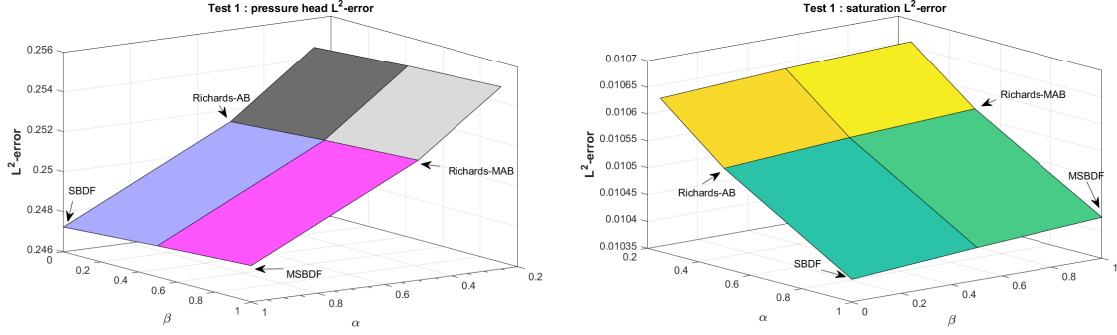


Figure 2.7 Analysis of the pressure head and the saturation L^2 -error with respect to the free parameters (α, β) with $\Delta t = 10^{-2}$ day and 25×25 elements in the first test scenario.

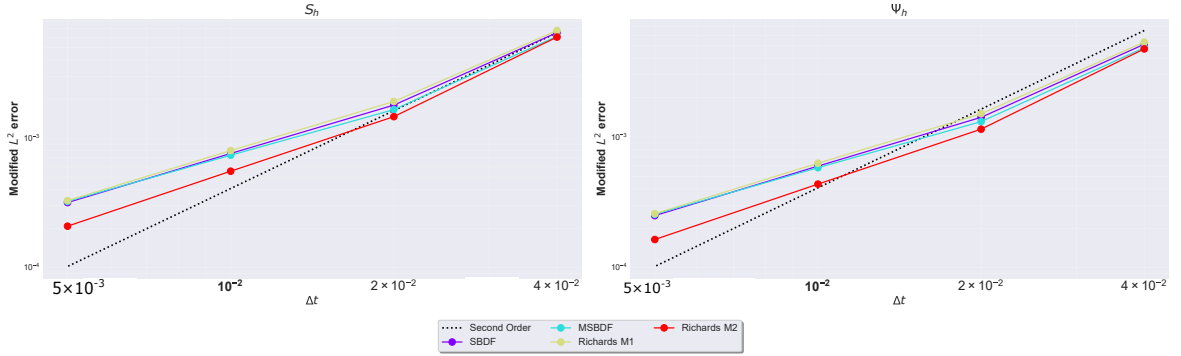


Figure 2.8 Evaluation of the modified L^2 -error in the log-log scale of the numerical schemes as a function of the time step Δt in the first test scenario.

To investigate the influence of the soil parameter $\tilde{\alpha}$, Table 2.4 displays the L^2 -errors for various semi-implicit (S, Ψ) -schemes. SBDF, MSBDF, Richards-AB, Richards-MAB, and Richards-M1 remain stable as $\tilde{\alpha}$ increases. Despite its accuracy, Richards-M2 requires smaller time steps to maintain stability, especially when gravity forces dominate over capillarity forces.

The examination of mass conservation of the proposed semi-implicit (S, Ψ) -schemes was performed using both test cases. Figure 2.9 illustrates the evolution of the total mass and the relative mass error in the computational domain for the numerical schemes

Table 2.3 The computed pressure head and saturation L^2 -errors for the semi-implicit (S, Ψ) -schemes at time $T = 10$ days.

Scheme	h	Δt	Test 1		Test 2	
			L^2 -error Ψ_h	L^2 -error S_h	L^2 -error Ψ_h	L^2 -error S_h
SBDF	1.9	1e-2	2.47e-1	1.04e-2	2.96e-1	1.18e-2
	0.8	5e-3	7.67e-2	3.09e-3	8.41e-2	3.42e-3
	0.4	2.5e-3	2.34e-2	9.32e-4	2.67e-2	1.08e-3
MSBDF	1.9	1e-2	2.48e-1	1.04e-2	2.96e-1	1.18e-2
	0.8	5e-3	7.72e-2	3.11e-3	8.47e-2	3.45e-3
	0.4	2.5e-3	2.37e-2	9.44e-4	2.71e-2	1.10e-3
Richards-AB	1.9	1e-2	2.51e-1	1.05e-2	2.95e-1	1.20e-2
	0.8	5e-3	8.28e-2	3.27e-3	9.37e-2	3.76e-3
	0.4	2.5e-3	2.93e-2	1.11e-3	3.73e-2	1.41e-3
Richards-MAB	1.9	1e-2	2.52e-1	1.06e-2	2.95e-1	1.20e-2
	0.8	5e-3	8.34e-2	3.29e-3	9.45e-2	3.79e-3
	0.4	2.5e-3	2.95e-2	1.11e-3	3.78e-2	1.42e-3
Richards-M1	1.9	1e-2	2.48e-1	1.04e-2	2.96e-1	1.18e-2
	0.8	5e-3	7.68e-2	3.09e-3	8.46e-2	3.44e-3
	0.4	2.5e-3	2.36e-2	9.37e-4	2.71e-2	1.10e-3
Richards-M2	1.9	1e-2	2.46e-1	1.03e-2	2.95e-1	1.18e-2
	0.8	5e-3	7.49e-2	3.01e-3	8.21e-2	3.32e-3
	0.4	2.5e-3	2.18e-2	8.67e-4	2.41e-2	9.81e-4

and the exact solution. The results indicate that all numerical schemes yield accurate results in terms of mass conservation.

2.5.1.2 Manufactured solution

In the previous test case, we assessed the accuracy and efficiency of the proposed numerical schemes for the Gardner model. In this model, the soil-water diffusivity, defined as the ratio of the hydraulic conductivity K to the specific moisture capacity C , and the derivative of the unsaturated hydraulic conductivity with respect to saturation are considered constant. Exact solutions for realistic dependence of these variables have been previously established (e.g., [Sander *et al.* \(1988\)](#)). In this section, we extend our analysis to more realistic soils using the method of manufactured solutions ([Keita *et al.*, 2021b](#)). We consider a two-dimensional soil domain of shape $[0, l] \times [0, L]$, and the final time of simulation is set to $T = 120$ seconds. The manufactured solution for the

Scheme	$\tilde{\alpha} = 0.164$		$\tilde{\alpha} = 0.328$	
	L^2 -error Ψ_h	L^2 -error S_h	L^2 -error Ψ_h	L^2 -error S_h
SBDF	7.67e-2	3.09e-3	1.08e+0	3.05e-2
MSBDF	7.72e-2	3.11e-3	1.08e+0	3.06e-2
Richards-AB	8.28e-2	3.27e-3	1.08e+0	3.06e-2
Richards-MAB	8.34e-2	3.29e-3	1.08e+0	3.06e-2
Richards-M1	7.68e-2	3.09e-3	1.08e+0	3.05e-2
Richards-M2	7.49e-2	3.01e-3	unstable	

Table 2.4 The computed pressure head and effective saturation L^2 -errors for the semi-implicit (S, Ψ) -schemes in function of the soil parameter $\tilde{\alpha}$ at time $T = 10$ days with $\Delta t = 5 \times 10^{-3}$ day and 50×50 elements in the first test scenario.

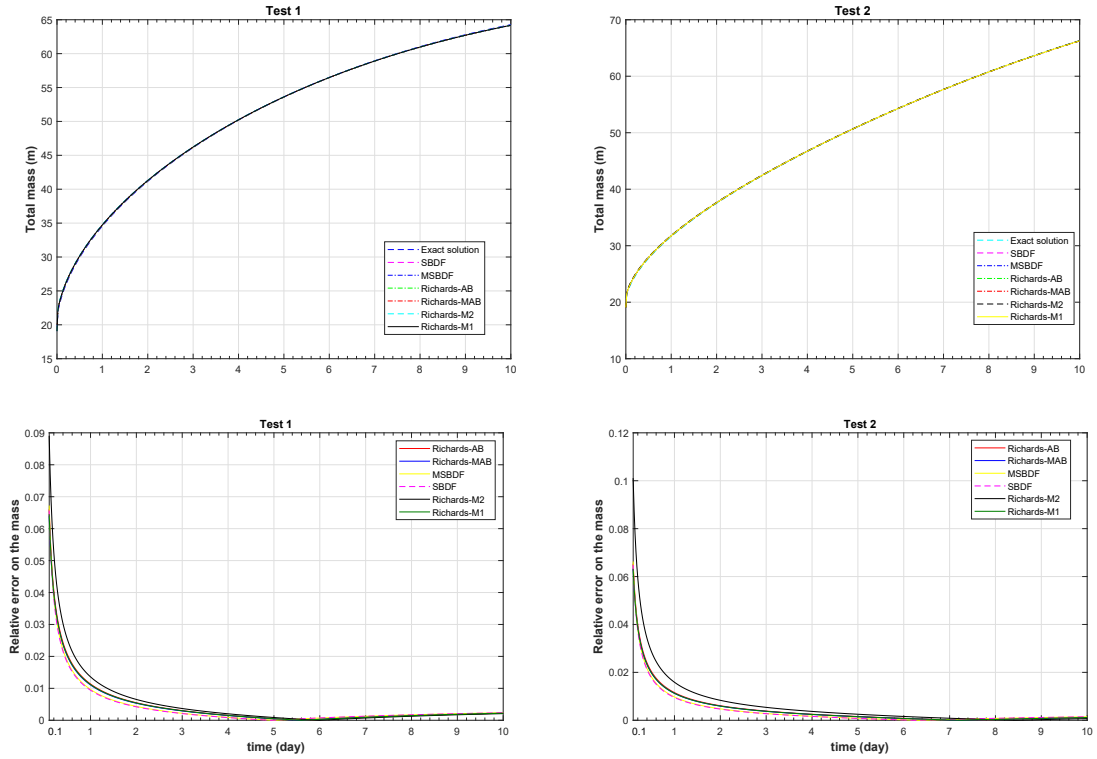


Figure 2.9 Time-evolution and the relative error of the total mass in the computational domain: Exact and numerical schemes with $\Delta t = 10^{-2}$ day and 25×25 elements.

pressure head Ψ is given by:

$$\Psi(x, z, t) = 20.4 \tanh \left(0.5 \left(z + \frac{t}{12} - 15 \right) \right) - 41.1. \quad (2.48)$$

The constitutive relationships are based on the Haverkamp model to account for the nonlinear dependency of the diffusivity function on saturation. The complete expressions for the Haverkamp model are provided in Table 2.2. The pressure head function (2.48) is substituted into the Richards equation (2.1) to derive a source term, which is then added to the equation. This derived source term is independent of the water saturation, eliminating the need for additional numerical treatment. The parameters for this numerical test are as follows:

$$l = 2 \text{ cm}, L = 20 \text{ cm}, K_s = 9.44 \times 10^{-3} \text{ cm/seconds}, \tilde{\alpha} = 0.0271 \text{ cm}^{-1}, \quad (2.49)$$

$$\theta_s = 0.287, \theta_r = 0.075, \beta_0 = 3.96, \gamma = 4.74, A = 0.0524 \text{ cm}^{-1}.$$

For numerical simulations, the Richards equation is supplemented with initial and Dirichlet boundary conditions derived from the exact solution (2.48). We employ the SBDF scheme with a spatial mesh size of $h = 0.1 \text{ cm}$ and a time step of $\Delta t = 0.01 \text{ s}$. It is worth noting that no regularization of the derivative of the Leverett function is required in this numerical test, as the saturation levels remain well below full saturation throughout the entire simulation period. The resulting profiles for pressure head and saturation at various time snapshots are presented in Figure 2.10. The SBDF scheme provides excellent agreement with the exact solution for both variables. To perform a temporal convergence analysis, we fix a spatial mesh size of $h = 0.1 \text{ cm}$ and use four different time steps $\Delta t = 4, 2, 1, 0.5 \text{ seconds}$. The modified L^2 -error defined in (2.47) is used to compute the error between the numerical and exact solutions. Figure 2.10 (left) presents the results obtained for the SBDF and Richards-M2 schemes, with the convergence order reported for all time steps used. The Richards-M2 scheme demonstrates an approximate convergence order of 1.85 for both Ψ_h and S_h across all time steps. Similarly, the SBDF, MSBDF, and Richards-M1 schemes exhibit nearly the same convergence order for both variables, approximately 1.83. In contrast, the Richards-AB and Richards-MAB schemes show the lowest convergence order, around 1.56 for both variables. Additionally, the spatial convergence order is evaluated for the proposed schemes. For this purpose, we use a fixed time step of $\Delta t = 0.01 \text{ second}$ and four different spatial mesh sizes $h = 4, 2, 1, 0.5 \text{ cm}$. All schemes demonstrate a convergence order of approximately 1.9 in the L^2 -norm and around 1 in the H^1 -norm. The results are presented for both variables Ψ_h and S_h .

This numerical experiment, utilizing realistic capillary pressure models, validates the effectiveness of the proposed methods for solving the Richards equation, particularly the SBDF and Richards-M2 schemes. The results demonstrate that these techniques can accurately and efficiently simulate water flow in unsaturated soils. We now turn our attention to a more challenging problem involving a heterogeneous soil configuration.

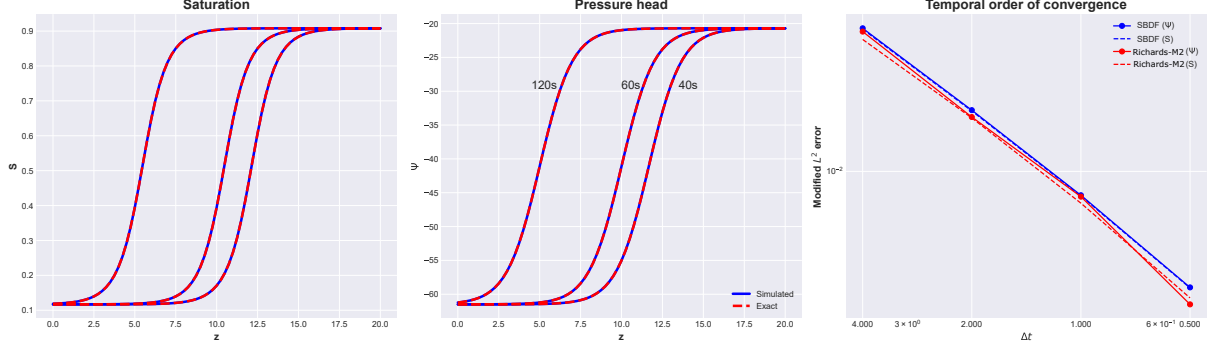


Figure 2.10 (Left) Evolution of saturation over time. (Middle) Evolution of pressure head over time. (Right) Modified L^2 -error as a function of time step Δt . Simulations were performed using a time step of $\Delta t = 0.01$ second and a mesh size of $h = 0.1$ cm.

2.5.1.3 2D heterogeneous soil

Here, we will perform a challenging test that involves a curvilinearly layered soil based on the van Genuchten-Mualem water retention model (Manzini and Ferraris, 2004). Our aim in this test is to investigate the capability of the generalized semi-implicit (S, ψ) -schemes to handle more complicated geometries and heterogeneous conditions, as was done in the case of the SBDF scheme (Keita *et al.*, 2021b). The geometry of the domain and the parameters used in this test are shown in Figure 2.11 and Table 2.5 respectively. The consecutive relationships in the van Genuchten-Mualem model are presented in Table 2.2. The unsaturated hydraulic conductivity $K(\Psi)$ is described using Mualem's model (Mualem, 1976). The two layers of the soil are separated by the curve defined by the function:

$$\xi(x) = 100 (0.1(1 - \cos(\pi x/100)) + 0.45) \quad \text{for } x \in [0, 100 \text{ cm}]. \quad (2.50)$$

The initial and boundary conditions used for this heterogeneous test are illustrated in Figure 2.11. We used the time step $\Delta t = \frac{1}{60}$ hour and $h = 1$ cm. All numerical schemes perform well and show good agreements with those shown in (Keita *et al.*, 2021b; Manzini and Ferraris, 2004). For instance, Figure 2.12 presents the simulation results using the SBDF scheme at different time snapshots.

To examine the order of convergence for the studied semi-implicit (S, Ψ) -schemes in

	K_s (cm. h ⁻¹)	θ_s	θ_r	$\tilde{\alpha}$ (cm ⁻¹)	n_v
$z \geq \xi(x)$	0.25	0.50	0.12	0.028	3.00
$z < \xi(x)$	2	0.46	0.034	0.016	1.37

Table 2.5 Parameters for the soil textures in the curvilinearly layered soil test case.

this numerical solution test, a reference solution is computed using the SBDF scheme

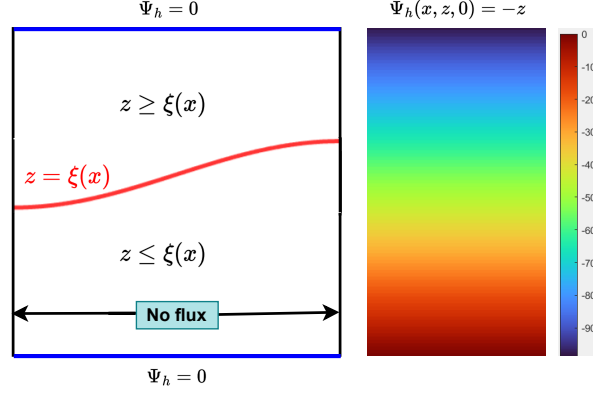


Figure 2.11 2D curvilinearly layered soil: $[0, 100 \text{ cm}]^2$ - initial and boundary conditions.

Scheme	Ψ_h	S_h
SBDF	1.77	1.8
MSBDF	1.75	1.79
Richards-M2	1.78	1.801

Table 2.6 Order of convergence for the semi-implicit (S, Ψ) -schemes in the curvilinearly layered soil test case.

with a finer mesh size $h = 0.25 \text{ cm}$, and $\Delta t = \frac{1}{240}$ day. The L^2 -errors for the semi-implicit (S, Ψ) -schemes are depicted in Figure 2.13. The three numerical schemes exhibit almost the same errors in both pressure head and saturation variables. The order of convergence for the proposed schemes for both variables is presented in Table 2.6. The results suggest that the convergence order for both Ψ_h and S_h is approximately 2, based on the two smallest mesh sizes, $h = 0.837$ and $h = 0.418$. To assess the CPU time efficiency of the semi-implicit (S, Ψ) -schemes, we conduct simulations for this heterogeneous test infiltration up to a time $T = 6$ hours. We used the same parameters as mentioned earlier, with a mesh size of $h = 2.4 \text{ cm}$ and a time step of $\Delta t = \frac{1}{60}$ hour. SBDF scheme stands out with the shortest CPU time at 136.37 seconds, whereas Richards-M2 and MSBDF require 154.09 and 136.69 seconds, respectively, to simulate the 6 hours of water infiltration into the heterogeneous layered soil.

2.5.2 Water flow and solute transport

In this section, our aim is to evaluate the proposed coupled semi-implicit schemes for modeling water flow and solute transport in unsaturated soil using different benchmarks. A convergence and stability analysis of the coupled semi-implicit schemes is carried out in the following section to determine the most efficient and accurate scheme. The selected scheme is then used in the remaining sections, which focus on numerical

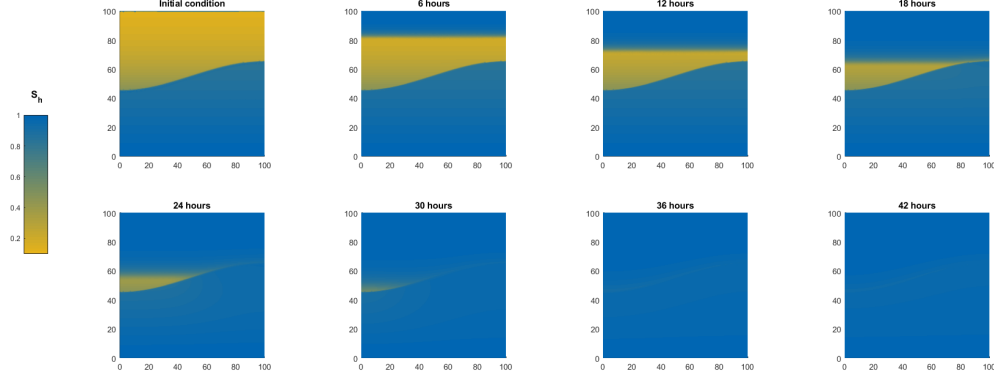


Figure 2.12 Saturation profile evolution using the SBDF scheme at various times.

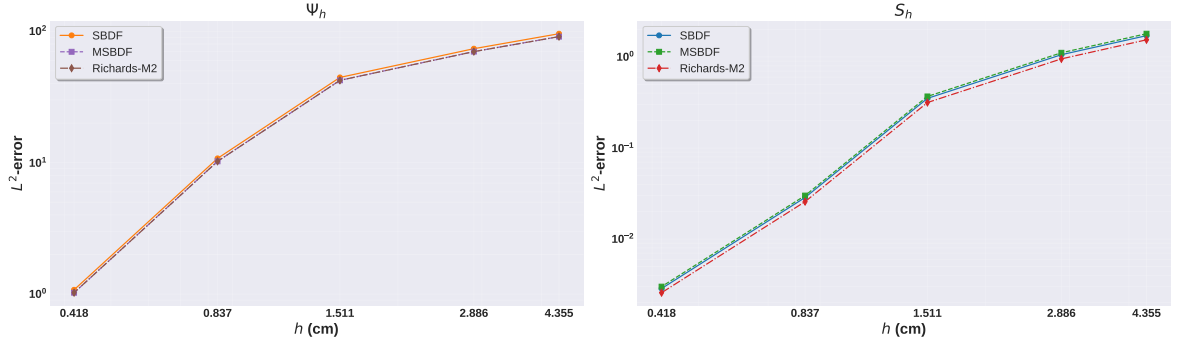


Figure 2.13 Pressure head and saturation L^2 -errors in function of the mesh size h for the studied semi-implicit (S, Ψ) -schemes in the curvilinearly layered soil test case.

tests to assess various scenarios, including the transport of single and multispecies in one and two dimensions.

2.5.2.1 Convergence and stability analysis of the coupled semi-implicit schemes

In this section, we aim to analyze the accuracy and stability of the proposed coupled schemes for modeling water-solute transport in unsaturated soils. Analytical solutions for the coupled water-solute system are scarce and assume simple boundary conditions, and only for one-dimensional settings. Additionally, the majority of these solutions suppose steady-state conditions where water flux and water content do not vary with time. Thus, we require an exact solution with unsteady-state water flux as well as nonlinear dependency between the saturation and soil-water diffusivity for realistic soils. In this paper, we opt for the exact solution developed by Sander et al. (Sander and Braddock, 2005) for the coupled model of water flow and solute transport. The exact solution is derived by neglecting the gravity term, and thus water moves only by

capillary forces. The Richards equation in terms of the water content-based form is given as follows:

$$\begin{cases} \frac{\partial \theta}{\partial t} + \frac{\partial q}{\partial x} = 0, \\ q = -D(\theta) \frac{\partial \theta}{\partial x}, \end{cases} \quad (2.51)$$

where $D(\theta)$ is the soil-water diffusivity, which is given as:

$$D(\theta) = \frac{\lambda w_f^2}{2} S^\lambda \left(1 - \frac{S^\lambda}{1 + \lambda} \right). \quad (2.52)$$

We recall that S is the saturation given by equation (2.2). In equation (2.52), λ is a soil parameter that should be larger than 3, and w_f is the water-wetting front position. The Richards equation in its water-content form (2.51) is coupled to the following solute transport equation:

$$\begin{cases} \frac{\partial \theta c}{\partial t} + \frac{\partial q_c}{\partial x} = 0, \\ q_c = -\theta D_s \frac{\partial c}{\partial x} + q c, \end{cases} \quad (2.53)$$

where q_c is the solute flux, and D_s is the dispersion function for the solute which is given by:

$$\theta D_s = \theta D_w + |q| D_L, \quad (2.54)$$

with D_w being the molecular diffusion constant, and D_L is the dispersivity. The dispersivity is assumed to be a function of scale and time and parametrized as follows:

$$D_L(x, t) = \varepsilon_0 \phi^w \sqrt{t}, \quad (2.55)$$

where $\phi = x/\sqrt{t}$ is a Boltzmann transformation, w and ε_0 are parameterized constants. This dispersivity function is different from the longitudinal dispersivity (2.11). This one has an explicit dependency on both time and space.

The coupled water-solute system is subjected to the initial and boundary conditions presented in Figure 2.14. Due to neglecting the gravity term $\frac{\partial K}{\partial z}$ in the Richards equation, the flow is horizontal in the x -axis direction. We assume an initial water content and solute concentration equal to θ_r and c_r , respectively. At the soil surface $x = 0$, the water and solute are imposed as Dirichlet boundary conditions with $\theta = \theta_s$ and $c = c_s$. Both water and solute are assumed to equal their initial condition at the position $x = 1$. The exact solution for the coupled model of water and solute subjected to the previously mentioned initial and boundary conditions is given in reference (Sander and Braddock, 2005).

To assess the order of convergence of the proposed coupled semi-implicit schemes, we discretized the soil column into 101 elements and used different time steps Δt . The

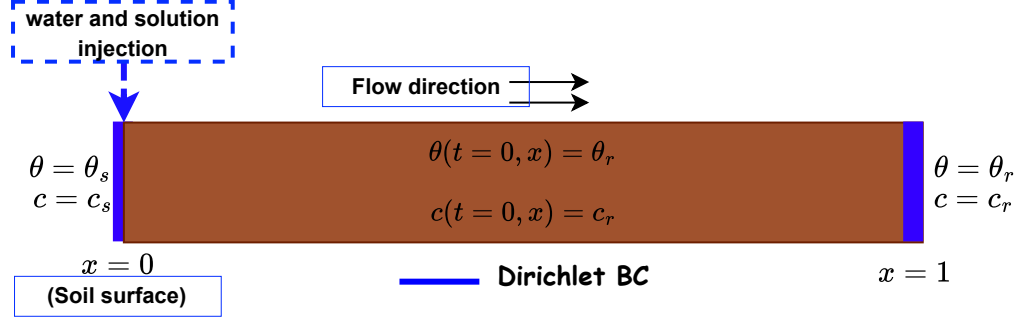


Figure 2.14 Horizontal infiltration test domain.

critical parameter for this test is ε_0 , and as it tends to zero, the problem becomes an advection-dominated transport phenomenon with a wave speed of q/θ . This is due to the dispersion term D_s in equation (2.54) approaching zero. Thus, for small values of ε_0 , the solute profile will be steeper. In our simulation, we fixed the value of ε_0 at 0.1. The other parameters were set as follows:

$$\theta_r = 0.1, \theta_s = 0.2, c_r = 0, c_s = 0.1, w_f = 1, \lambda = 5, D_w = 0, w = 1. \quad (2.56)$$

The results obtained from the SBDF scheme for both saturation and solute concentration are given in Figure 2.15 at the final simulation time $T = 1$ for different time steps. We can observe the convergence of the SBDF scheme as Δt decreases. An order of convergence of approximately 1.78 is obtained for this test case. The SBDF scheme provides stable profiles even with larger time steps, but the profiles do not match the exact solutions for both water and solute when using a larger time step. Good agreement, with a relative L^2 -error of less than 1%, is obtained between the SBDF scheme and the exact saturation profile when using $\Delta t = 10^{-3}$. However, we need $\Delta t = 10^{-4}$ to achieve an acceptable relative L^2 -error for the solute concentration solution. If the time step is fixed at $\Delta t = 10^{-4}$, the SBDF scheme can achieve a relative L^2 -error of less than 0.5% for saturation and 0.41% for solute concentration.

During the numerical simulations, it was observed that the SBDF scheme allows for larger time steps if only water infiltration is considered. However, when coupled to the solute transport equation, the SBDF scheme becomes more restrictive in terms of time step which should be chosen to obtain stable solutions.

To assess the influence of the free parameters (α, β) on the accuracy and stability of the proposed coupled semi-implicit schemes, we solve the same numerical test with the same parameters provided in (2.56), and the results obtained are shown in Figure 2.16. No significant difference is observed between the schemes for saturation. However, the difference becomes clear for the solute concentration profiles, where the accuracy decreases as α tends to 0. The family of schemes with $\alpha = 1$ provides the best choice

regarding the value of the free parameter β . This demonstrates the efficiency of the SBDF and MSBDF schemes for modeling this coupled system. Regarding stability, the simulation results show that when $\alpha = 0$, the coupled scheme is unstable regardless of the chosen value of β . This is expected due to the fact that the scheme becomes fully explicit. As α approaches 1, the scheme becomes more efficient in terms of both accuracy and stability, and the optimal choice is the SBDF scheme with $\alpha = 1$.

In summary, this numerical test has shown that the choice of free parameters (α, β) plays a crucial role in the accuracy and stability of the coupled semi-implicit schemes for modeling water-solute transport in porous media. The SBDF scheme, with $\alpha = 1$, has proven to be the most efficient and accurate for modeling the coupled water-solute transport system. Therefore, in the subsequent sections, we focus on the integrated model of water flow and solute transport using the SBDF numerical scheme.

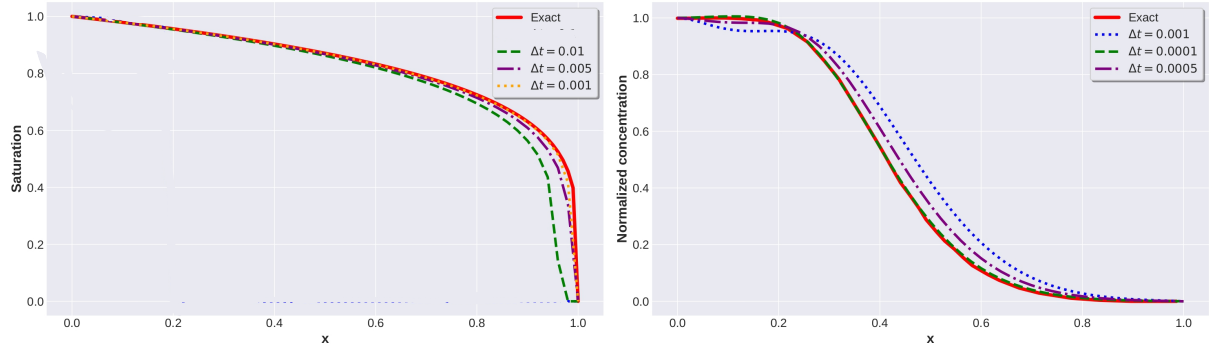


Figure 2.15 Saturation (left) and normalized concentration profiles (right) for different time steps Δt . The final simulation time is $T = 1$ and the mesh size is $h = 0.01$. The normalized concentration is given by: $c = (c - c_r)/(c_s - c_r)$.

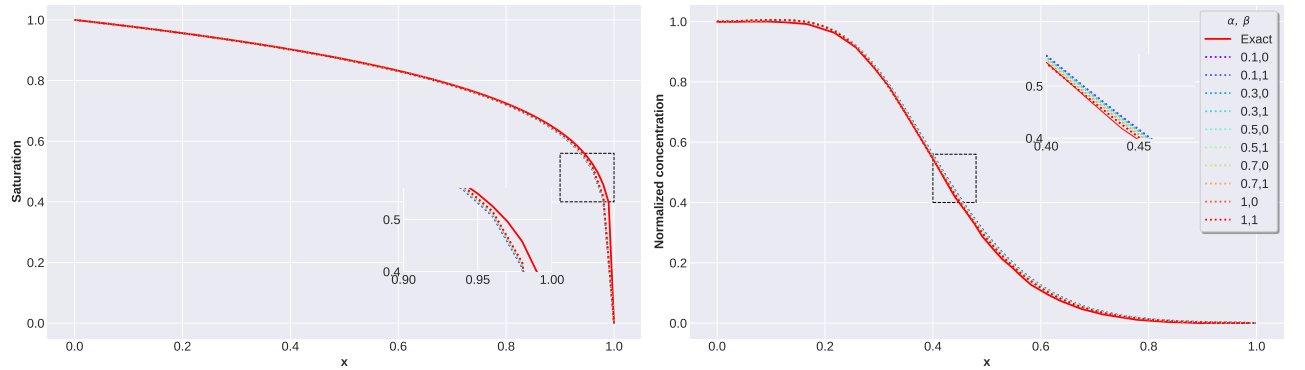


Figure 2.16 Saturation and normalized concentration profiles for different α and β values. The final simulation time is $T = 1$ and the mesh size is $h = 0.01$.

2.5.2.2 1D single-species solute transport

In this section, we solve the coupled water and solute transport problem in a one-dimensional scenario and then extend it to a two-dimensional configuration. To assess our results, we used the HYDRUS-1D software, which is well-known in the field and serves as our point of comparison ([Šimůnek et al., 2005](#)). The simulation domain is a rectangle $[0, L]^2$, and the coupled model is supplemented by specific initial and boundary conditions, outlined as follows:

$$\text{Water flow : } \begin{cases} \Psi(x, z, 0) = -1 \text{ m} & \text{for } (x, z) \in [0, L]^2, \\ \Psi(x, 0, t) = -1 \text{ m} & \text{for } (t, z) \in (0, T] \times [0, L], \\ \Psi(x, L, t) = -0.2 \text{ m} & \text{for } (t, x) \in (0, T] \times [0, L], \end{cases} \quad (2.57)$$

$$\text{Solute transport : } \begin{cases} c(x, z, 0) = 0 \text{ mmol/m}^3 & \text{for } (x, z) \in [0, L]^2, \\ c(x, 0, t) = 0 \text{ mmol/m}^3 & \text{for } (t, x) \in (0, T] \times [0, L], \\ c(x, L, t) = 1 \text{ mmol/m}^3 & \text{for } (t, x) \in (0, T] \times [0, L], \end{cases} \quad (2.58)$$

and no flux boundary condition is imposed on the other sides of the domain. The van Genuchten-Mualem model for capillary pressure is used in this test, and the related functions are given in Table 2.2 with the following data for clay soil:

$$L = 1 \text{ m}, K_s = 0.048 \text{ m/day}, \tilde{\alpha} = 0.8 \text{ m}^{-1}, \theta_s = 0.38, \theta_r = 0.068, \quad (2.59)$$

$$D_L = 0.1 \text{ m}, D_w = 0 \text{ m}^2/\text{day}, \Delta t = 5 \times 10^{-4} \text{ day}, h = 10^{-2} \text{ m}, n_v = 1.09.$$

In this test, we assume that no adsorption occurs between the solid phase of soil and the solute ($R = 1$). The solute concentration and pressure head profiles, along with their cross-sections, are depicted in Figure 2.17. The results from the proposed schemes and HYDRUS-1D exhibit significant similarity. To evaluate mass conservation for water and solute, we employed the formulas (2.42) and (2.43) to calculate the numerical mass and compared it to the reference solution, which is determined using HYDRUS-1D. Figure 2.18 presents the results, demonstrating a good agreement between the numerical and reference solutions.

This numerical test confirms the accuracy of the proposed schemes for single-species solute transport. In the following subsection, we investigate the performance of the numerical schemes in the context of multiple solute transport.

2.5.2.3 1D multispecies nitrogen transport

In this 1D numerical test, a fully coupled system modeling a first-order decay chain of nitrogen species is selected to examine the efficacy of the SBDF scheme in addressing this scenario. This numerical test involves multiple solute concentrations, with chemical reactions in their governing partial differential equations for solute transport. The

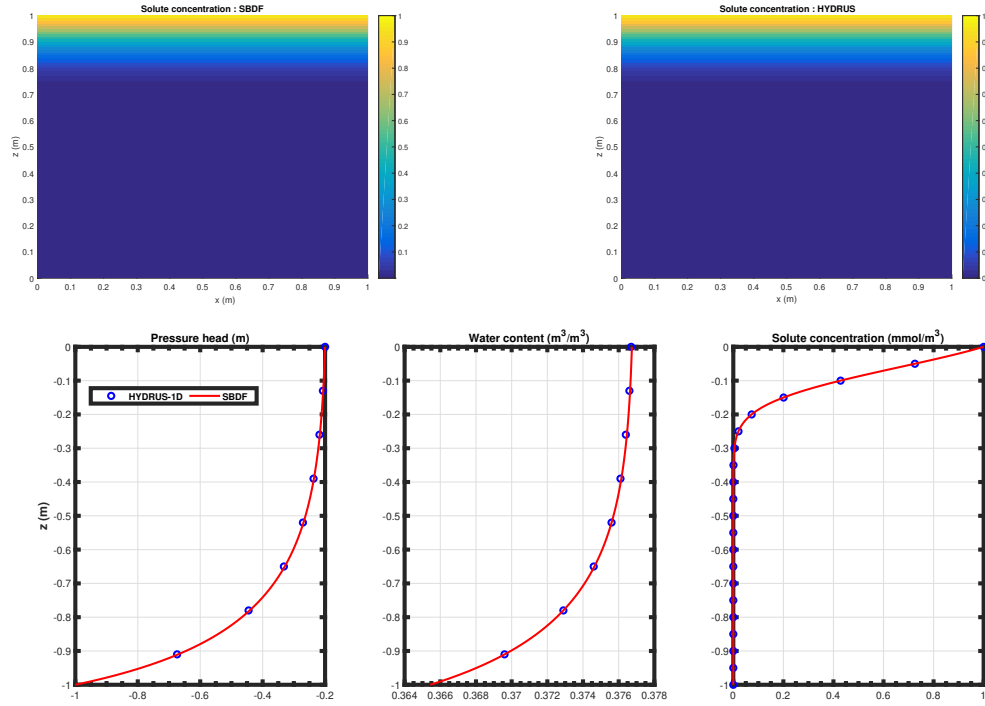


Figure 2.17 Concentration profiles for the SBDF scheme and HYDRUS-1D at $T = 1$ day (top) and cross-section at $x = 0.5$ m for pressure head, water content, and solute concentration at $T = 10$ days (down).

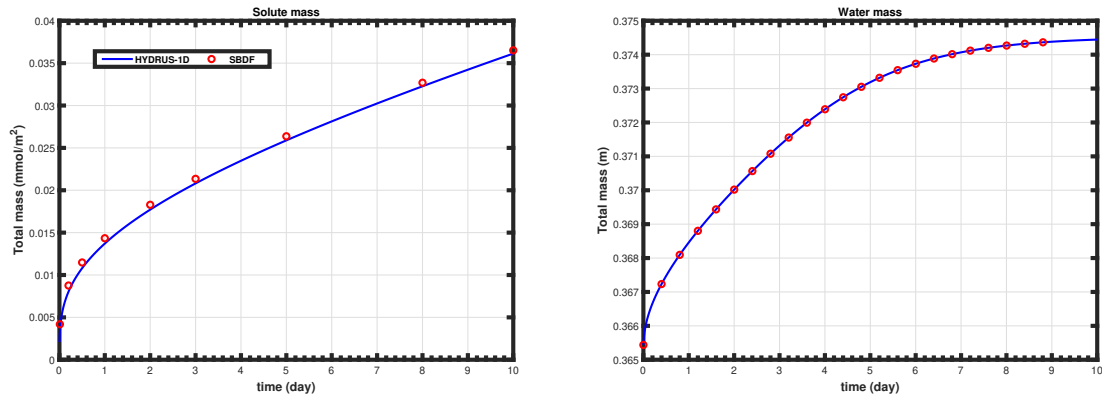


Figure 2.18 Time-evolution of the total mass of water and solute obtained by the SBDF scheme and HYDRUS-1D.

comparison is made using the HYDRUS-1D software. In the water flow equation, the initial and bottom boundary pressure head values are set to -1000 cm, while the soil surface pressure head value is set to -75 cm. The initial sources of nitrogen are assumed

to be ammonium NH_4^+ and nitrate NO_3^- . The ammonium is nitrified by autotrophic bacteria to form nitrite NO_2^- and subsequently nitrate NO_3^- , which is denitrified to produce dissolved nitrogen N_2 and N_2O (see Figure 2.19 left column). The partial differential equations governing the concentrations of nitrogen multispecies (NH_4^+ , NO_2^- and NO_3^-) are (Hanson *et al.*, 2006; Kaluarachchi and Parker, 1988; Šimůnek *et al.*, 2005):

$$\begin{cases} \frac{\partial R\theta c_1}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c_1 - c_1 \mathbf{q}) - \mu_1 \theta c_1, \\ \frac{\partial \theta c_2}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c_2 - c_2 \mathbf{q}) + \mu_1 \theta c_1 - \mu_2 \theta c_2, \\ \frac{\partial \theta c_3}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c_3 - c_3 \mathbf{q}) + \mu_2 \theta c_2 - \mu_3 \theta c_3, \end{cases} \quad (2.60)$$

where c_1 , c_2 , and c_3 denote the concentrations of NH_4^+ , NO_2^- , and NO_3^- respectively, and μ_1 , μ_2 , and μ_3 represent the first-order rate coefficients for the nitrification of NH_4^+ , nitrification of NO_2^- , and denitrification of NO_3^- .

In this model, we exclusively considered first-order degradation reactions (nitrification, denitrification, and adsorption). Only ammonium was assumed to be linearly adsorbed to the solid phase of soil, which is represented by the retardation factor R . Certainly, additional terms can be incorporated into the model as needed. The soil was assumed to be initially free from any form of nitrogen species. The concentrations at the soil surface were fixed at 50, 0, and 10 mmol/cm³ for ammonium, nitrite, and nitrate, respectively. Zero-concentration gradients were assumed on the bottom boundary for all nitrogen species. The van Genuchten-Mualem model for capillary pressure is employed in this test with the following parameters:

$$L = 100 \text{ cm}, K_s = 796.608 \text{ cm/day}, \tilde{\alpha} = 0.0335 \text{ cm}^{-1}, \theta_s = 0.381, \theta_r = 0.102,$$

$$D_L = 4 \text{ cm}, D_w = 0.288 \text{ cm}^2/\text{day}, \Delta t = 10^{-4} \text{ day}, h = 1 \text{ cm}, n_v = 2, p = 1, \quad (2.61)$$

$$\rho = 1.6 \times 10^6 \text{ mg/l}, K_d = 0.34 \times 10^{-6} \text{ l/mg}, \mu_1 = 0.12 \text{ day}^{-1}, \mu_2 = 0.048 \text{ day}^{-1}, \mu_3 = 0.012 \text{ day}^{-1},$$

where L is the depth of the soil profile.

The nitrogen kinetics parameters μ_i were obtained from (Bailey *et al.*, 2013), while other parameters were sourced from (Berlin and Suresh Kumar, 2018). Figure 2.19 (right column) displays the simulated concentrations of nitrogen species after 1 day, showing excellent agreement with HYDRUS-1D. The proposed schemes were tested using different model parameters. The numerical schemes demonstrated their capability to capture the effects of the nitrogen kinetics parameters μ_i on the fate of each species (results not shown).

2.5.2.4 2D water flow and pore-water electrical conductivity

In this section, we solve the coupled water flow and solute transport equations employing the SBDF scheme. This test scenario was previously employed in the study

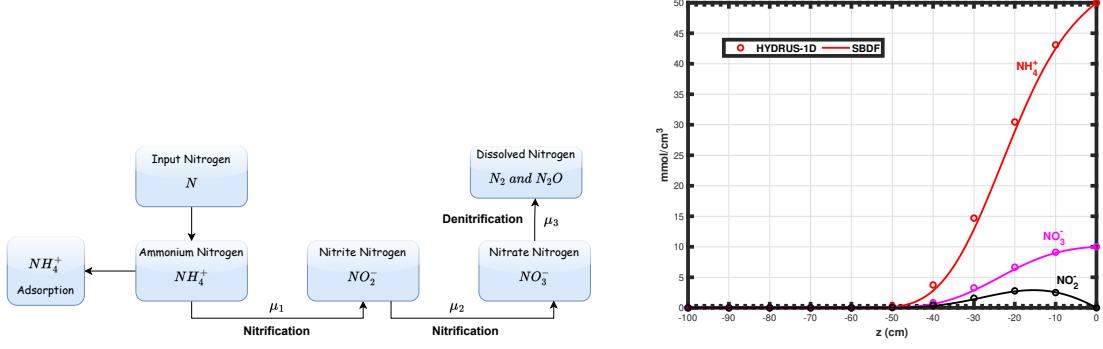


Figure 2.19 Diagram illustrating the first-order decay chain of nitrogen species (left) and nitrogen species concentrations at time $T = 1$ day (right).

conducted by (Haruzi and Moreno, 2023c) to simulate two-dimensional water flow and pore-water electrical conductivity (pore-water EC, denoted as σ) during infiltration and redistribution processes. In their study (Haruzi and Moreno, 2023c), the authors solved the coupled model through both physics-informed neural networks and numerical simulation. We utilize the dataset (Haruzi and Moreno, 2023b), which includes their obtained results, to conduct a comparative analysis with our findings. The domain's geometry, along with the boundary conditions applied, is visualized in Figure 2.20 (left column). The water table is situated far below the domain of interest. Consequently, a free drainage boundary condition was implemented at the bottom, expressed as $\frac{\partial \Psi_h}{\partial z} = 0$. As a result, the Neumann boundary condition, q_N , implicitly depends on the pressure head solution Ψ_h through the following relationship:

$$q_N(\Psi_h^{n+1}) = K(\Psi_h^{n+1}). \quad (2.62)$$

To achieve the primary research goal of avoiding the need for an iterative procedure, the extrapolation method (2.18) will also be applied to equation (2.62).

Water infiltrates the domain from a center-located section with a width of 10 cm at a constant flow rate of $q_z = -0.1$ m/day. In the same section, a third-type boundary condition (5.11) is applied, with a constant solute feeding of $\sigma_0 = 1$ S/m. A zero-flux boundary condition was assumed for the remaining domain boundaries. No adsorption is considered between the pore-water EC and the solid phase of soil ($R = 1$). The infiltration occurred for a duration of 1 day. The parameters and initial conditions for water flow and the transport of pore-water EC are as follows:

$$\Psi(x, z, 0) = -1.3 \text{ m}, \sigma(x, z, 0) = 0.1 \text{ S/m}, K_s = 0.25 \text{ m/day}, \tilde{\alpha} = 3.6 \text{ m}^{-1}, \quad (2.63)$$

$$\theta_s = 0.43, \theta_r = 0.078, n_v = 1.56, D_L = 0.5 \text{ m}, D_T = 0.1 \text{ m}.$$

The soil parameters characterize loamy soil, with the capillary pressure modeled using the van Genuchten-Mualem model.

For the numerical schemes, we employed a time step $\Delta t = 10^{-3}$ days and a mesh with

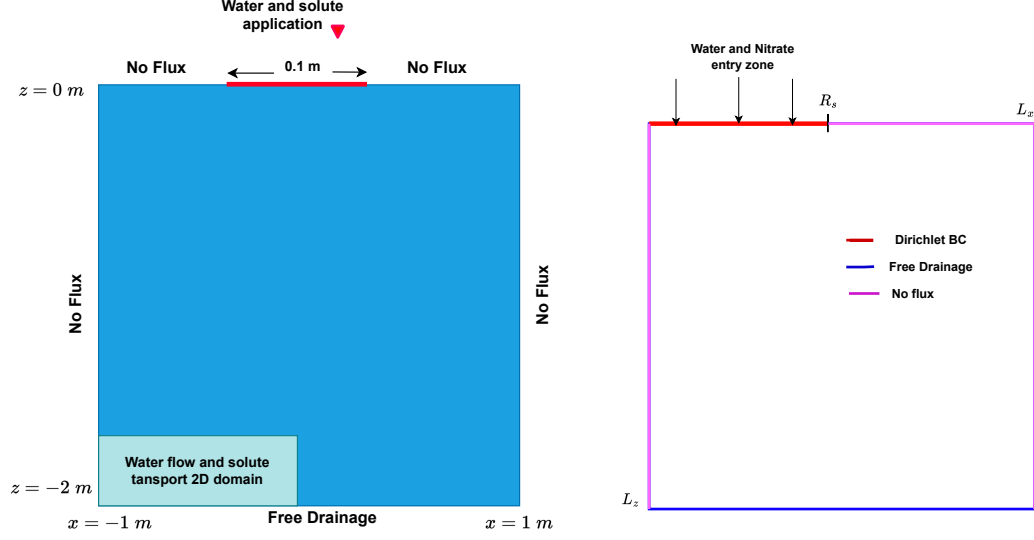


Figure 2.20 Water flow and solute transport 2D domain for pore-water EC test (left) and for nitrate test (right).

4947 nodes and 9640 triangular elements with a small mesh size near the soil surface. In the case of the numerical simulation for the reference solution used in (Haruzi and Moreno, 2023c), the authors utilized 40401 nodes and 80000 triangular elements. Figure 2.21 illustrates the pressure head and pore-water EC at various times. These results show good agreement with those presented in (Haruzi and Moreno, 2023c). At position $x = 0 \text{ m}$, cross-sections depicting both pressure head and pore-water EC as functions of soil depth are illustrated in Figure 2.22, which corresponds to the same profiles as those obtained in (Haruzi and Moreno, 2023c). Based on this numerical test, we conclude that the SBDF scheme remains stable and demonstrates good accuracy in solving the coupled processes of infiltration and transport within a two-dimensional geometry. The subsequent numerical test focuses on a real experiment involving infiltration and nitrate transport in 2D unsaturated soil.

2.5.2.5 Modeling 2D nitrate transport: Validation with experimental data

In this last numerical test, our aim is to evaluate the performance of the proposed scheme in simulating an experimental infiltration and transport scenario. This test is grounded in an experiment conducted in (Li *et al.*, 2005), where the authors measured water content and nitrate concentration in a laboratory experiment, comparing the

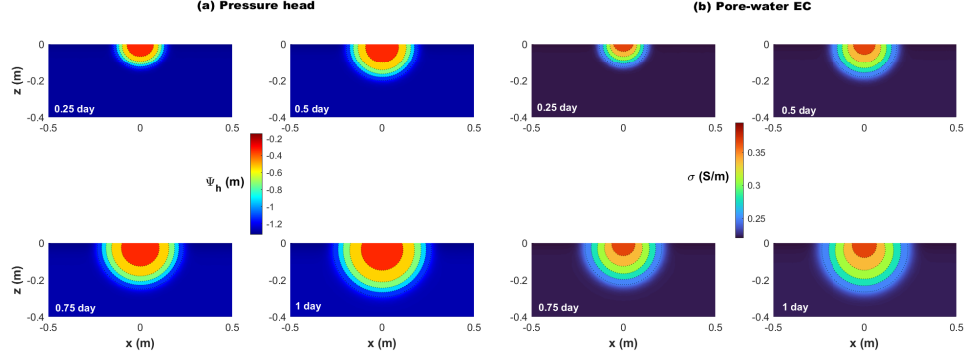


Figure 2.21 (a) Pressure head and (b) Pore-water EC profiles at various times obtained using the SBDF scheme.

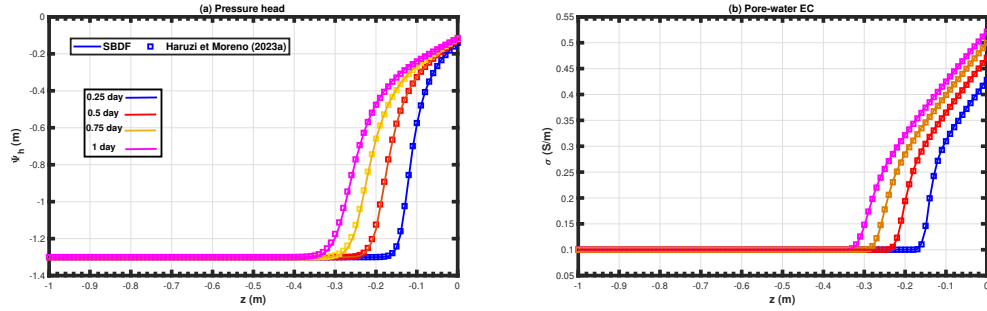


Figure 2.22 (a) Pressure head and (b) Pore-water EC vertical profiles ($x = 0$ m) as functions of soil depth at various times: SBDF scheme results and comparison with Haruzi and Moreno data (Haruzi and Moreno, 2023c).

results with a numerical simulation. The experiment was conducted on a cylindrical domain with a radius $L_x = 41$ cm and depth $L_z = 40$ cm. Cylindrical coordinates with axisymmetrical flow were used to model water flow and nitrate transport. Here, the radial coordinate is denoted by the variable x . Water infiltrated with a constant pressure head Ψ_0 into the soil $[0, L_x] \times [0, L_z]$ through the left portion of the surface. This section was considered time-independent and had a length of R_s . In their experiment (Li *et al.*, 2005), the authors considered two nitrogen species: ammonium (NH_4^+) and nitrate (NO_3^-), both sourced from an ammonium nitrate fertilizer. The authors focused on the simulations and measurements of nitrate. The ammonium nitrate fertilizer was applied in the same location where water was introduced. Figure 2.20 (right column) depicts the 2D domain utilized in this experiment, along with the boundary conditions. No adsorption between the solid phase of soil and nitrate is considered. The van Genuchten-Mualem model was used for capillary pressure. In our numerical

test, to solve this problem, an unstructured mesh is adopted, with finer mesh elements near the region where water and solute were applied. The total number of triangles used in this test was 3126, and there were 1639 nodes with a mesh size of 1.86 cm. We employed a time step of $\Delta t = 10^{-4}$ hours, along with the following data:

$$\begin{aligned}\theta(x, z, 0) &= 0.13, \theta_s = 0.41, \theta_r = 0.047, n_v = 1.48, \tilde{\alpha} = 0.015 \text{ cm}^{-1}, \\ K_s &= 1.96 \text{ cm/h}, D_L = 0.32 \text{ cm}, D_T = D_L/100 \text{ cm}, D_w = 0.09 \text{ cm}^2/\text{h}, \\ L_x &= 41 \text{ cm}, L_z = 40 \text{ cm}, R_s = 5.6 \text{ cm}, \Psi_0 = 0.15 \text{ cm}.\end{aligned}\tag{2.64}$$

Figure 2.23 illustrates the cross-section of water content and nitrate concentration as a function of depth, with x fixed at 2.5 cm and 12.5 cm. The simulation involves continuous water and nitrate applications for 8 hours, starting with an initial nitrate concentration of 247.5 mg/L. The proposed schemes exhibit a good correspondence with the experimental data. Any observed deviations could potentially arise from water loss during the soil sampling process, as the water content was measured using the gravimetric method, as reported by the authors in (Li *et al.*, 2005). To evaluate nitrate transport, three water and fertilizer application strategies were employed in their study (Li *et al.*, 2005), as depicted in Figure 2.24. A Dirichlet boundary condition is used with a constant nitrate concentration of $c_0 = 54$ mg/L. The soil is supposed to be free from nitrate, and the duration of nitrate fertigation is the same across the three strategies, lasting 3 hours, with continuous water application for 6 hours. We employ the SBDF scheme to perform numerical simulations, investigating the evolution of nitrate under the different fertigation strategies. Figures 2.25 and 2.26 depict cross-sections of nitrate distribution and snapshots of water and nitrate in the 2D domain. As observed in (Li *et al.*, 2005), strategy A results in a profound depth for the peak of the nitrate profile, whereas strategy B retains more nitrate in the upper of the soil, which proves to be the most efficient strategy. In the scenario where the loam soil contains a non-negligible nitrate concentration, there is no noticeable distinction among the three strategies, consistent with the observations made in (Li *et al.*, 2005) (results not presented here). These findings collectively demonstrate the efficiency of the SBDF scheme for modeling water flow and nitrate transport in unsaturated soils.

2.6 Conclusion

This study presents several numerical time-stepping schemes for solving the coupled model of water flow and solute transport. Building upon the findings in (Keita *et al.*, 2021b), we developed a family of second-order accurate schemes for solving the Richards equation using a mixed finite element method for spatial discretization. Concurrently, the time-stepping schemes were extended for the transport equation, using the Galerkin finite element method in space. Numerical experiments for water flow reveal that SBDF, MSBDF, and Richards-M2 persist as robust schemes for homogeneous and het-

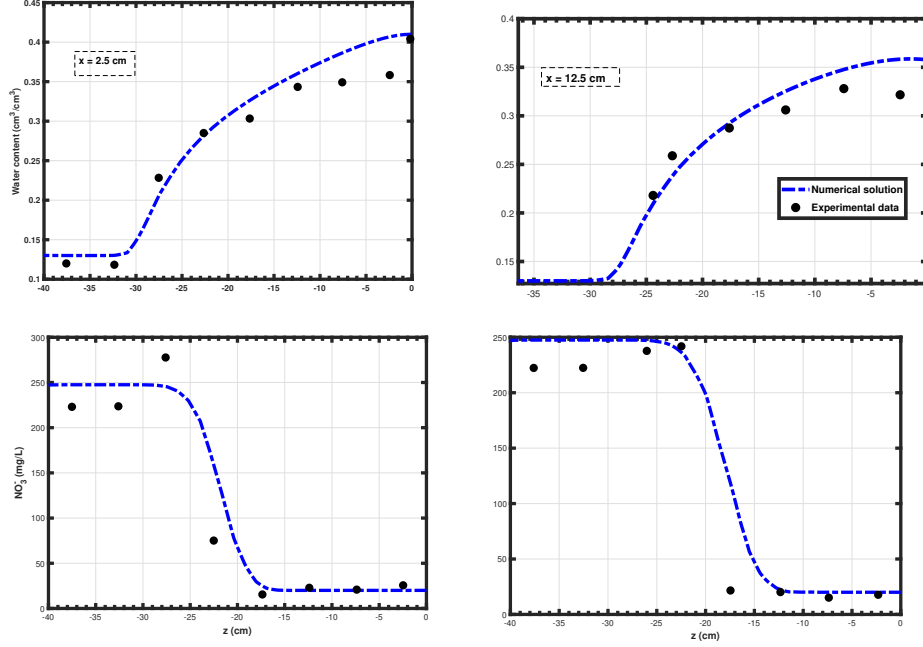


Figure 2.23 Cross-sections of water content (top) and nitrate concentration (down) at $x = 2.5$ cm and $x = 12.5$ cm after 8 hours of application.

erogeneous soils. SBDF has the advantage of computational efficiency and stability when gravity forces dominate over capillary forces. Through numerical analysis of the coupled semi-implicit schemes, our results affirm the SBDF scheme's superior robustness, establishing it as the optimal choice among the proposed numerical methods. Consequently, we used the SBDF scheme to solve the coupled water flow and solute transport model. Through various numerical tests, we simulated diverse scenarios in 1D and 2D configurations, including single and multispecies nitrogen transport, pore water electrical conductivity, and nitrate transport. The SBDF scheme demonstrated good accuracy in simulating the coupled model, with good agreement observed in comparison to reference solutions and experimental data.

The primary advantage of the proposed schemes lies in their noniterative approach, which generally converges for highly nonlinear constitutive soil relationships. Additionally, adaptive time and space techniques can be incorporated into the proposed schemes to improve their efficiency.

Our investigation reveals that introducing nonzero values for the free parameter β does not provide significant advantages in terms of stability or accuracy, while the free parameter α has a large impact on the schemes. The family of schemes with $\alpha = 1$ and $\beta \in [0, 1]$ yield approximately the same results, with the SBDF scheme $(\alpha, \beta) = (1, 0)$ being the best choice. Our results invite future theoretical studies for the SBDF scheme.

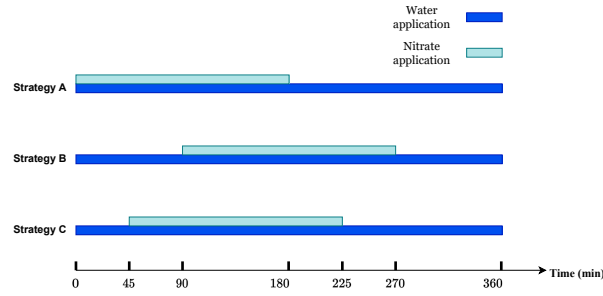


Figure 2.24 Schematic description of nitrate fertigation strategies.

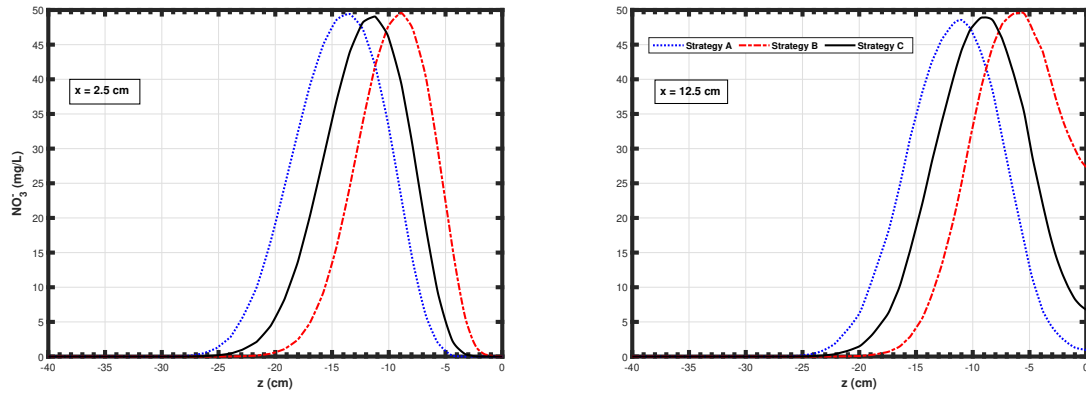


Figure 2.25 Cross-sections of nitrate concentration for each fertigation strategy at $x = 2.5$ cm (left) and $x = 12.5$ cm (right).

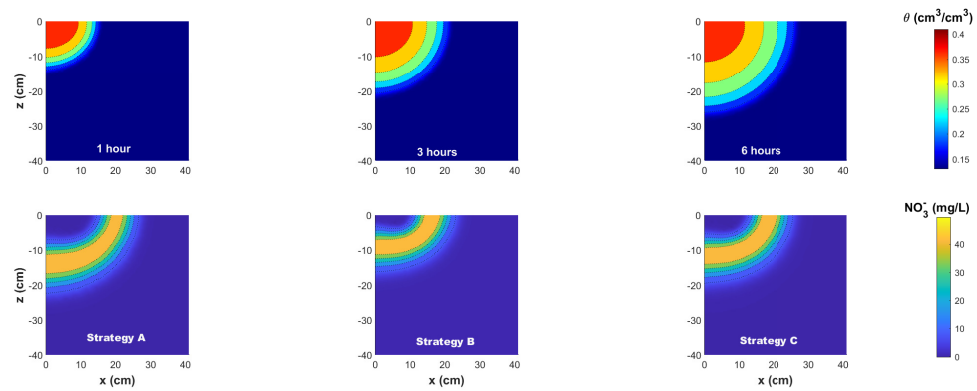


Figure 2.26 Water content at various times (top) and nitrate concentration at 6 hours for each fertigation strategy (down) using the SBDF scheme.

The future goal is to extend the SBDF scheme to model the complex interconnections between water infiltration in soils, nutrient uptake by plant roots, nutrient dynamics, and evapotranspiration. Furthermore, a three-dimensional extension of the proposed

schemes will be a focus of future endeavors, enabling the simulation of water-solute dynamics in more realistic configurations.

To summarize, this chapter introduced and validated an efficient second-order semi-implicit numerical scheme for modeling coupled water flow and solute transport in unsaturated soils. The proposed non-iterative mixed FEM showed satisfactory accuracy and stability, making it suitable for generating high-fidelity reference solutions to be used as benchmarks for training and evaluation of machine learning models. In the following chapters, we shift focus toward scientific machine learning-based predictive algorithms. Specifically, the next chapter is dedicated to developing a data-free PINN model and assessing its performance by comparing it against the FEM reference solutions.

Appendix. Error Analysis

We present a proof of the order of accuracy for the proposed time-stepping schemes. We used the following notations:

$$\left\{ \begin{array}{l} \llbracket \partial_t S_h^{n+1} \rrbracket_\alpha = \frac{\left(\alpha + \frac{1}{2}\right) S_h(t_{n+1}) - 2\alpha S_h(t_n) + \left(\alpha - \frac{1}{2}\right) S_h(t_{n-1})}{\Delta t}, \\ \llbracket K_h^{n+1} \rrbracket_\alpha = (\alpha + 1)K(\Psi_h(t_n)) - \alpha K(\Psi_h(t_{n-1})), \\ \llbracket J_h^{n+1} \rrbracket_{\alpha, \beta} = \left(\alpha + \frac{\beta}{2}\right) J'(S_h(t_n))(S_h(t_{n+1}) - S_h(t_n)) + \left(1 - \frac{\beta}{2}\right) J(S_h(t_n)) + \frac{\beta}{2} J(S_h(t_{n-1})), \\ \partial_t^n(*) = \frac{\partial^n(*)}{\partial t^n}. \end{array} \right. \quad (2.65)$$

For the sake of simplicity, we have omitted the inclusion of the spatial coordinate in our analysis.

Under the assumption of sufficient smoothness, and by centering our scheme around the time $t_{n+\alpha}$, we can employ Taylor series to provide a proof for the following equations:

$$\begin{aligned} \llbracket \partial_t S_h^{n+1} \rrbracket_\alpha &= \partial_t S_h(t_{n+\alpha}) + \frac{\Delta t^2}{6} \left((1-\alpha)^3 \left(\alpha + \frac{1}{2} \right) \partial_t^3 S_h(\bar{t}_1) + 2\alpha^4 \partial_t^3 S_h(\bar{t}_2) \right. \\ &\quad \left. - (1+\alpha)^3 \left(\alpha - \frac{1}{2} \right) \partial_t^3 S_h(\bar{t}_3) \right) = \partial_t^3 S_h(t_{n+\alpha}) + O(\Delta t^2), \end{aligned} \quad (2.66)$$

where $\bar{t}_1, \bar{t}_2, \bar{t}_3 \in [t_{n-1}, t_{n+1}]$.

$$\begin{aligned} \llbracket K_h^{n+1} \rrbracket_\alpha = (\alpha + 1) & \left[K(\Psi_h(t_{n+\alpha})) + \left(\Psi_h(t_n) - \Psi_h(t_{n+\alpha}) \right) K'(\Psi_h(t_{n+\alpha})) + \right. \\ & \left. \frac{(\Psi_h(t_n) - \Psi_h(t_{n+\alpha}))^2}{2} K''(\Psi_h(\bar{c}_1)) \right] - \alpha \left[K(\Psi_h(t_{n+\alpha})) + \left(\Psi_h(t_{n-1}) - \Psi_h(t_{n+\alpha}) \right) \times \right. \\ & \left. K'(\Psi_h(t_{n+\alpha})) + \frac{(\Psi_h(t_{n-1}) - \Psi_h(t_{n+\alpha}))^2}{2} K''(\Psi_h(\bar{c}_2)) \right], \end{aligned} \quad (2.67)$$

where $\bar{c}_1, \bar{c}_2$ are between $\Psi_h(t_{n-1})$ and $\Psi_h(t_{n+\alpha})$. We expand again $\Psi_h(t_{n-1}), \Psi_h(t_n)$ in (2.67) around the time level $t = t_{n+\alpha}$:

$$\Psi_h(t_n) = \Psi_h(t_{n+\alpha}) - \alpha \Delta t \partial_t \Psi_h(t_{n+\alpha}) + \frac{(\alpha \Delta t)^2}{2} \partial_t^2 \Psi_h(\bar{\beta}_1), \quad (2.68)$$

we get:

$$\begin{aligned} \llbracket K_h^{n+1} \rrbracket_\alpha = K(\Psi_h(t_{n+\alpha})) + \frac{\Delta t^2}{2} & \left[K'(\Psi_h(t_{n+\alpha})) \left((\alpha + 1) \alpha^2 \partial_t^2 \Psi_h(\bar{\beta}_1) - (1 + \alpha)^2 \alpha \partial_t^2 \Psi_h(\bar{\beta}_2) \right) + \right. \\ & \left. \alpha^2 (1 + \alpha) K''(\Psi_h(\bar{c}_1)) (\partial_t \Psi_h(\bar{\gamma}_1))^2 - \alpha (1 + \alpha)^2 K''(\Psi_h(\bar{c}_2)) (\partial_t \Psi_h(\bar{\gamma}_2))^2 \right], \end{aligned} \quad (2.69)$$

where $\bar{\beta}_1, \bar{\beta}_2, \bar{\gamma}_1, \bar{\gamma}_2 \in [t_{n-1}, t_{n+\alpha}]$.

$$\begin{aligned} \llbracket J_h^{n+1} \rrbracket_{\alpha, \beta} = (\alpha + \frac{\beta}{2}) & \left[J(S_h(t_{n+\alpha})) + \left(S_h(t_{n+1}) - S_h(t_{n+\alpha}) \right) J'(S_h(t_{n+\alpha})) + \right. \\ & \left. \frac{(S_h(t_{n+1}) - S_h(t_{n+\alpha}))^2}{2} J''(S_h(\bar{g}_1)) \right] + (1 - \alpha - \beta) \left[J(S_h(t_{n+\alpha})) + \right. \\ & \left. \left(S_h(t_n) - S_h(t_{n+\alpha}) \right) J'(S_h(t_{n+\alpha})) + \frac{(S_h(t_n) - S_h(t_{n+\alpha}))^2}{2} J''(S_h(\bar{g}_2)) \right] + \\ & \frac{\beta}{2} \left[J(S_h(t_{n+\alpha})) + \left(S_h(t_{n-1}) - S_h(t_{n+\alpha}) \right) J'(S_h(t_{n+\alpha})) + \right. \\ & \left. \frac{(S_h(t_{n-1}) - S_h(t_{n+\alpha}))^2}{2} J''(S_h(\bar{g}_3)) \right], \end{aligned} \quad (2.70)$$

where $\bar{g}_1, \bar{g}_2, \bar{g}_3 \in [t_{n-1}, t_{n+\alpha}]$.

$$\begin{aligned} \llbracket J_h^{n+1} \rrbracket_{\alpha, \beta} = & J(S_h(t_{n+\alpha})) + \frac{\Delta t^2}{2} \left[J'(S_h(t_{n+\alpha})) \left((\alpha + \frac{\beta}{2})(1 - \alpha)^2 \partial_t^2 S_h(\bar{\alpha}_1) + \right. \right. \\ & (1 - \alpha - \beta) \alpha^2 \partial_t^2 S_h(\bar{\alpha}_2) + \frac{\beta}{2} (1 + \alpha)^2 \partial_t S_h(\bar{\alpha}_3) \Big) + \\ & (\alpha + \frac{\beta}{2})(1 - \alpha)^2 J''(S_h(\bar{d}_1)) (\partial_t S_h(\bar{l}_1))^2 + (1 - \alpha - \beta) \alpha^2 J''(S_h(\bar{d}_2)) (\partial_t S_h(\bar{l}_2))^2 + \\ & \left. \left. \frac{\beta}{2} (1 + \alpha)^2 J''(S_h(\bar{d}_1)) (\partial_t S_h(\bar{l}_3))^2 \right) \right], \end{aligned} \tag{2.71}$$

where $\bar{\alpha}_1, \bar{\alpha}_2, \bar{l}_1, \bar{l}_2, \bar{l}_3 \in [t_{n-1}, t_{n+\alpha}]$.

We conclude that all the time approximation schemes maintain second-order accuracy.

CHAPTER 3 Physics-informed neural networks vs finite element method for modeling coupled water and solute flow in unsaturated soils

Abstract

Accurate modeling of water infiltration and solute transport in unsaturated soils is critical for various applications. These include optimizing irrigation practices to conserve water and minimize environmental impact, as well as predicting the fate of contaminants in soil and groundwater. This study explores the application of the vanilla physics-informed neural network (PINN) approach for modeling the coupled system of water flow and solute transport in unsaturated soils. We compare the performance of PINN with the Galerkin finite element method (FEM) to evaluate their effectiveness. Various techniques are implemented to improve the PINN solver, including adaptive activation functions. Numerical tests were carried out to evaluate the efficiency of the PINN solver in comparison to the FEM. The findings reveal that PINN can achieve accuracy comparable to FEM, albeit at a significantly higher computational cost during training, while maintaining fast inference times.

3.1 Introduction

Soil moisture and solute dynamics are fundamental to hydrological modeling, influencing a wide range of environmental and engineering challenges. Precise knowledge of water flow and solute behavior in soils is essential to tackle diverse environmental problems. Understanding these dynamics is critical for developing efficient irrigation systems (Dastorani *et al.*, 2010), which ultimately leads to improved agricultural product quality. Modeling water flow and solute transport in unsaturated soils traditionally relies on a coupled system of the Richards equation (Richards, 1931) for water flow, and advection-dispersion equations (Bear, 2013) to describe solute movement. This approach necessitates solving a complex system of partial differential equations (PDEs). The highly nonlinear nature of the Richards equation, coupled with the intricate nature of realistic boundary conditions, renders analytical solutions for the coupled water flow and solute transport model highly challenging to obtain. Consequently, numerical methods provide a powerful alternative approach to solving these PDEs. While various numerical methods exist for solving the Richards equation alone (Keita *et al.*, 2021c), their application to the coupled model of water flow and solute transport remains less explored (Kamil *et al.*, 2024a; Toutlini *et al.*, 2024). Standard numerical solvers often face challenges due to their computational demands, particularly during mesh generation within the domain. Additionally, these methods require adherence to strict time step and mesh size limitations to ensure accurate solutions. Furthermore, they typically assume precise knowledge of initial/boundary conditions and soil properties, which can be unrealistic in practical scenarios where uncertainty and incomplete

information about these conditions are common.

Data-driven approaches offer an alternative surrogate to standard numerical methods. A recent method, introduced as physics-informed neural networks (PINNs), aims to enhance efficiency and alleviate the requirement for extensive training data (Raissi *et al.*, 2019). In the PINN approach, a deep learning solver is trained to solve a desired PDE while adhering to the underlying physical laws encoded within it. This methodology has demonstrated significant advancements in solving various engineering problems (e.g., Roy *et al.* (2023)). Other variants of PINNs have been proposed to enhance the approach's efficiency in tackling complex PDEs with discontinuous and sharp solutions (Kharazmi *et al.*, 2021).

PINN solvers have emerged as powerful tools for simulating water flow and solute transport, consistently achieving impressive results. For instance, Depina *et al.* (Depina *et al.*, 2022) employed PINNs to estimate unknown soil parameters, while Bandai *et al.* (Bandai and Ghezzehei, 2021) developed a PINN framework to estimate key soil properties such as the water retention curve, hydraulic conductivity function, and surface water flux. Furthermore, Haruzi *et al.* (Haruzi and Moreno, 2023a) addressed the coupled model to predict pressure head and pore water electrical conductivity with unknown initial conditions.

A critical gap persists in the literature concerning coupled models for water flow and solute transport, particularly in scenarios where no data are available. Building upon prior comparisons between the FEM and PINNs for various PDEs (Grossmann *et al.*, 2023), this study specifically concentrates on a comparative analysis of their efficacy, with a focus on accuracy and efficiency, in solving the forward problem of the highly nonlinear Richards equation coupled with the advection-dispersion equation. We integrate the adaptive activation function (Jagtap *et al.*, 2020) to enhance the training of the PINN solver. Two numerical tests, namely, 1D and 2D tests, are conducted to provide a detailed comparison of the two approaches.

3.2 The physical models

3.2.1 Water flow

Modeling water flow in unsaturated soils is typically described using the Richards equation (Richards, 1931), which is represented by the following equation:

$$\begin{cases} \frac{\partial \theta}{\partial t} + \nabla \cdot \mathbf{q} = 0, \\ \mathbf{q} = -K \nabla (\Psi + z), \end{cases} \quad (3.1)$$

where Ψ is the pressure head, z is the vertical dimension, $\mathbf{q}$ is the water flux, $K = K_s k_r$ is the unsaturated hydraulic conductivity, K_s is the saturated hydraulic conductivity, k_r is the water relative permeability, and θ is the volumetric water content.

To solve the Richards equation, it is essential to provide explicit functions for K and

θ as functions of Ψ . In this paper, we employed the Mualem-van Genuchten model (Van Genuchten, 1980) expressed by the following equations:

$$\theta = \begin{cases} \theta_r + (\theta_s - \theta_r) \left(1 + (\alpha_v |\Psi|)^{n_v} \right)^{-m_v}, & \text{if } \Psi \leq 0 \\ \theta_s, & \text{if } \Psi > 0 \end{cases} \quad (3.2a)$$

$$K = \begin{cases} K_s \frac{\left[1 - (\alpha_v |\Psi|)^{n_v-1} \left(1 + (\alpha_v |\Psi|)^{n_v} \right)^{-m_v} \right]^2}{\left(1 + (\alpha_v |\Psi|)^{n_v} \right)^{m_v/2}}, & \text{if } \Psi \leq 0 \\ K_s, & \text{if } \Psi > 0 \end{cases} \quad (3.2b)$$

$$m_v = 1 - \frac{1}{n_v}; \quad n_v > 1, \quad (3.2c)$$

where θ_s and θ_r represent the saturated and residual volumetric water contents, respectively; n_v , m_v and α_v are the van Genuchten empirical parameters.

3.2.2 Solute transport

To account for solute transport alongside water flow, the equilibrium advection-dispersion equation is typically coupled with the Richards equation, resulting in the following governing model (Bear, 2013):

$$\frac{\partial \theta c}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c - c \mathbf{q}), \quad (3.3)$$

where c represents the concentration of the solute in the liquid phase, $\mathbf{D}$ represents the dispersion tensor, θ and $\mathbf{q}$ represent the water content and water flux, respectively. The components of the dispersion tensor are computed as follows (Bear, 2013):

$$\theta \mathbf{D}_{ij} = (D_T \|\mathbf{q}\| + \theta D_w \tau) \delta_{ij} + (D_L - D_T) \frac{\mathbf{q}_i \mathbf{q}_j}{\|\mathbf{q}\|}. \quad (3.4)$$

Here, D_w denotes the molecular diffusion constant, while D_L and D_T represent the longitudinal and transversal dispersivities, respectively. The symbol δ_{ij} denotes the Kronecker symbol, while τ represents the tortuosity factor in the liquid phase. The Euclidean norm of $\mathbf{q}$ is denoted by $\|\mathbf{q}\|$, where $\mathbf{q}_i$ and $\mathbf{q}_j$ represent the components of the water flux in the corresponding spatial directions. The tortuosity factor is determined using the formula (Bear, 2013): $\tau = \frac{\theta^{7/3}}{\theta_s^2}$.

3.3 Methods

3.3.1 Finite element method: FEM

We adopt the mixed FEM model developed in Chapter 2 as the reference solution for evaluation. For further details on the formulation and implementation, the reader is referred to that chapter.

3.3.2 Physics-informed neural networks: PINNs

In this study, we utilize the vanilla PINN (Raissi *et al.*, 2019) to address the coupled model of water flow and solute transport in unsaturated soils. We approximate the solutions of the PDEs given by equations (5.1)-(3.3), $\Psi(t, x)$ and $c(t, x)$, using a feed-forward neural network (FNN). The outputs of the FNN are represented as $\hat{\Psi}(t, x, \Theta)$ and $\hat{c}(t, x, \Theta)$, where Θ denotes the trainable parameters of the neural network, and the inputs consist of the time-space coordinates (t, x) . We gather data comprising collocation points distributed throughout the time-space domain $I \times \Omega$. These data points are classified into residual collocation points $(t^r, x^r) \in (0, T] \times \Omega$, initial collocation points $(0, x^{ic}) \in \{0\} \times \Omega$, and boundary collocation points $(t^b, x^b) \in (0, T] \times \partial\Omega$. Subsequently, the following residual functions can be derived using automatic differentiation techniques (Baydin *et al.*, 2018):

$$\begin{cases} \hat{f}_R(t^r, x^r) = \frac{\partial \hat{\theta}}{\partial t} - \nabla \cdot \left[\hat{K} \nabla (\hat{\Psi} + z) \right], \\ \hat{f}_C(t^r, x^r) = \frac{\partial \hat{\theta} \hat{c}}{\partial t} - \nabla \cdot (\hat{\theta} \hat{\mathbf{D}} \nabla \hat{c} - \hat{c} \hat{\mathbf{q}}). \end{cases} \quad (3.5)$$

The hat notation denotes predictions made by the FNN. The training of the PINN solver involves minimizing the following loss function:

$$J(\Theta) = \alpha_r J_r(\Theta) + \alpha_b J_b(\Theta) + \alpha_{ic} J_{ic}(\Theta), \quad (3.6)$$

where

$$J_r(\Theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \left(\hat{f}_R(t_i^r, x_i^r) \right)^2 + \left(\hat{f}_C(t_i^r, x_i^r) \right)^2, \quad (3.7a)$$

$$J_{ic}(\Theta) = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} \left(\Psi(0, x_i^{ic}) - \hat{\Psi}(0, x_i^{ic}) \right)^2 + \left(c(0, x_i^{ic}) - \hat{c}(0, x_i^{ic}) \right)^2. \quad (3.7b)$$

Here, J_r , J_b , and J_{ic} represent the residual, boundary, and initial condition losses, respectively. Further, N_r and N_{ic} denote the total number of residuals and initial collocation points. The term J_b accounts for the disparities between the predicted value of the FNN and the true boundary value. Its formulation depends on the specific type of boundary condition utilized.

In both the 1D and 2D numerical tests, we utilized a total of 10,000 residual collocation points and 2,000 initial/boundary collocation points. The FNN architecture consists of 5 hidden layers, each containing 50 neurons. The output function $-\exp(x)$ is applied to the neuron corresponding to the predicted pressure head to enforce its negativity, while the identity function is employed for the neuron representing solute concentration. The activation function employed in the FNN is the hyperbolic tangent with an adaptive activation technique (Jagtap *et al.*, 2020), with the learning parameter initialized to 0.05 and the scale factor set to 20. The weights of the FNN parameters were initialized using the Xavier random initialization method (Glorot and Bengio, 2010a), while the bias parameters b were set to zero initially. A two-step training technique was implemented, employing two optimizers sequentially, starting with Adam (Kingma and Ba, 2014) and followed by L-BFGS-B (Byrd *et al.*, 1995). We employed the Adam optimizer with exponential decay of the learning rate for optimization. The initial learning rate was configured to 10^{-3} , with a decay rate of 0.9, and the decay step was set to 1000. Additionally, the maximum number of iterations was limited to 20000. Other parameters were left at their default values. The parameters of the L-BFGS-B optimizer were set as follows: $\text{ftol} = 10^{-10}$, $\text{gtol} = 10^{-8}$, $\text{maxcor} = 50$, $\text{maxls} = 50$, $\text{maxiter} = 50,000$, and $\text{maxfun} = 50,000$, while the remaining parameters were kept at their default values. To improve training, a mini-batching technique was applied for the Adam optimizer, whereas a full-batch approach was employed for the L-BFGS-B optimizer. The implementation of the PINN solver was performed using the TensorFlow library (Abadi *et al.*, 2016). Figure 3.1 presents a conceptual illustration of the PINN solver for modeling water flow and solute transport in unsaturated soils.

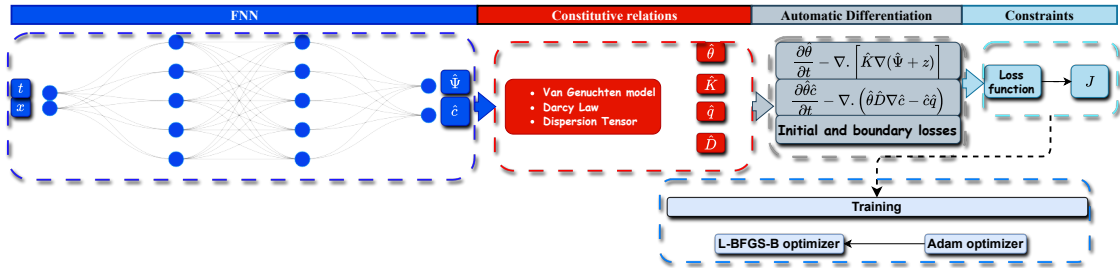


Figure 3.1 Diagram illustrating the PINN solver for the coupled water flow and solute transport model.

3.4 Numerical experiments

In this section, we present two numerical experiments to compare the performance of PINN and FEM for solving the coupled model of water flow and solute transport. The first experiment involves a one-dimensional soil column test, while the second numerical

test is conducted on a 2D soil with different boundary conditions.

For both numerical experiments, we used the following loamy soil parameters: $K_s = 0.2496 \text{ m/day}$, $\alpha_v = 3.6 \text{ m}^{-1}$, $\theta_s = 0.43$, $\theta_r = 0.078$, $n_v = 1.56$. For the one-dimensional experiment, the initial and boundary conditions are outlined as follows: For $(t, z) \in (0, 1] \times [-1, 0]$, $\Psi(0, z) = -1 \text{ m}$, $\mathbf{q}(t, 0) = -0.05 \text{ m/day}$, $\frac{\partial \Psi}{\partial z}(t, -1) = 0$, $c(0, z) = 0 \text{ mmol/m}^3$, $\bar{c}_0 = 50 \text{ mmol/m}^3$, $\frac{\partial c}{\partial z}(t, -1) = 0$, where $\bar{c}_0$ indicates the concentration of the applied solute at the soil surface. The following solute parameters were used: $D_L = 0.1 \text{ m}$, $D_w = 0.0 \text{ m}^2/\text{day}$.

For the two-dimensional experiment, the initial and boundary conditions are outlined as follows: For $(t, x, z) \in (0, 1] \times [-1, 0] \times [-1, 1]$, $\Psi(0, x, z) = -1.3 \text{ m}$, $\Psi(t, x, 0) = -0.2 \text{ m}$, $\frac{\partial \Psi}{\partial z}(t, x, 0) = 0$, $c(0, x, z) = 0.1 \text{ g/m}^3$, $c(t, x, 0) = 1 \text{ g/m}^3$, $\frac{\partial c}{\partial z}(t, x, -1) = 0$. Additionally, a no-flux boundary condition was enforced along the lateral sides of the domain. The following solute parameters were used: $D_L = 0.5 \text{ m}$, $D_T = 0.1 \text{ m}$, $D_w = 0.0 \text{ m}^2/\text{day}$.

To provide a benchmark for comparison with the PINN solution described in 3.3.2, we computed a reference solution using the FEM model developed in Chapter 2. For the 1D numerical test, the reference solution was obtained with $\Delta t_{\text{ref}} = 10^{-5} \text{ day}$ and $\Delta z_{\text{ref}} = 10^{-4} \text{ m}$. In the 2D numerical test, the reference time step used was $\Delta t_{\text{ref}} = 10^{-4} \text{ day}$, and the computational domain was discretized into a 100×100 grid. The obtained results are depicted in Figure 3.2, and a summary of accuracy and computational cost is illustrated in Table 3.1. Both FEM and PINN demonstrate good accuracy for pressure head and solute concentrations, with errors lower than 1% for both numerical tests. However, PINN exhibits significantly faster prediction times compared to FEM. Therefore, combining these two techniques could show promise for advancing subsurface modeling, particularly in terms of fast and accurate predictive capabilities.

Table 3.1 The obtained results for both PINN and FEM in solving the one- and two-dimensional infiltration-solute model.

1D numerical test					
	CPU of training (seconds)	CPU of prediction (seconds)	relative L^2 -error for Ψ	relative L^2 -error for c	Total loss
FEM	-	5.29	1.9e-3	2.2e-3	-
PINN	619.45	0.544	2.19e-3	3.42e-3	6.05e-3
2D numerical test					
FEM	-	58.75	1.5e-3	1.09e-3	-
PINN	1572.6	0.55	5.9e-3	3.06e-3	1.86e-05

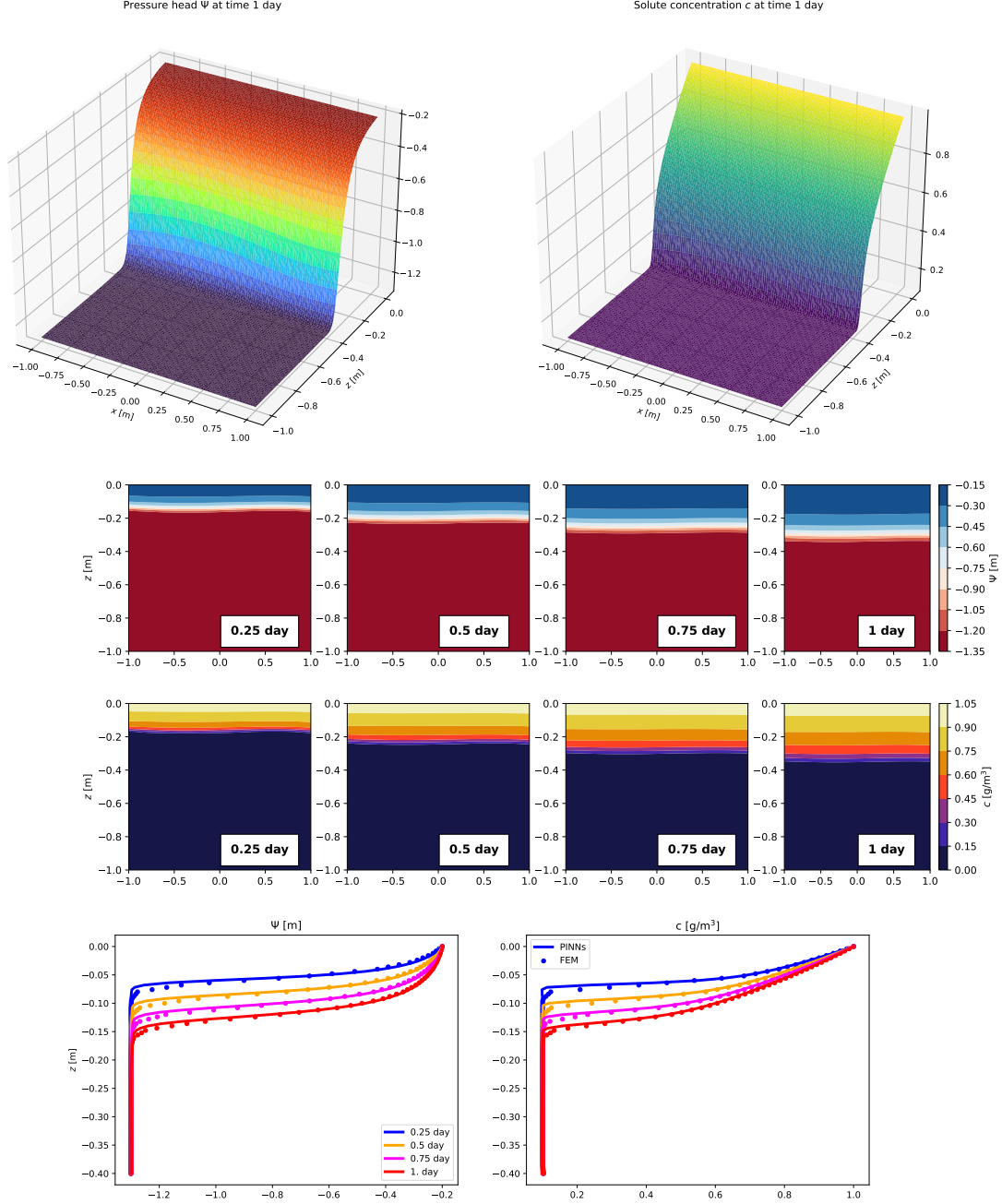


Figure 3.2 2D numerical test- Snapshots of the obtained results from PINN and cross-sections at $x = 0$ of the pressure head and solute concentration obtained using PINN and FEM solvers.

3.5 Conclusion

This study highlights the accuracy and effectiveness of utilizing a vanilla PINN approach to solve the coupled model of water flow and solute transport. It compares the efficacy of this approach with the mixed Galerkin FEM. PINN is enhanced with

techniques such as adaptive activation functions to better address the high nonlinearity inherent in the Richards equation. The results indicate that PINN can attain accuracy levels similar to FEM, despite requiring notably more computational resources during training. However, PINN maintains quick inference times, suggesting its potential as a viable alternative for modeling water-solute systems, albeit with a trade-off in computational cost.

The next chapter focuses on identifying effective PINN strategies for modeling water infiltration and single- or multi-species solute transport in unsaturated soils. A comparative study is conducted for both forward and inverse problems using sparse and noisy data. One- and two-dimensional scenarios are considered, and the results are evaluated against FEM reference solutions and experimental data.

CHAPTER 4 A comparative study of physics-informed neural network strategies for modeling water and nitrogen transport in unsaturated soils

Abstract

A deep understanding of subsurface flow dynamics, including water infiltration and the transport of single or multiple solutes in unsaturated soils, is critical for a wide range of engineering applications. Traditionally, these complex processes have been modeled using standard numerical solvers, which remain conventional in many studies. However, a recent and promising methodology gaining traction is physics-informed neural networks (PINNs). This approach is based on training neural networks to solve partial differential equations by combining available data with the physical principles embedded in the equations. In this study, we analyse several PINN solvers to tackle the coupled model of water flow and single or multispecies solute transport in unsaturated soils. This model is governed by the highly nonlinear Richards equation and advection-dispersion equations. To improve the training of the solvers, we integrate several strategies aimed at capturing the system's full complexity.

The numerical experiments cover one- and two-dimensional scenarios, tackling forward and inverse problems. The results obtained from PINN are compared with reference solutions and experimental data sourced from existing literature. Our analysis underscores the effectiveness of employing sequential training alongside an adaptive activation technique for modeling the coupled water-solute system. This methodology not only improves accuracy and training efficiency but also enables an accurate estimation of the unknown ammonium nitrification rate from sparse measurements.

4.1 Introduction

Modeling water flow and the transport of solutes in unsaturated soils is a critical task with numerous applications. For instance, it helps determine the quality and quantity of water recharging a saturated zone (Feyen *et al.*, 1998). This information is vital for designing efficient irrigation systems (Kandelous and Šimůnek, 2010) and managing the impact of water and solutes on groundwater (Perrin *et al.*, 2011). Modeling the dynamics of nitrogen and other solute species, is essential for comprehending their migration within the root zone. This understanding aids in making informed decisions regarding their efficient application (Li *et al.*, 2015).

The most widely adopted approach for modeling flow and transport phenomena relies on applying the mass- and momentum-balance physical laws represented by partial differential equations (PDEs). Within this framework, the Richards equation serves as the fundamental basis for modeling water flow in the vadose zone. Different extensions to the Richards equation have been proposed for modeling infiltration in porous media (see, e.g. DiCarlo (2013); Xiong (2014); Cueto-Felgueroso and Juanes (2009); Eliassi

and Glass (2002) for an overview). Beljadid, Cueto-Felgueroso, and Juanes (Beljadid *et al.*, 2020) proposed a continuum model of unstable infiltration through porous media. The authors developed a new PDE: the BCJ-R equation (Equation 18 in (Beljadid *et al.*, 2020)), which is endowed with an entropy function as an extension of the traditional Richards equation. The BCJ-R equation (Beljadid *et al.*, 2020) is based on a second gradient theory using a free energy functional as a symmetric expansion of the capillary energy density in the Richards equation. Advection-dispersion models can be incorporated to simulate the dynamic behavior of solutes. Despite their widespread use, these mathematical models frequently lack analytical solutions, particularly when confronted with complex geometries and boundary conditions. Conversely, numerical methods offer a practical solution for coupled water flow and solute transport models, effectively handling complex geometries and different boundary conditions. In the scientific literature, a variety of numerical techniques have been proposed for solving the coupled model of water flow and solute transport (Chen *et al.*, 2019b; Kamil *et al.*, 2024b; Toutlini *et al.*, 2025b). Despite their efficiency and versatility in handling PDEs, these numerical methods come with certain limitations: (a) For complex geometries, mesh generation can be computationally expensive and complex (Karniadakis *et al.*, 2021); (b) Solving high-dimensional problems poses some challenges, such as the curse of dimensionality (Hu *et al.*, 2024); (c) Simulating physical phenomena becomes challenging when initial/boundary conditions are unknown or when the physics is incomplete (Moreno and Paster, 2018); and (d) Stability criteria must be respected, imposing restrictions on the time step and mesh size, particularly in the presence of significant changes in the computational domain.

Deep learning methods (DLs) are an alternative approach for approximating any desired linear or nonlinear function, as established by their universal approximation property (Hornik *et al.*, 1989). In DLs, neural networks are trained to capture the relationships between inputs and outputs. One notable advantage of DLs is their ability to directly address nonlinear problems without the need for linearization techniques or time-space discretizations. However, despite their universal approximation capabilities, standard DL approaches may struggle to provide acceptable solutions when faced with highly nonlinear problems, especially in the absence of constraints derived from the physical laws encoded in the PDEs (Raissi *et al.*, 2019).

Recently, a novel methodology that integrates physical constraints with available data is to use physics-informed neural networks (PINNs) (Raissi *et al.*, 2019). In this approach, a neural network is employed to approximate the solution of the desired PDEs while adhering to their prescribed initial and boundary conditions. The fundamental difference between PINNs and conventional DLs lies in the definition of the loss function that incorporates the physical constraints. PINNs have garnered considerable attention within the scientific community, primarily owing to their meshfree nature and ability to compute derivatives of the neural network inputs without relying on a time and space stepping scheme. This latter attribute is possible mainly by using the

chain rule for differentiating the compositions of functions and employing automatic differentiation methods (Baydin *et al.*, 2018). Another contributing factor is the significant advancements in CPUs (central processing units) and GPUs (graphics processing units), which accelerate the training process of DL algorithms. The main advantage of PINNs over other meshless methods is their ability to be trained on randomly generated points, with the flexibility to easily adapt and regenerate collocation points as needed (Lu *et al.*, 2021b).

Several studies have successfully employed the PINN approach to solve various engineering problems (Bertels and Willems, 2023; Cai *et al.*, 2021a; Guo *et al.*, 2023). PINNs have also been effectively used for modeling water flow (Bandai and Ghezzehei, 2021; Tartakovsky *et al.*, 2020; Depina *et al.*, 2022; Chen *et al.*, 2023b; Lan *et al.*, 2024a; Vemuri *et al.*, 2024; Zhang *et al.*, 2023), as well as for modeling coupled water flow with solute transport or heat transfer in unsaturated soils (Haruzi and Moreno, 2023a; Wang *et al.*, 2023b; Kamil *et al.*, 2024e,c; Faroughi *et al.*, 2023). For a comprehensive review of the applications of PINNs in fluid and solid mechanics, as well as their limitations, we refer readers to the literature review by (Faroughi *et al.*, 2024).

In the studies mentioned above, when addressing ill-posed problems, PINN models were trained on sparse data. These cases often involved uncertainties regarding initial or boundary conditions, or the complex relationship between soil moisture and matric head. Notably, all of these studies effectively tackled inverse problems, demonstrating the robust efficacy of PINNs as powerful solvers in such complex scenarios.

The development of efficient PINN solvers for coupled water flow and solute transport in unsaturated soils remains limited (Haruzi and Moreno, 2023a; Wang *et al.*, 2023b; Kamil *et al.*, 2024c,e). Furthermore, a significant gap persists in the literature regarding the modeling of multicomponent solute transport using PINNs. In particular, using a standard PINN solver for solving systems involving multiple coupled PDEs has been shown to be inefficient (Laubscher, 2021; Kamil *et al.*, 2024c). This inefficiency can be attributed to several fundamental challenges inherent to the problem. These include the coupling between water flow and solute dynamics, the high nonlinearity of the Richards equation, and the need to simultaneously predict multiple hydraulic and transport variables. These difficulties are further compounded in multispecies transport problems, where modeling the interactions among several solutes, such as nitrogen species (nitrate, ammonium) and other ions (potassium, calcium), requires solving complex systems of coupled PDEs. Therefore, the development of efficient PINN solvers is essential for accurately capturing these processes and for supporting the design of effective irrigation and fertigation management strategies.

In this study, we aim to investigate the effectiveness of PINNs in solving the coupled model of water flow and single or multi-species solute transport in unsaturated soils. We explore various one-dimensional (1D) and two-dimensional (2D) scenarios, addressing both forward and inverse problems. The main contribution of this study is the development of three distinct PINN solvers, which are evaluated across various

benchmarks to determine the most effective approach. The first solver is the standard PINN model, where a single neural network is utilized to predict all the desired variables simultaneously. The second solver employs a multi-neural network architecture (MPINN), with separate neural networks dedicated to computing each variable of interest. Finally, we introduce a sequential PINN solver (S-PINN), which decouples the training of neural networks, allowing each PDE to be solved independently. To enhance the performance and computational efficiency of the proposed PINN solvers, we incorporate advanced optimization techniques such as the adaptive activation function (Jagtap *et al.*, 2020), and sequential optimization using both the Adam and L-BFGS algorithms. The selection of the standard PINN, MPINN, and S-PINN solvers was motivated by the need to systematically evaluate different strategies for addressing the challenges inherent to coupled flow and multispecies transport in unsaturated soils. The standard PINN provides a baseline for simultaneous prediction, MPINN allows for specialized networks for each variable, and S-PINN enables independent training to better manage complex multiphysics interactions. These approaches were chosen to investigate how variable coupling, network interaction, and training dynamics affect the overall efficiency, accuracy, and robustness of PINN solvers for multicomponent subsurface modeling.

Our primary focus in this study is on forward problems where no data is assumed to be available for training the PINN solvers. Another objective of this study is to determine the most efficient PINN solver for modeling multiple solute species in unsaturated soils, with a specific focus on the nitrogen first-order decay chain. In this context, we solve a set of coupled advection-diffusion-reaction PDEs to derive the variables of interest. In addition, we address the inverse problem of identifying the nitrification rate of ammonium in a hypothetical scenario. This study aims to enhance the application of PINNs to challenging scenarios in hydrological modeling, particularly within unsaturated porous media, and to establish them as a viable and efficient alternative to standard numerical solvers.

The rest of the paper is structured as follows. Section 7.2 outlines the governing equations for water flow and solute transport in unsaturated soil, including the initial and boundary conditions. The methodology for the proposed PINN solvers is explained in Section 5.4. The numerical results are presented and discussed in Section 5.5. Sections 4.5 and 4.6 offer a summary of the study, key conclusions, and outline some future perspectives.

4.2 The mathematical model

4.2.1 Water flow model

In this study, we use the Richards equation to model water infiltration into unsaturated soils. It is derived by combining the mass conservation principle and Darcy-Buckingham's law (Buckingham, 1907). The expression of the Richards equation in

the mixed form is as follows:

$$\begin{cases} \frac{\partial \theta}{\partial t} + \nabla \cdot \mathbf{q} = 0, \\ \mathbf{q} = -K \nabla (\Psi + z), \end{cases} \quad (4.1)$$

where Ψ is the matric head, z is the vertical dimension directed upward, $\mathbf{q}$ is the water flux, $K = K_s k_r$ is the unsaturated hydraulic conductivity, K_s is the saturated hydraulic conductivity, k_r is the water relative permeability, and θ is the volumetric water content. The time coordinate is represented by t and $\mathbf{x} = (x, z)$ represents the spatial coordinates, where x signifies the horizontal position in two-dimensional scenarios.

The van Genuchten model (Van Genuchten, 1980) is employed to describe the water content θ as a function of Ψ , while the unsaturated hydraulic conductivity K is given according to Mualem's model (Mualem, 1976). The full expressions are as follows:

$$\theta = \begin{cases} \theta_r + (\theta_s - \theta_r) \left(1 + (\alpha_v |\Psi|)^{n_v} \right)^{-m_v} & \text{if } \Psi \leq 0, \\ \theta_s & \text{if } \Psi > 0, \end{cases} \quad (4.2a)$$

$$K = \begin{cases} K_s \frac{\left[1 - (\alpha_v |\Psi|)^{n_v-1} \left(1 + (\alpha_v |\Psi|)^{n_v} \right)^{-m_v} \right]^2}{\left(1 + (\alpha_v |\Psi|)^{n_v} \right)^{m_v/2}} & \text{if } \Psi \leq 0, \\ K_s & \text{if } \Psi > 0, \end{cases} \quad (4.2b)$$

$$m_v = 1 - \frac{1}{n_v}, \quad n_v > 1, \quad (4.2c)$$

where θ_s and θ_r represent the saturated and residual volumetric water contents, respectively, and n_v , m_v and α_v are the van Genuchten empirical parameters.

4.2.2 Solute transport model

To model the coupled system of water flow and solute transport in unsaturated soils, the advection-dispersion equation (ADE) is coupled to the Richards equation. This ADE takes the following form (Bear, 2013; Šimůnek *et al.*, 2005):

$$\frac{\partial \theta c}{\partial t} + \frac{\partial \rho s}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c - c \mathbf{q}), \quad (4.3)$$

where c represents the concentration of the solute in the liquid phase, s denotes the concentration of the solute in the solid phase, ρ stands for the soil bulk density, and $\mathbf{D}$ represents the dispersion tensor. The dispersion tensor encompasses the impact of both molecular diffusion and hydrodynamic dispersion. Additionally, θ and $\mathbf{q}$ represent

the water content and water flux, respectively, as governed by the Richards equation (5.1).

The relationship between s and c can be expressed using the Freundlich equation (Šimůnek *et al.*, 2005):

$$s = K_d c^p, \quad (4.4)$$

where p is the Freundlich constant, which in the case of a linear model equals 1, and K_d is an empirical coefficient that is equal to zero when the interaction between the solute and soil particles is ignored.

The solute transport equation (5.3) can be modified into the following compact form by using the Freundlich equation (4.4):

$$\frac{\partial R \theta c}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c - c \mathbf{q}), \quad (4.5)$$

where the retardation factor, R , is defined by the following formula and takes into consideration the interaction between the solute and the solid phase of soil:

$$R = 1 + \rho K_d \frac{c^{p-1}}{\theta}. \quad (4.6)$$

The components of the dispersion tensor are calculated as follows when the gaseous phase is excluded (Bear, 2013):

$$\theta \mathbf{D}_{ij} = (D_T \|\mathbf{q}\| + \theta D_w \tau) \delta_{ij} + (D_L - D_T) \frac{\mathbf{q}_i \mathbf{q}_j}{\|\mathbf{q}\|}, \quad (4.7)$$

where D_w represents the molecular diffusion constant, while D_L and D_T denote the longitudinal and transversal dispersivities, respectively. The coefficient δ_{ij} stands for the Kronecker symbol, and τ represents the tortuosity factor in the liquid phase. The Euclidean norm of $\mathbf{q}$ is represented by $\|\mathbf{q}\|$, where $\mathbf{q}_i$ and $\mathbf{q}_j$ correspond to the components of the water flux in the respective spatial directions. The tortuosity factor is calculated using the formula provided in (Bear, 2013):

$$\tau = \frac{\theta^{7/3}}{\theta_s^2}. \quad (4.8)$$

In situations where multiple solute transport equations are involved, considering interactions between solutes is a standard practice. This is typically accomplished by incorporating decay coefficients (Šimůnek *et al.*, 2005). As explored in the test case 4.4.2, the dynamics of multiple solutes can be modeled using the following system

(Šimůnek *et al.*, 2005):

$$\begin{cases} \frac{\partial R\theta c_1}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c_1 - c_1 \mathbf{q}) - \mu_1 \theta c_1, \\ \frac{\partial \theta c_2}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c_2 - c_2 \mathbf{q}) + \mu_1 \theta c_1 - \mu_2 \theta c_2, \\ \frac{\partial \theta c_3}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c_3 - c_3 \mathbf{q}) + \mu_2 \theta c_2 - \mu_3 \theta c_3, \end{cases} \quad (4.9)$$

where the concentration and the first-order decay coefficient of solute k are indicated by c_k and μ_k . For additional details, especially in the context of multiple nitrogen species, refer to the numerical test 4.4.2.

4.2.3 Initial and boundary conditions

To complete equations (5.1) and (5.5), it is necessary to define suitable initial and boundary conditions. In this study, the following initial and boundary conditions are utilized for the Richards equation:

$$\begin{cases} \Psi(0, \mathbf{x}) = \Psi_0(\mathbf{x}) & \text{in } \Omega, \\ \Psi = \Psi_D & \text{in } I \times \Gamma_D \subset I \times \partial\Omega, \\ \mathbf{q} \cdot \mathbf{n} = q_0 & \text{in } I \times \Gamma_N \subset I \times \partial\Omega, \end{cases} \quad (4.10)$$

where $I \times \Omega$ represents the time-space domain, with $I = (0, T]$. In addition, the terms Ψ_0 , Ψ_D , and q_0 correspond to the initial condition, Dirichlet, and Neumann boundary conditions, respectively. The term $\partial\Omega$ denotes the boundary of Ω , and Γ_D and Γ_N refer to the Dirichlet and Neumann boundaries, respectively. The outward unit normal vector to the boundary Γ_N is represented by $\mathbf{n}$.

For the transport equation (5.5), the following initial and boundary conditions are used:

$$\begin{cases} c(0, \mathbf{x}) = c_0(\mathbf{x}) & \text{in } \Omega, \\ c = c_D & \text{in } I \times \Gamma_D \subset I \times \partial\Omega, \\ -(\theta \mathbf{D} \nabla c - \gamma c \mathbf{q}) \cdot \mathbf{n} = \bar{c}_0 q_0 & \text{in } I \times \Gamma_N \subset I \times \partial\Omega, \end{cases} \quad (4.11)$$

where c_0 and c_D denote the initial and Dirichlet boundary conditions. When $\gamma = 0$, the third equation in (5.11) represents a second-type (Neumann) boundary condition, and when $\gamma = 1$, that same equation represents a third-type (Cauchy) boundary condition. In addition, $\bar{c}_0$ indicates the concentration of the applied solute and q_0 is the specific discharge. It is essential to specify the initial and boundary conditions for each solute when dealing with more than one.

4.3 Methods

4.3.1 Feedforward neural networks: FNNs

Here, we provide an overview of fully connected feedforward neural networks (FNNs). An FNN predicts an output $\hat{y}$ for an input $\tilde{x}$ through linear transformations followed by a nonlinear activation function σ . It consists of an input layer, one or multiple hidden layers with neurons applying transformations, and an output layer producing $\hat{y}$. The forward pass of an FNN is expressed as:

$$\begin{cases} y_0 = \tilde{x}, \\ y_l = \sigma(W_l y_{l-1} + b_l), \quad \text{for } l = 1, \dots, L, \\ \hat{y} = o(y_L), \end{cases} \quad (4.12)$$

where L is the number of hidden layers, W_l and b_l are the weights and biases, σ is the activation function, and o is the output activation.

In regression tasks, the network parameters $\Theta = (W_l, b_l)_l$ are optimized by minimizing the empirical loss function:

$$J_d(\Theta) = \frac{1}{N_d} \sum_{i=1}^{N_d} (y_i - \hat{y}_i)^2, \quad (4.13)$$

where N_d is the number of labeled data points y_i .

The activation function can be made adaptive during training to improve learning (Jagtap *et al.*, 2020). This is done by introducing a scale factor s and a learnable parameter α , modifying the activation into the form $\sigma(s\alpha_l y_l)$, where α_l is specific to each layer. For example, in $\tanh(s\alpha x)$, increasing $s\alpha$ reduces the width of the linear region, allowing the network to capture more complex patterns (see Figure 4.1). However, if $s\alpha$ becomes too large, it can cause exploding or vanishing gradients, making the training unstable. To avoid this, careful initialization of $s\alpha$ is necessary. With adaptive activation, the network parameters become $\Theta = (W_l, b_l, \alpha_l)_l$. We refer to this method as AA throughout the paper.

4.3.2 Physics-informed neural networks

In a PINN model (Raissi *et al.*, 2019), the inputs are time-space collocation points sampled from the domain $I \times \Omega$ (see Figure 4.2), and the output corresponds to the solution of the target PDEs. To effectively approximate the solution of the coupled model of water flow and solute transport, the network is trained not only to capture the data-driven aspects, but also to respect the physical principles encoded in the equations, including their initial and boundary conditions. To achieve this, a set of collocation points from the time-space domain $I \times \Omega$, must be sampled. If we randomly select a set of points $(t_i^r, \mathbf{x}_i^r) \in (0, T] \times \tilde{\Omega}$, where $\tilde{\Omega} = \Omega \setminus \partial\Omega$, these points are referred

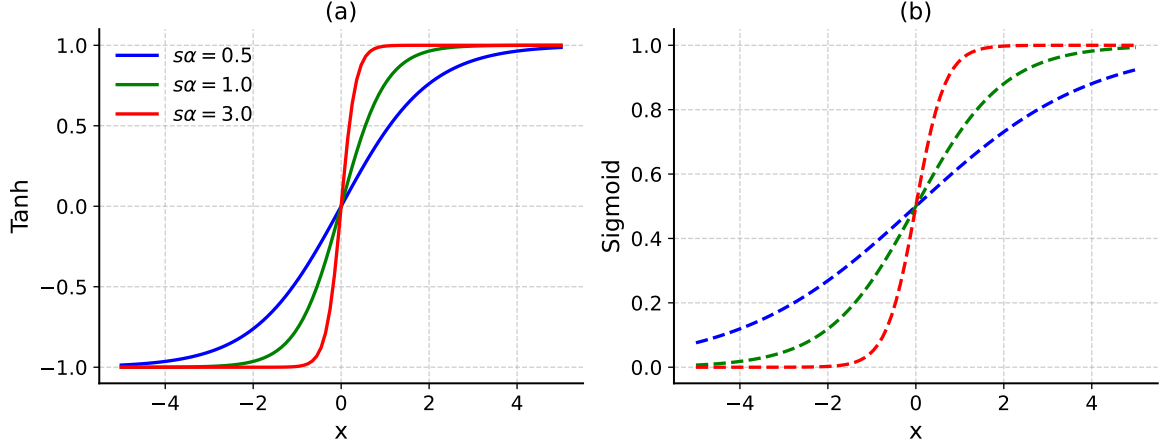


Figure 4.1 Adaptive activation function. Hyperbolic tangent (a) and sigmoid (b) activation functions for various values of $s\alpha$.

to as residual points. By feeding these points into the network, the output provides predictions for the solution of the Richards equation and of the solute transport model $(\hat{\Psi}, \hat{c})$ at the specified points $(t_i^r, \mathbf{x}_i^r)$. To enforce satisfaction of the coupled model, automatic differentiation is applied to derive the following residual functions:

$$\begin{cases} \hat{f}_R(t_i^r, \mathbf{x}_i^r) = \frac{\partial \hat{\theta}}{\partial t} - \nabla \cdot \left[\hat{K} \nabla (\hat{\Psi} + z) \right], \\ \hat{f}_C(t_i^r, \mathbf{x}_i^r) = \frac{\partial \hat{R} \hat{\theta} \hat{c}}{\partial t} - \nabla \cdot \left(\hat{\theta} \hat{\mathbf{D}} \nabla \hat{c} - \hat{c} \hat{\mathbf{q}} \right), \end{cases} \quad (4.14)$$

where the functions $\hat{\theta}$, $\hat{K}$, $\hat{\mathbf{q}}$, and $\hat{\mathbf{D}}$ are computed according to the relations given in equations (8.4a)–(8.4b).

On the other hand, to sample initial and boundary training points, we can also randomly select points $(0, \mathbf{x}_i^{ic}) \in \{0\} \times \Omega$ and $(t_i^b, \mathbf{x}_i^b) \in (0, T] \times \partial\Omega$. The values $(t_i^b, \mathbf{x}_i^b)$ can belong to the Dirichlet boundary, denoted as $(t_i^D, \mathbf{x}_i^D)$, or to the Neumann-Cauchy boundaries, denoted as $(t_i^N, \mathbf{x}_i^N)$.

After supplying the residual, initial, and boundary collocation points to the FNNs, the following loss functions can be derived:

$$J_r(\Theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \left(\frac{\hat{f}_R(t_i^r, \mathbf{x}_i^r)}{\tau_{f_R}} \right)^2 + \left(\frac{\hat{f}_C(t_i^r, \mathbf{x}_i^r)}{\tau_{f_C}} \right)^2, \quad (4.15a)$$

$$J_b^D(\Theta) = \frac{1}{N_b^D} \sum_{i=1}^{N_b^D} \left(\frac{\Psi(t_i^D, \mathbf{x}_i^D) - \hat{\Psi}(t_i^D, \mathbf{x}_i^D)}{\tau_\Psi} \right)^2 + \left(\frac{c(t_i^D, \mathbf{x}_i^D) - \hat{c}(t_i^D, \mathbf{x}_i^D)}{\tau_C} \right)^2, \quad (4.15b)$$

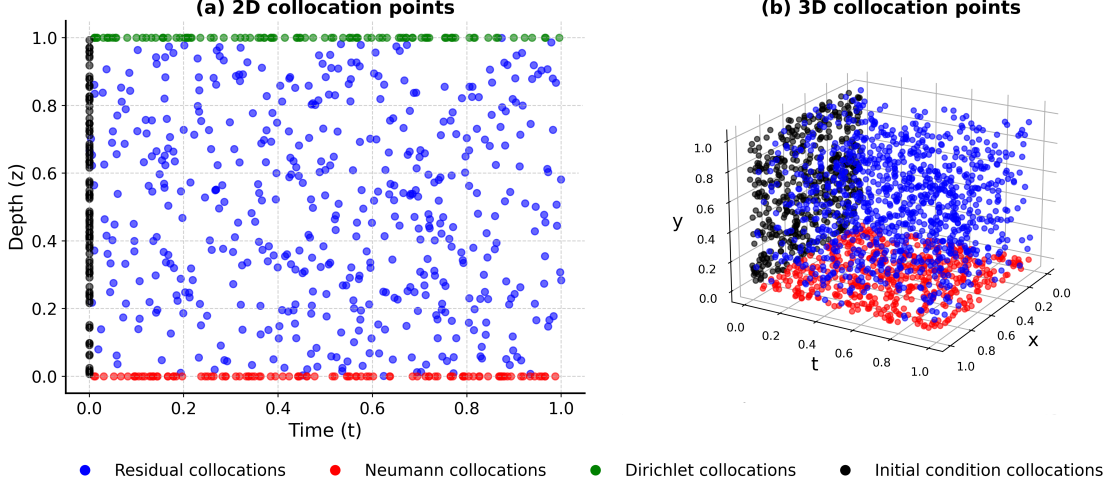


Figure 4.2 Example of 2D time-space collocation points (a) and 3D time-space collocation points (b).

$$J_b^N(\Theta) = \frac{1}{N_b^N} \sum_{i=1}^{N_b^N} \left(\frac{\mathcal{N}_\Psi(t_i^N, \mathbf{x}_i^N) - \hat{\mathcal{N}}_\Psi(t_i^N, \mathbf{x}_i^N)}{\tau_{\mathcal{N}_\Psi}} \right)^2 + \left(\frac{\mathcal{N}_c(t_i^N, \mathbf{x}_i^N) - \hat{\mathcal{N}}_c(t_i^N, \mathbf{x}_i^N)}{\tau_{\mathcal{N}_c}} \right)^2, \quad (4.15c)$$

$$J_b(\Theta) = J_b^D(\Theta) + J_b^N(\Theta), \quad (4.15d)$$

$$J_{ic}(\Theta) = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} \left(\frac{\Psi(0, \mathbf{x}_i^{ic}) - \hat{\Psi}(0, \mathbf{x}_i^{ic})}{\tau_\Psi} \right)^2 + \left(\frac{c(0, \mathbf{x}_i^{ic}) - \hat{c}(0, \mathbf{x}_i^{ic})}{\tau_C} \right)^2, \quad (4.15e)$$

where J_r , J_b^D , J_b^N , and J_{ic} represent the residual, Dirichlet boundary, Neumann boundary, and initial condition losses, respectively. Further, N_r , N_b^D , N_b^N , and N_{ic} denote the total number of residual, Dirichlet boundary, Neumann-Cauchy boundary, and initial collocation points, while $\mathcal{N}$ denotes the boundary operator defined by the Neumann-Cauchy boundary conditions. The terms τ_{f_R} , τ_{f_C} , τ_Ψ , τ_C , $\tau_{\mathcal{N}_\Psi}$, and $\tau_{\mathcal{N}_c}$ are the characteristic scaling factors introduced to ensure a non-dimensional form for all loss components, preventing imbalance due to differing units. It is worth noting that other techniques, such as the exact imposing with distance functions methodology developed in (Sukumar and Srivastava, 2022), can also be employed for enforcing initial and boundary conditions. However, in this study, the approach of using a soft manner, as described in equations (4.15b), (4.15c), and (4.15e), has been adopted.

The loss functions described above can be adapted for scenarios involving multiple solute transport equations by aggregating the losses for each solute component.

Finally, the total loss of a PINN solver can be defined as follows:

$$J(\Theta) = \alpha_d J_d(\Theta) + \alpha_r J_r(\Theta) + \alpha_b J_b(\Theta) + \alpha_{ic} J_{ic}(\Theta), \quad (4.16)$$

where α_d , α_r , α_b , and α_{ic} are weights assigned to the measured data, residual, boundary condition, and initial condition losses, respectively. Determining these weights can be achieved through a manual process involving trial and error, or they can be updated dynamically during training, as proposed in (Chen *et al.*, 2023b) for the Richards equation. Here, the selection of weight parameters was conducted through a trial-and-error procedure and is dependent on the specific test. The impact of these weight parameters on the accuracy of the proposed PINN solvers is discussed in Appendix 6.6. It is clear that the resulting solution fully satisfies the coupled equation and its corresponding initial and boundary conditions as the loss function J approaches zero. In this study, we propose and investigate three PINN formulations to address the water flow and solute transport model in unsaturated soils, which will be discussed in detail in the following subsections. We assume that there are n_s solute species under investigation.

4.3.3 Single PINN solver

To solve the coupled model of water flow and solute transport, our initial strategy revolves around the training of a unified neural network $\mathcal{N}(\Theta)$, where Θ denotes the trainable parameters. The neural network $\mathcal{N}(\Theta)$ takes time-space coordinates $(t, \mathbf{x})$ as its input and produces the matric head Ψ and solute concentrations c_i for $i = 1, \dots, n_s$, as its output. Therefore, the output layer of $\mathcal{N}(\Theta)$ has a dimension of $1 + n_s$ and the input dimension is either 2 (resp. 3) for 1D (resp. 2D) spatial problem. It should be mentioned that a comparable methodology was utilized in (Raissi *et al.*, 2019) to address the Navier–Stokes equation.

The hyperbolic tangent function is used as the activation function for all neurons in each layer. The output activation is set to $-\exp(x)$ specifically for the neuron corresponding to the predicted matric head. This choice is made to enforce the negativity of the matric head (Bandai and Ghezzehei, 2021). For the remaining output functions representing concentrations, the identity function ($\sigma(x) = x$) is used. This approach is referred to as (vanilla) PINNs. The total loss function for the PINN solver is defined by equation (5.16). Refer to Algorithm 1 in Appendix 6.6 for a detailed description of the PINN solver, and see Figure 4.3 for its conceptual representation. The architectures of the PINN solver for each numerical test are shown in Table 4.2.

4.3.4 Multiphysics-informed neural networks: MPINN solver

A second strategy is the multiphysics-informed neural network (MPINN) approach (Tartakovsky *et al.*, 2020), where multiple coupled PINNs are trained simultaneously. Each PINN is designed to approximate the solution to a specific governing PDE, such as the Richards equation or solute transport models. Specifically, $1+n_s$ neural networks are required: $\mathcal{N}(\Theta_\Psi)$ for the matric head and $\mathcal{N}(\Theta_{c_i})$ for the solute concentration of

solute i . The input to each neural network consists of time-space coordinates $(t, \mathbf{x})$, and the output layer of each PINN has a dimension of 1, corresponding to the solution that this PINN approximates. Each PINN has its own residual, data, initial, and boundary condition losses. The PINNs communicate through the total loss function defined in (5.16), which is constructed by summing the losses for all the governing PDEs. Previous studies have used this methodology to solve 2D coupled water flow and pore water electrical conductivity (Haruzi and Moreno, 2023a), and estimate spatially varying hydraulic conductivity, hydraulic head, and concentration fields from sparse measurements (He *et al.*, 2020).

We used the same activation functions as those employed in the PINN solver. Refer to Algorithm 2 in Appendix 6.6 for a detailed description of the MPINN solver, and see Figure 4.3 for its conceptual representation. The architecture of the MPINN solver depends on the specific test case, and Table 4.2 provides details for each numerical test.

It is important to note that the primary difference between PINN and MPINN lies in the number of neural networks used. In the PINN approach, a single neural network is employed to predict all the variables (matric head and species concentrations), whereas in the MPINN approach, a distinct neural network is assigned to each variable. This results in an increased computational cost for MPINN, as the number of trainable parameters grows proportionally with the number of predicted variables.

4.3.5 Sequential physics-informed neural network: S-PINN solver

This solver is based on the principle that the solute transport equation does not affect the solution of the Richards equation (Nielsen *et al.*, 1986). It can also be applied to cases where species in the upstream part of the transport chain do not depend on species in the downstream part, allowing for the independent solution of each species equation (e.g., the nitrogen chain, as demonstrated in test case B). Therefore, we aim to solve the coupled system of water flow and solute transport sequentially. In sequential training, the process is divided into multiple steps, with each step using a different loss function to guide the corresponding PINN solver.

The first step is to solve the Richards equation using a PINN solver, denoted as $\mathcal{N}(\Theta_\Psi)$. After training this solver, the outputs consist of water content, matric head, and water flux. These outputs are saved to be used later for all solute transport PDEs.

Next, the first solute transport PDE is solved using a PINN model $\mathcal{N}(\Theta_{c_i})$. If a decay chain is considered, such as the nitrogen chain, this solver will handle the first solute species. The output from this solver is then used to derive the necessary reaction term for the next solute species' PDE. This process continues sequentially until the last solute species in the decay chain is addressed.

Each solver uses a different set of trainable parameters, and there is no requirement for the neural networks to have the same architecture. This approach is referred to as the S-PINN solver. The same activation functions used in other solvers are adopted

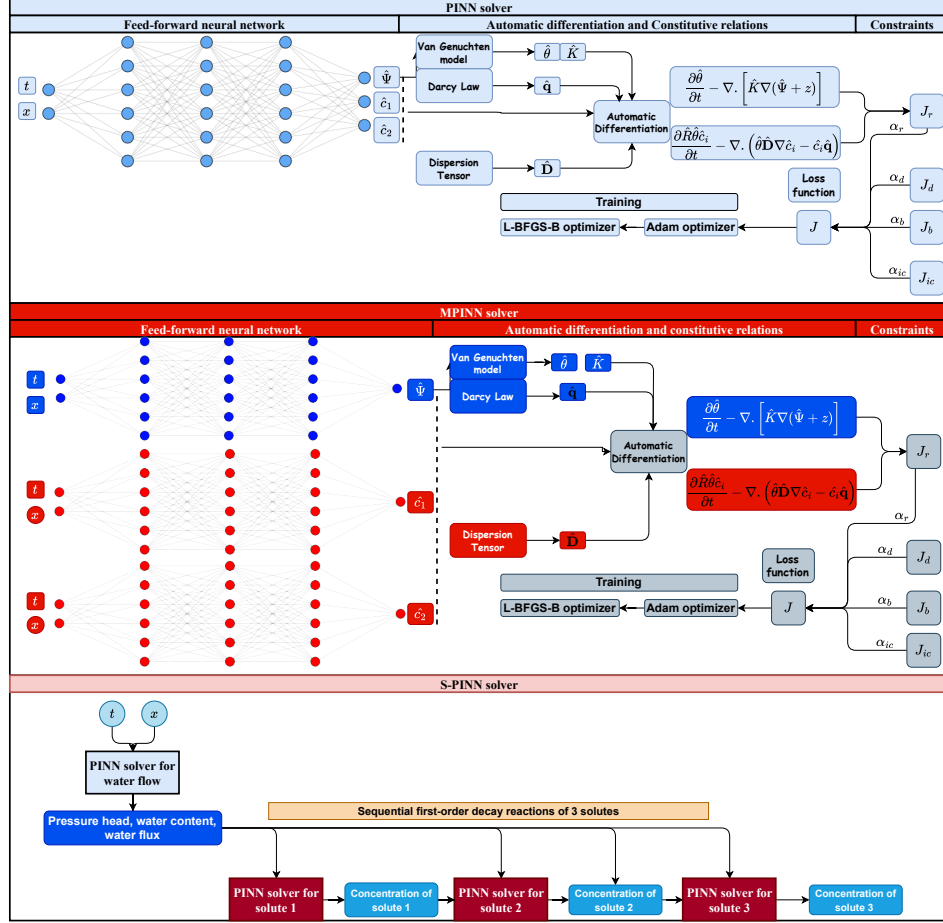


Figure 4.3 Schematic representation of the proposed PINN solvers for the coupled water flow and solute transport model. For notations, refer to Section 5.4.

here. The details of this approach are provided in Algorithm 3 in Appendix 6.6 and illustrated in Figure 4.3.

It is important to note that the S-PINN solver differs significantly from previously proposed PINN approaches. The S-PINN solver tackles the system sequentially, while other methods attempt to solve the entire system simultaneously. Consequently, this sequential approach may incur a higher computational cost. This solver can also be viewed as a variant of the MPINN solver, where at each step, only one PINN is trained while the others have their parameters frozen and their losses set to zero. At each subsequent step, the next PINN solver is activated to address the corresponding PDE, utilizing the necessary outputs from the previously trained solvers.

4.3.6 Algorithm implementation

The implementation of all PINN solvers was performed using the open-source software library TensorFlow (Abadi *et al.*, 2016). As previously indicated, the solver’s architecture is test case-dependent, and a trial-and-error procedure was used to determine the best combination for each numerical test. The impact of the solver architecture on its accuracy is outlined in Appendix 6.6. The Xavier algorithm (Glorot and Bengio, 2010a) was used for initializing the weights of the PINN solvers. At the start of the training stage, this approach prevents the activation function from becoming saturated (Glorot and Bengio, 2010a). Additionally, the bias parameters (b) were initialized to zero. When employing an adaptive activation function, the learning parameter α was initially set to 0.05 and the scale factor s to 20.

To solve the optimization problem posed by the objective function J , various optimization algorithms can be employed. Notably, the Adam optimizer (Kingma and Ba, 2014), a stochastic gradient descent variant widely applied in deep learning algorithms, and L-BFGS (Byrd *et al.*, 1995), belong to the quasi-Newton methods family. In this study, a two-step training technique was implemented, utilizing both these optimizers sequentially, beginning with the Adam and followed by the L-BFGS. The impact of this optimization scheme on the accuracy and training efficiency of the PINN solvers is discussed in Appendix 6.6. We used the Adam optimizer with exponential decay (initial learning rate 10^{-3} , decay rate 0.9, decay step 1000). For L-BFGS, the settings were: $\text{ftol} = 10^{-10}$, $\text{gtol} = 10^{-8}$, $\text{maxcor} = 50$, $\text{maxls} = 50$, $\text{maxiter} = 50,000$, and $\text{maxfun} = 50,000$, with other parameters left at default. To enhance training, a mini-batching technique was utilized for the Adam optimizer, whereas a full-batch approach was adopted for the L-BFGS optimizer.

The numerical experiments in this study were conducted using an NVIDIA V100 GPU. The source code is available on the GitHub repository:

<https://github.com/Hamza-Kamil/PINN-MPINN-Water-Solute>.

4.4 Numerical experiments

This Section presents several numerical tests conducted to evaluate the performance of the proposed PINN solvers. In test case A, we address the coupled model of water flow and solute transport in a 1D soil. Our evaluation focuses on assessing key properties of the three solvers, with a specific emphasis on accuracy and the preservation of mass within the spatial domain. In the second numerical test, we consider a system of PDEs describing the nitrogen species transport. This test case is bifurcated into two problems: the forward problem and the inverse problem. The forward problem involves studying the system’s behavior under well-defined parameters, including initial/boundary conditions and soil and solute parameters, while the inverse problem revolves around deducing the model parameters based on observed data. In the final two numerical tests, simulations are conducted in a 2D soil. The first of these involves

a numerical simulation of uniform water and solute infiltration. In the last test, we assess the accuracy of the accurate PINN solver by comparing it with real experimental data. Refer to Table 4.1 for an overview of the numerical tests presented in this paper.

Table 4.1 Summary of numerical tests in this paper.

Test case	Details
A	Coupled model of water flow and solute transport in a 1D soil.
B	System of PDEs governing the transport of nitrogen species, bifurcated into forward and inverse problems.
C	Numerical simulation in a 2D soil, involving simulation of uniform water and solute infiltration.
D	Accuracy assessment of the S-PINN solver by comparing it with real experimental data in a 2D soil.

4.4.1 Test case A: 1D coupled water flow and solute transport

In this numerical test, we evaluate the performance of the proposed PINN solvers for a 1D coupled water flow and solute transport model. HYDRUS-1D software ([Šimůnek et al., 2005](#)) is used to provide a reference solution. To quantify the error between the predicted and the reference solution, the following relative L^2 -error in the time-space domain $[0, T] \times [0, L]$ is employed:

$$\tau^u = \frac{\sum_{t \in [0, T]} \sum_{z \in [0, L]} (\hat{u}(t, z) - u(t, z))^2}{\sum_{t \in [0, T]} \sum_{z \in [0, L]} u(t, z)^2}, \quad (4.17)$$

where T and L are the final time of the simulation and the soil column depth, respectively, $\hat{u}$ and u are the predicted and reference solutions, respectively.

We considered a loamy soil with the following parameters:

$$K_s = 0.2496 \text{ m/day}, \alpha_v = 3.6 \text{ m}^{-1}, \theta_s = 0.43, \theta_r = 0.078, n_v = 1.56, L = 1 \text{ m}. \quad (4.18)$$

In the water flow equation, the initial matric head value was set to -1 m, the soil surface had a water flux of $\mathbf{q} = -0.05$ m/day, and a free drainage boundary condition was applied at the bottom of the soil $\left(\frac{\partial \Psi}{\partial z} \Big|_{z=L} = 0 \right)$. For the solute transport equation, the soil was assumed to be free from any solute species $c(t = 0, z) = 0$ mmol/m³. A Cauchy boundary condition was applied at the soil surface, with a constant solute feeding of $\bar{c}_0 = 50$ mmol/m³. A zero-gradient boundary condition was applied at the bottom of the soil column $\left(\frac{\partial c}{\partial z} \Big|_{z=L} = 0 \right)$. We assumed there was no interaction between the solid and the liquid soil phases, and so the retardation factor was set to $R = 1$. The molecular diffusion was neglected ($D_w = 0$ m²/day) and the longitudinal dispersivity was set to $D_L = 0.1$ m.

We simulate the infiltration of water and solute transport up to the final time $T = 1$ day, using parameters for each solver as presented in Table 4.2. We set the weights constant in the loss function (5.16) for the PINN and MPINN solvers as follows: $\alpha_d = 0$, $\alpha_r = 1$, $\alpha_b = 10$, $\alpha_{ic} = 10$, while for the S-PINN solver, we set $\alpha_d = 0$, $\alpha_r = 1$, $\alpha_b = 100$, and $\alpha_{ic} = 1$. In this selection, we assumed that no available data is presented for training the three solvers ($\alpha_d = 0$). We forced the initial and boundary conditions by using high weights to improve the training of the three solvers. {The scaling factors are set as follows: $\tau_\Psi = \tau_{f_R} = \tau_{f_C} = \tau_{N_\Psi} = 1$, $\tau_C = 50$, $\tau_{N_c} = 10$. Figure 4.4 depicts the profiles, absolute errors, and cross-sections of the matric head Ψ , water flux $\mathbf{q}$, and solute concentration c predictions obtained from the MPINN solver with the adaptive activation technique. Good agreement was found for all the solutions ($\Psi, \mathbf{q}, c$) in comparison with the reference solution. Figure 4.4 shows that at the start of the simulation, there are noticeable inaccuracies close to the soil surface, which are caused by a high flux entering the domain. To effectively tackle this problem, it is advised to exactly impose the boundary conditions via appropriate distance functions to the boundaries (Sukumar and Srivastava, 2022). Additionally, we can observe high errors along the sharp front propagated over time, particularly where the matric head exhibits steep gradients.

To compare the accuracy of the three solvers, we computed the relative L^2 -error (5.20). Table 4.3 presents the relative L^2 -error obtained for each solver and each predicted variable, as well as the final total and residual losses after the training stage. These results show that using the adaptive activation function technique improves the accuracy and the training of all solvers. The S-PINN solver outperforms the other solvers in terms of accuracy and training efficiency, achieving less than 1% error for matric head, 2.25% for solute concentration, and 3% for water flux. By employing sequential training, the S-PINN solver achieves significantly lower losses for water content, reaching below 10^{-7} when utilizing adaptive activation techniques. Compared to other solvers, the S-PINN solver demonstrates much greater efficiency in modeling the coupled system of water and solute.

The evolution of the loss functions for all solvers is illustrated in Figure 4.5. The adaptive activation technique improves the training of all three solvers, with the lowest loss observed for the S-PINN solver, confirming the results obtained in Table 4.3.

For further comparison of the three solvers in this numerical test, we plot cross-sections of the matric head, solute concentration, and water flux in Figure 4.6. The results for predicting solute concentration were identical for all solvers, indicating no influence of employing an adaptive activation technique. The main differences were observed for the matric head and the water flux variables. As shown in Figure 4.6, a significant improvement was obtained when using an adaptive activation technique, especially for the PINN solver. S-PINN and MPINN solvers appear to be less sensitive to the adaptive activation technique. In contrast, the PINN solver without adaptive activation required more training epochs to accurately predict matric head and water flux

profiles. Even though there may not be a visible difference when using the adaptive activation technique for the S-PINN and MPINN solvers in Figure 4.6, Table 4.3 clearly demonstrates its significant impact and essential improvement across all solvers.

Ensuring accurate mass balance without introducing any additional artificial masses, whether in water or solute, is a fundamental assessment of the numerical solver’s reliability. To calculate the total mass of water and solute at time t , we used the following formulas:

$$T_w M(t) = \int_{\Omega} \theta(t, z) \, d\Omega, \quad T_s M(t) = \int_{\Omega} \theta(t, z) c(t, z) \, d\Omega, \quad (4.19)$$

where $T_w M(t)$ and $T_s M(t)$ represent the total water mass and total solute mass, respectively, at time t . The integrals defined in equation (4.19) are approximated using Simpson’s rule.

The evolution of water and solute masses is displayed in Figure 4.7. All solvers provided good results, and the adaptive technique helped to speed up the training of the PINN solver. Once again, the S-PINN solver outperforms others in terms of mass, resulting in an error of less than 5%.

The current investigation aims to assess the generalization capacity of the S-PINN solver. This entails monitoring the evolution of its total loss throughout each iteration, encompassing evaluations at both the training points and new unseen test points. We have generated 20,000 new residual collocation points, and 5,000 new initial and boundary collocation points. A visual representation of the outcomes can be found in Figure 4.8. The total loss continues to decrease during training for both the water and solute solvers, whereas the test loss exhibits a comparatively larger discrepancy. To effectively navigate the bias-variance trade-off, it’s recommended to conduct a comprehensive grid search across all hyperparameters in the S-PINN solver. This includes crucial parameters like learning rate, batch size, and the number of collocation points. Additionally, integrating a specialized regularization technique within the loss function can further refine model performance and generalization capabilities. Although we did not observe a U-shaped error curve commonly associated with deep learning solvers. This test case reveals that the S-PINN solver is more efficient in solving the coupled model of water and solute transport than the other architectures. Another advantage of all solvers is their ability to ensure the positivity of solute concentrations after a few epochs, although no theoretical guarantees exist. This criterion is essential for adhering to physical laws, such as the negativity of matric head and the positivity of concentration.

In the following numerical test, we aim to explore the PINN solvers in modeling the dynamics of multiple nitrogen species in unsaturated soil.

Test case A and B					
Residual points	IBC points	Hidden layers	Neurons	Adam epochs	L-BFGS max epochs
10000	1200	5	50	20000	50000
Test case C					
10000	2000	6	50	20000	50000
Test case D					
50000	12000	8	15	100000	50000

Table 4.2 Parameter configuration for training the PINN solvers across all examined test cases. IBC stands for initial and boundary conditions.

Table 4.3 Test case A - Relative L^2 -error for the PINN solvers with and without adaptive activation (AA). For the S-PINN solver, $\cdot/\cdot$ represents the total loss of the PINN water solver followed by the total loss of the PINN solute solver.

Solver	AA	τ^Ψ	τ^c	τ^q	J	J_r
MPINN	No	3.3e-2	2.32e-2	8.75e-2	3.35e-3	7.39e-4
MPINN	Yes	2.41e-2	2.5e-2	8.58e-2	4.56e-4	7.9e-05
PINN	No	2.63e-2	3.51e-2	9e-2	1.51e-4	2.71e-4
PINN	Yes	2.19e-2	3.42e-2	7.67e-2	7.79e-4	1.41e-4
S-PINN	No	1.6e-2	2.51e-2	3.58e-2	3.17e-7 / 7.7e-4	1.7e-7 / 5.9e-4
S-PINN	Yes	1.02e-2	2.25e-2	3.13e-2	9.3e-8 / 5e-4	5.9e-8 / 3.1e-4

4.4.2 Test case B: Nitrogen species transport

4.4.2.1 Forward problem

In this numerical test, we employed the PINN solvers to predict the dynamics of nitrogen species in an unsaturated soil column. A fully coupled system that represented the nitrogen species first-order decay chain was chosen. Equation (6.27) presents the governing PDEs for the full coupled system. This numerical test involves various solute concentrations and includes reaction terms. As with the previous test, HYDRUS-1D software was employed for comparison. In this test case, the soil parameters remained the same as those used in test 4.4.1. In addition, we retained the same initial and bottom boundary conditions for the water flow equation, while the upper boundary was adjusted to impose a constant head of -0.2 m. Ammonium NH_4^+ and nitrate NO_3^- were considered the initial nitrogen sources. Autotrophic bacteria nitrified the ammonium by a factor of μ_1 to produce nitrite NO_2^- . Subsequently, the same bacteria further nitrified the ammonium by a factor of μ_2 to produce nitrate NO_3^- . Finally, denitrification of nitrate occurred by a factor of μ_3 to produce dissolved nitrogen N_2 and N_2O (see Figure 5.1). To be consistent with equation (6.27), c_1 , c_2 , and c_3 denote

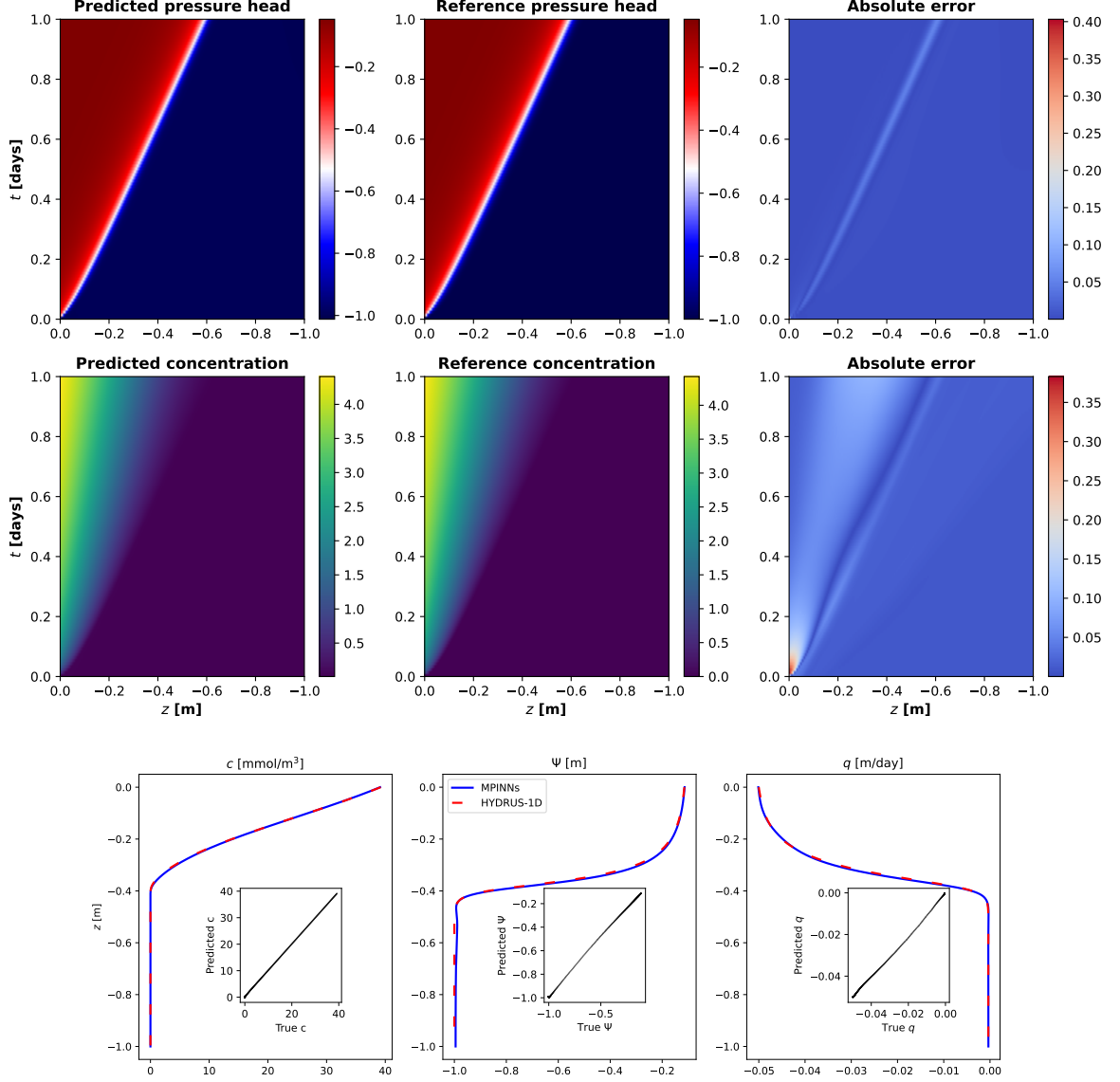


Figure 4.4 Test case A – Predictions of matric head Ψ and solute concentration c in the time-space domain $[0, 1] \times [0, 1]$ using MPINN with adaptive activation, along with the absolute error between the predicted and true profiles. Bottom: Predictions from MPINN of Ψ , c , and q at $T = 1$ day.

the concentrations of NH_4^+ , NO_2^- , and NO_3^- , respectively, and μ_1 , μ_2 , and μ_3 represent the first-order rate coefficients for the nitrification of NH_4^+ , the nitrification of NO_2^- , and the denitrification of NO_3^- , respectively. For simplicity, we omitted the nitrite species, as it rapidly nitrifies to nitrate (Kim *et al.*, 2008).

In this model, we focused solely on first-order degradation reactions, including nitrification, denitrification, and adsorption. Specifically, only ammonium was assumed to undergo linear adsorption to the soil solid phase, characterized by the retardation

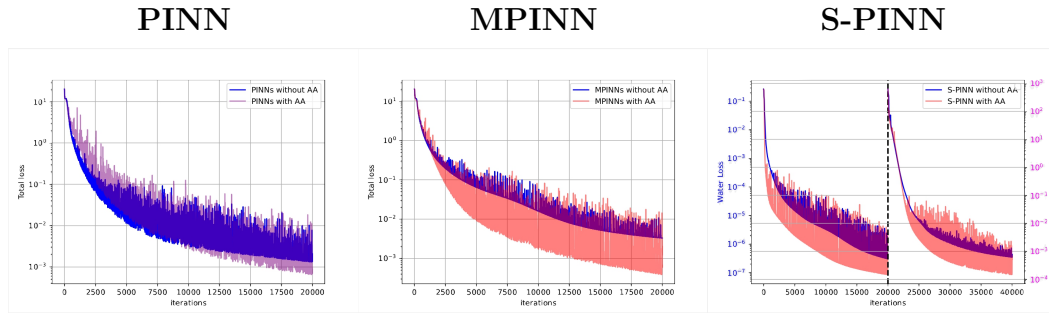


Figure 4.5 Test case A- Evolution of loss functions for the PINN solvers during Adam epochs.

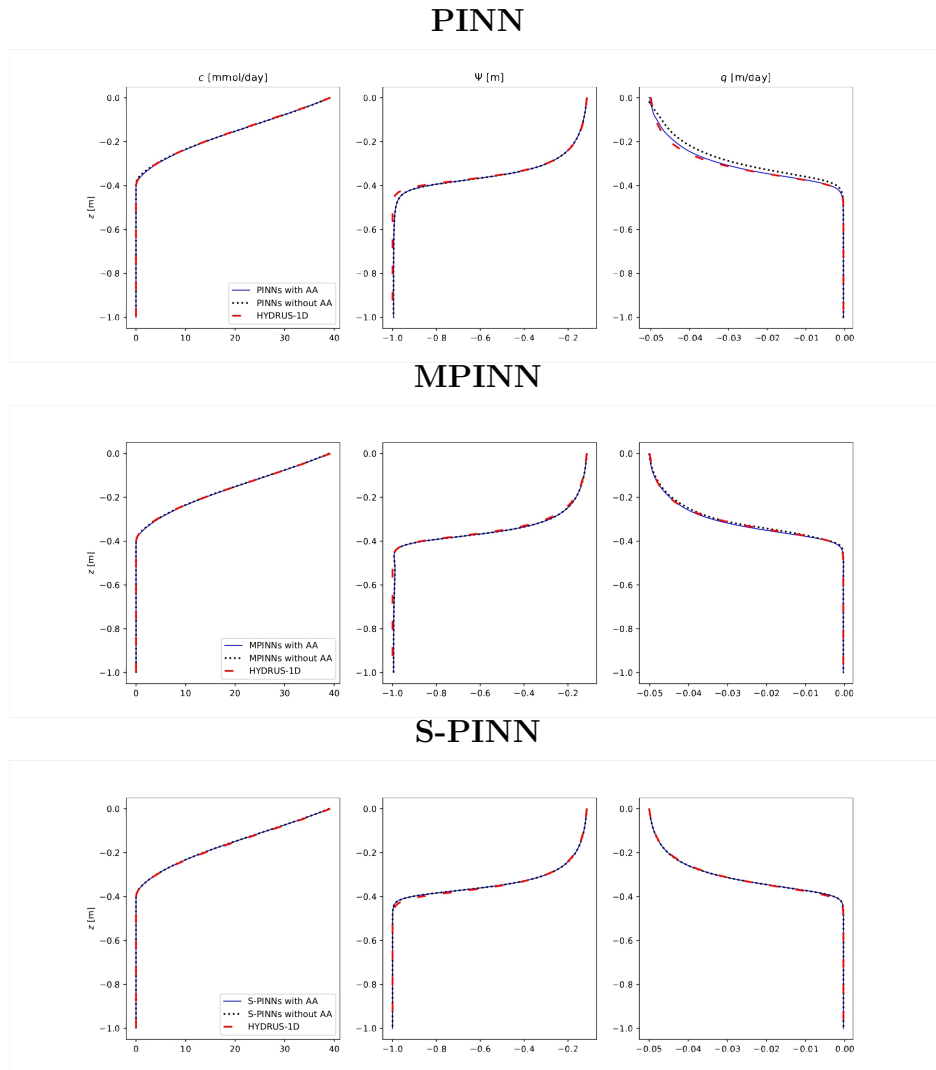
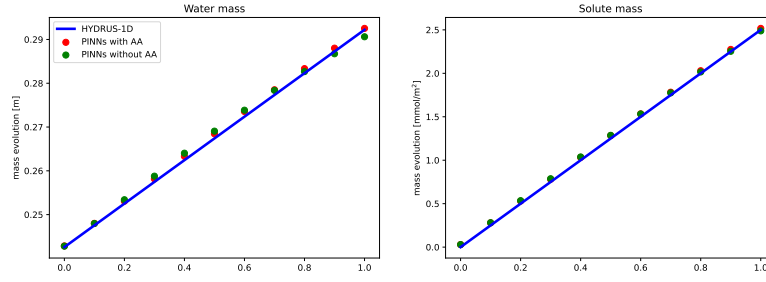
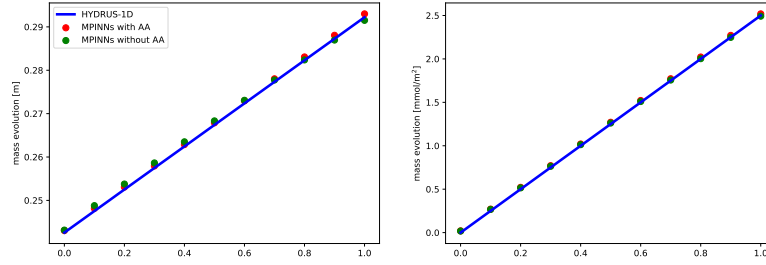


Figure 4.6 Test case A- The predicted matric head, solute concentration, and water flux obtained from various PINN solvers at time $T = 1$ day. The results are compared with those obtained from HYDRUS-1D.

PINN



MPINN



S-PINN

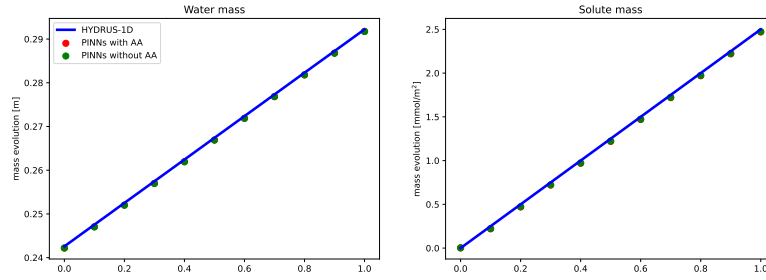


Figure 4.7 Test case A- Water and solute mass evolution as a function of time obtained from different PINN solvers compared with HYDRUS-1D.

factor R (we set $p = 1$). Additional terms can be introduced into the model as necessary. The initial state of the soil was considered free from any nitrogen species ($c_i = 0 \text{ g/m}^3$). Concentrations at the soil surface were fixed at 1 and 0.2 g/m^3 for ammonium and nitrate, respectively. Zero-concentration gradients were assumed on the bottom boundary for all nitrogen species.

The following parameters were used:

$$\begin{aligned}
 D_L &= 0.04 \text{ m}, \quad D_w = 0.288 \text{ cm}^2/\text{day}, \quad \rho = 1.6 \times 10^{-6} \text{ mg/l}, \\
 K_d &= 0.34 \times 10^{-6} \text{ mg/l}, \quad \mu_1 = 0.12 \text{ day}^{-1}, \quad \mu_3 = 0.048 \text{ day}^{-1}.
 \end{aligned} \tag{4.20}$$

All the parameters were obtained from (Bailey *et al.*, 2013; Berlin and Suresh Kumar,

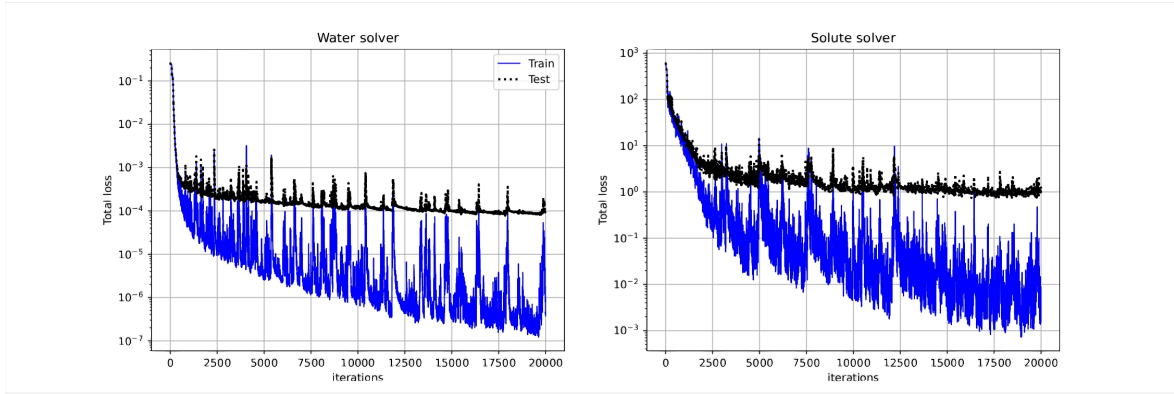


Figure 4.8 Test case A- Evolution of the total loss function during training and testing for the S-PINN solver with adaptive activation during Adam epochs.

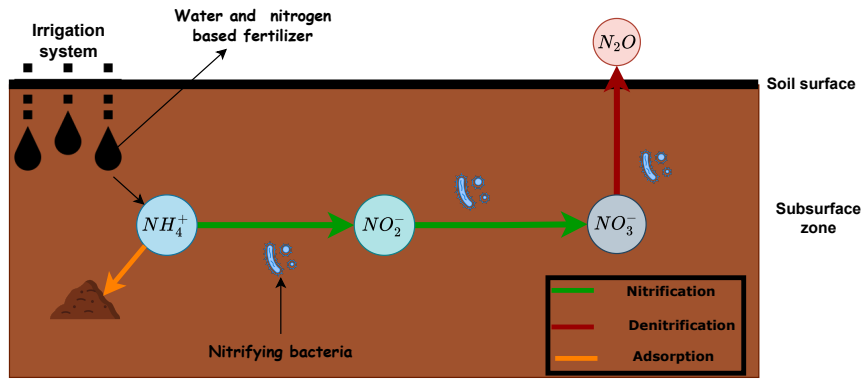


Figure 4.9 Test case B- Illustration of the nitrogen species first-order decay chain.

2018).

We solve this fully coupled system using the parameters for each solver as presented in Table 4.2. The weights in the loss function (5.16) are set as follows: $\alpha_d = 0$, $\alpha_r = 1$, $\alpha_b = 1$, and $\alpha_{ic} = 10$. Also in this numerical test, we suppose that no available data is presented for training all solvers ($\alpha_d = 0$). The scaling factors are set as follows: $\tau_\Psi = \tau_{f_R} = \tau_{f_{C_i}} = \tau_{N_\Psi} = \tau_{N_{C_i}} = 1$, $\tau_{C_i} = 50$, for $i = 1, 2, 3$.

The predicted matric head and concentrations of nitrogen species after one day are shown in Figure 5.8, which shows good agreement with HYDRUS-1D for all solvers. We omitted the MPINN solver results as they are the same as those of the PINN solver. It was observed that PINN and MPINN solvers initially focused on approximating the solution of one equation at a time. Specifically, they first approximate the water and the first concentration-governed equations within the first 500 Adam epochs (see Figure 4.11 for the case of the MPINN solver). Subsequently, the solvers proceeded to capture the other solute concentrations. This sequential approach is a common characteristic of numerical schemes in situations where the Richards equation needs

to be solved first to provide inputs (the water content and flux) for the other solute transport equations. The first solute concentrations are then solved to provide inputs for the subsequent equations, and so on. In general, the PINN and MPINN solvers face challenges in predicting the matric head due to the high nonlinear nature of the Richards equation, while they learn the linear transport equations more quickly. This numerical test illustrated a unique property of PINN solvers: they can capture the transport equation before fully realizing the Richards equation, which is the main source of their input data. Other research has found comparable results (Jagtap *et al.*, 2020). This characteristic is referred to as the spectral bias of neural networks (Rahaman *et al.*, 2019). Thus, this justifies the efficiency of S-PINN, which focuses on each equation individually, eliminating the need to tackle different types of PDEs simultaneously. Based on these two numerical tests, we conclude that S-PINN is the most recommended solver for modeling water flow and solute transport in unsaturated soils. In the next section, we challenge the PINN solvers to solve an inverse problem where we aim to predict the nitrification rate of ammonium.

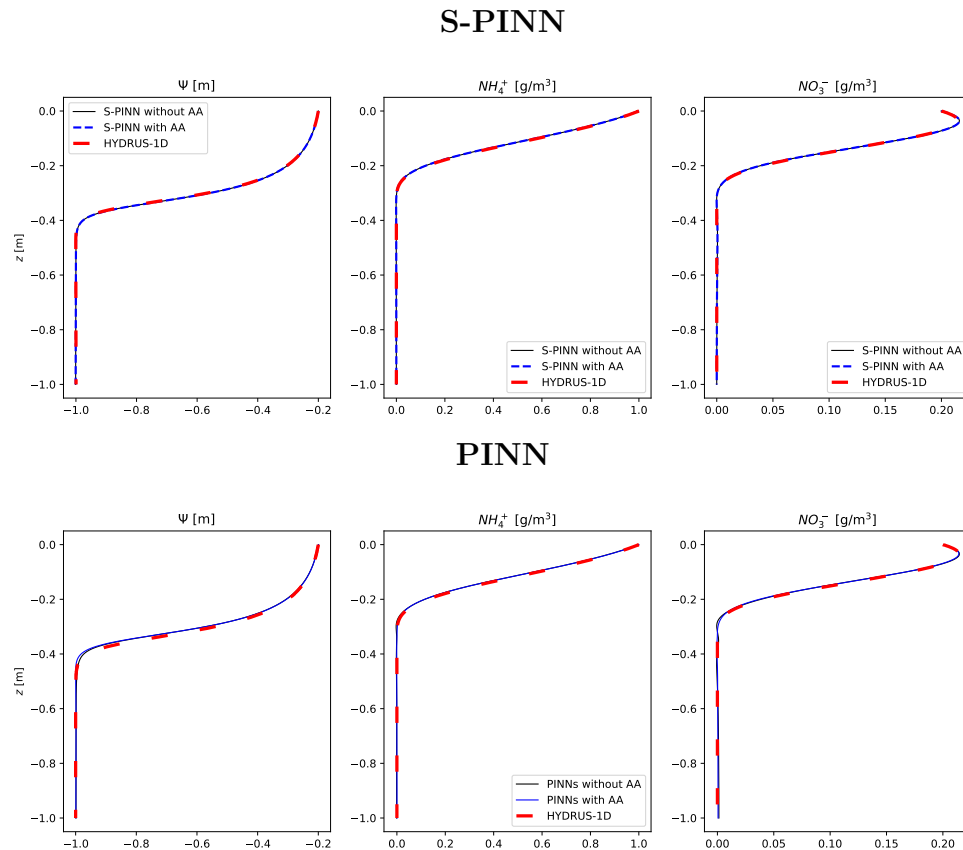


Figure 4.10 Test case B- Forward problem: Predicted matric head and nitrogen species concentrations obtained from various PINN solvers at time $T = 1$ day.

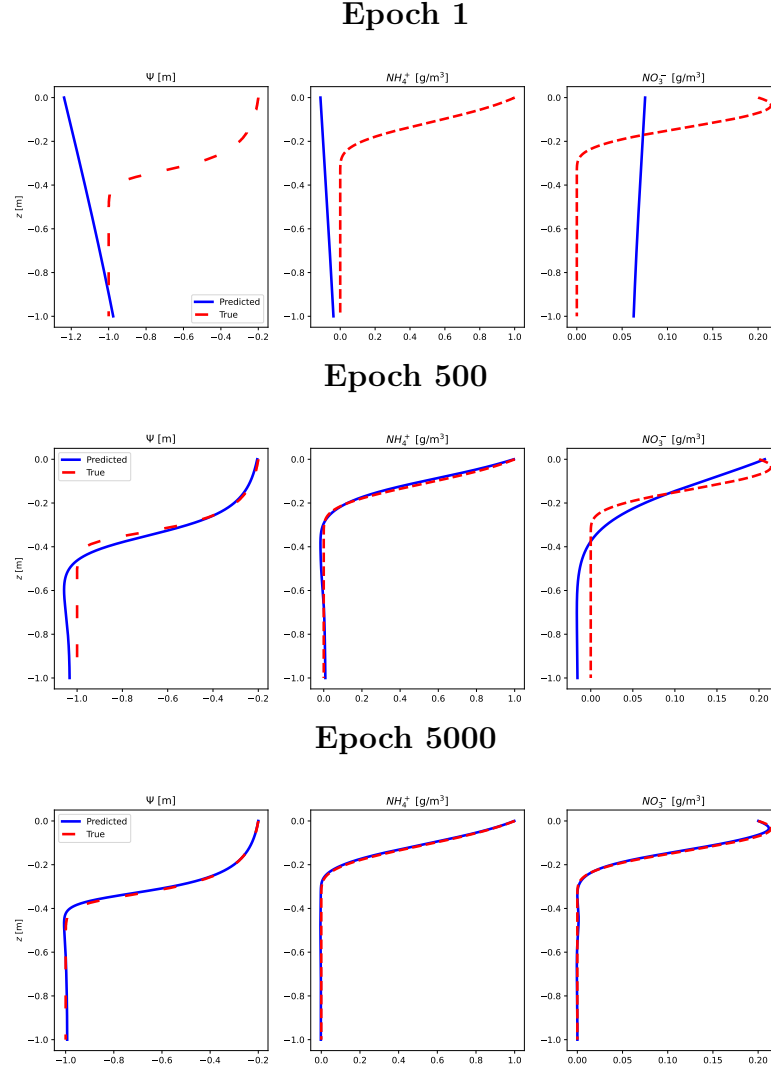


Figure 4.11 Test case B - Forward problem: Predicted matric head and nitrogen species concentrations using the MPINN solver with adaptive activation at different Adam epochs, at time $T = 1$ day.

4.4.2.2 Inverse problem: Ammonium nitrification rate estimation

In the second phase of the nitrogen species transport test, our goal is to identify the nitrification rate μ_1 . Calibrating this rate, along with other kinetics parameters, is a costly task. This inverse problem is more complex than the forward one. We exclusively focus on the transport of ammonium NH_4^+ and omit the other nitrogen species. In addition to the initial and boundary conditions, we also require the measured NH_4^+ data without the need for a measured matric head. To obtain this data, we utilized HYDRUS-1D, and collected measurements at times 0.25, 0.5, 0.75, and 1 day, corresponding to spatial points $z = -0.1, -0.2, -0.3, \dots, -0.9$ m. Consequently, we have a total of 36 measured concentrations of ammonium. These data points are

very sparse, as they do not sufficiently cover the computational time-space domain. It is important to note that we can omit the initial and boundary conditions for NH_4^+ transport if additional data is available. The weights in the loss function (5.16) are set as follows: $\alpha_d = \alpha_r = \alpha_b = 1$, $\alpha_{ic} = 10$. The scaling factors are set as follows: $\tau_\Psi = \tau_{f_R} = \tau_{f_{C_1}} = \tau_{\mathcal{N}_\Psi} = \tau_{\mathcal{N}_{C_1}} = 1$, $\tau_{C_1} = 50$.

The approach for identifying a PDE parameter using PINNs is straightforward. We incorporate the data loss into the global loss function for each PINN solver. Initially, the parameter μ_1 is randomly initialized within a predefined range ($\mu_1^{\min}, \mu_1^{\max}$). During training, μ_1 is treated as a trainable parameter and is updated at each epoch using the Adam optimization algorithm. This allows PINNs to simultaneously learn both the PDE parameters and the neural network weights by minimizing the constrained loss function.

We used the same architecture and collocation point parameters as in the forward problem (see Table 4.2). For the S-PINN solver, we set Adam epochs to 20,000 for the PINN water solver and 80,000 epochs for the PINN ammonium solver. For the other solvers, we set Adam epochs to 100,000. We justify using this number of epochs based on: 1) the sparsity of the data, and 2) the highly nonlinear and ill-posed nature of the problem. Our focus is on learning the parameter μ_1 alongside the desired solutions Ψ and c_1 using different PINN solvers.

Figure 4.12 presents the results for the matric head and the ammonium profiles at time $T = 1$ day using the S-PINN solver. The ammonium concentration is captured by the S-PINN solver despite the unknown parameter μ_1 . We did not observe any improvement in the predictions of S-PINN when using adaptive activation.

Table 4.4 shows the predicted value $\hat{\mu}_1$ of the nitrification rate μ_1 for each solver. The S-PINN solver with an adaptive activation technique demonstrated a lower error (15%) compared to the PINN and MPINN solvers. Once again, the S-PINN solver demonstrated its efficiency in solving this inverse problem, even with limited measurement data, confirming the results obtained in the forward problem.

In these 1D numerical tests, the S-PINN solver has demonstrated its superiority over the other PINN solvers, indicating that sequential training is the most accurate strategy. In the next sections, we utilize the S-PINN solver for 2D numerical tests to further evaluate its accuracy.

Table 4.4 Test case B - Inverse problem: Predicted value of the nitrification rate μ_1 in day^{-1} using three different PINN solvers with adaptive activation. Note that $\mu_1 = 0.12 \text{ day}^{-1}$.

Solver	Predicted value	Relative error (%)
PINN	0.079	34.16
MPINN	0.08	33.33
S-PINN	0.102	15

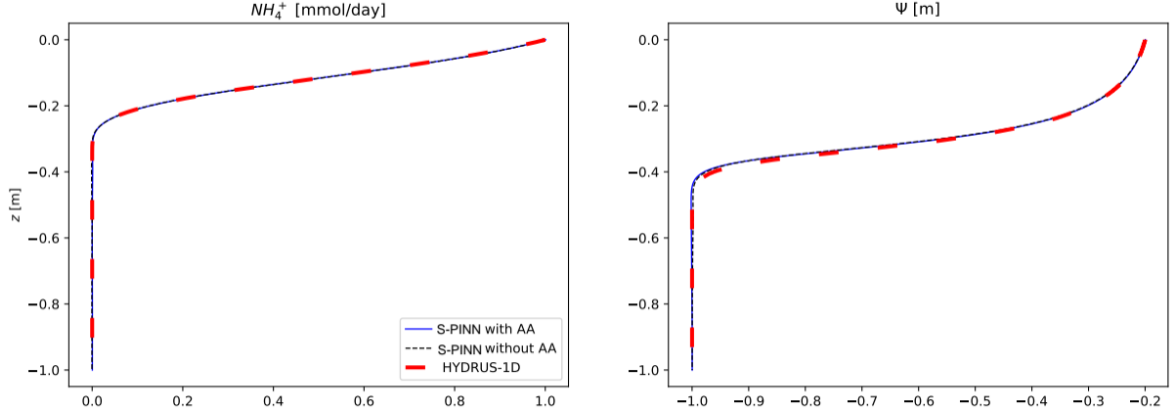


Figure 4.12 Test case B- Inverse problem: Predicted matric head and ammonium concentration obtained using the S-PINN solver at time $T = 1$ day.

4.4.3 Test case C: 2D water flow and solute transport under uniform boundary conditions

In this 2D test, we evaluate the S-PINN solver on a coupled model involving both x and y directions, which increases training complexity. We apply uniform Dirichlet conditions at the soil surface, free drainage at the bottom, and no-flux conditions on the lateral boundaries. Figure 4.13 presents a conceptual illustration. The soil parameters (5.12) are utilized, and no interaction between the solid and liquid phases of the soil was considered. The following solute parameters were used:

$$D_L = 0.5 \text{ m}, D_T = 0.1 \text{ m}, D_w = 0.0 \text{ m}^2/\text{day}. \quad (4.21)$$

For this 2D problem, the parameters given in Table 4.2 were used. We set the following weight constants: $\alpha_d = 0$, $\alpha_r = 1$, and, $\alpha_b = \alpha_{ic} = 10$. All scaling factors are set to unity.

In Figure 4.14, the results for the S-PINN solver with adaptive activation are depicted for the times 0.25 day, 0.5 day, 0.75 day, and 1 day. We used the finite element method (FEM) (Kamil *et al.*, 2024b) to solve the same problem to evaluate the accuracy of the S-PINN solver. The FEM was employed with a grid consisting of 100×100 elements and a time step of $\Delta t = 10^{-3}$ days.

The cross-sections at $x = 0$ of the S-PINN and FEM solutions are depicted in Figure 4.14. This 2D numerical test demonstrates the ability of the S-PINN solver to capture the 2D infiltration-transport problem. A major disadvantage of S-PINN is the CPU time required for training this 2D problem, which is comparatively longer. The FEM produced immediate results, in close to 30 seconds, while the S-PINN solver consumed more time during the training phase (6 minutes on a GPU). Even though, after training, the predictions are made in a fraction of a second. To address the limitation of long

CPU time in training, various techniques could be explored, such as incorporating specialized methods into the loss functions or relaxing the derivatives in the governing PDEs.

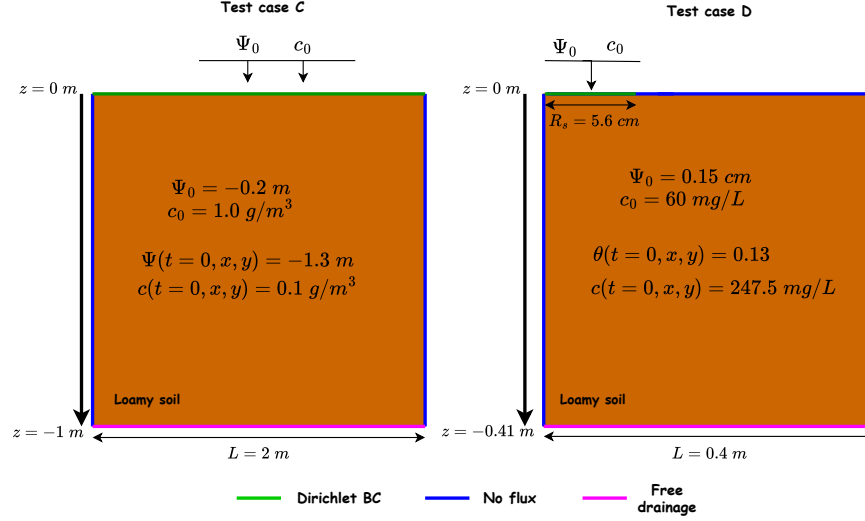


Figure 4.13 Water flow and solute transport 2D domain for test case C (left), and for test case D (right).

4.4.4 Test case D: 2D nitrate transport under nonuniform boundary conditions

In the final numerical test, our objective is to address a practical problem involving water infiltration coupled with the transport of nitrate species in a 2D loamy soil. This experiment was previously conducted and documented in (Li *et al.*, 2005), where various strategies for water and nitrate application were explored. The experiment took place within a cylindrical domain characterized by a radius of $L_x = 41 \text{ cm}$ and a depth of $L_z = 40 \text{ cm}$. Cylindrical coordinates were employed to model water flow and chemical transport under axisymmetrical conditions. In this context, the radial coordinate is represented by the variable x . We evaluate the performance of the S-PINN solver with the adaptive activation technique at simulating this 2D experiment. For this experiment, a loamy soil is considered, characterized by the following parameters:

$$K_s = 0.4704 \text{ m/day}, \alpha_v = 1.5 \text{ m}^{-1}, \theta_s = 0.41, \theta_r = 0.047, n_v = 1.48. \quad (4.22)$$

It is assumed that there is negligible interaction between the solid and liquid phases of the soil, and the following nitrate parameters are adopted:

$$D_L = 0.32 \text{ cm}, D_T = D_L/100 \text{ cm}, D_w = 0.09 \text{ cm}^2/\text{h}. \quad (4.23)$$

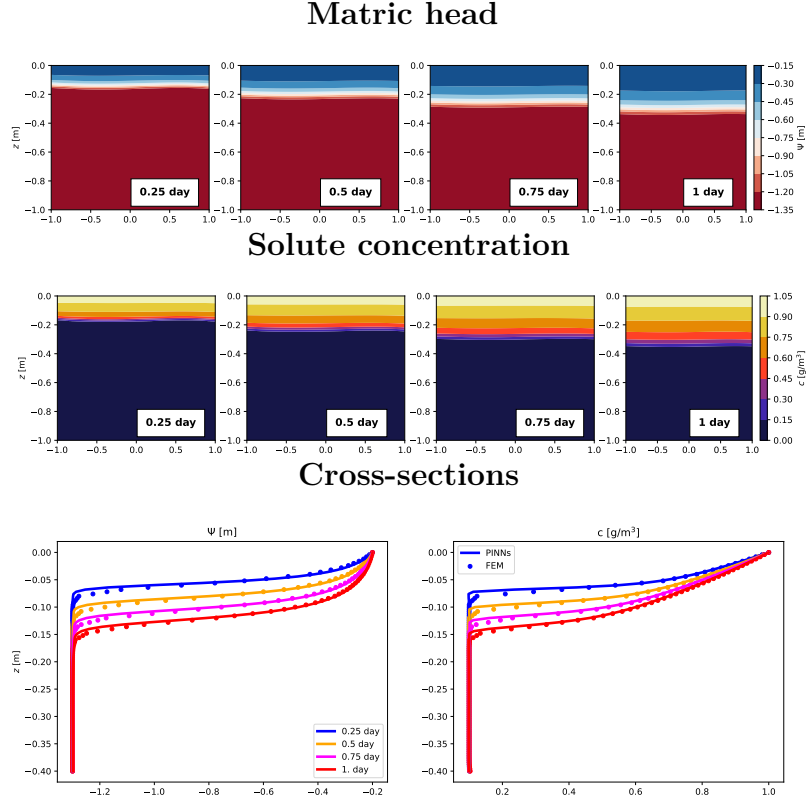


Figure 4.14 Test case C – Results obtained using the S-PINN solver with adaptive activation and the FEM. The first and second rows show the 2D temporal evolution of the matric head and solute concentration at snapshots taken at 0.25, 0.5, 0.75, and 1 day. The third row presents cross-sectional views of the matric head and solute concentration at different time points.

The associated initial and boundary conditions for this numerical test are illustrated in Figure 4.13. For the analytical form of the boundary conditions applied to the soil domain in the coupled water flow and nitrate transport model, refer to (Li *et al.*, 2005). To accommodate the axisymmetrical nature of the flow phenomena, we transformed the Richards equation to its cylindrical form, as employed by the authors in (Li *et al.*, 2005). In this experiment, water and nitrate were continuously applied from the left portion of the soil surface for a duration of 8 hours.

The architecture of the S-PINN solver and the size of the collocation points are outlined in Table 4.2. The weights in the loss function (5.16) are configured as follows: $\alpha_r = \alpha_b = \alpha_{ic} = 1$. Additionally, for this numerical test, we assume no available data is provided for training the S-PINN solver, and thus, we set $\alpha_d = 0$. All scaling factors are set to unity.

The 2D profiles of the water content and nitrate concentration are depicted in Figure 4.15, indicating realistic profiles. Since no analytical solution is available for this particular test case, we validate the S-PINN solver results by comparing them with ex-

perimental data provided in (Li *et al.*, 2005). The cross-section of the water content at fixed positions is presented in Figure 4.16. Our assessment indicates that the S-PINN solver with adaptive activation yields good results, showcasing a favorable comparison with measured data, as illustrated in Figure 4.16.

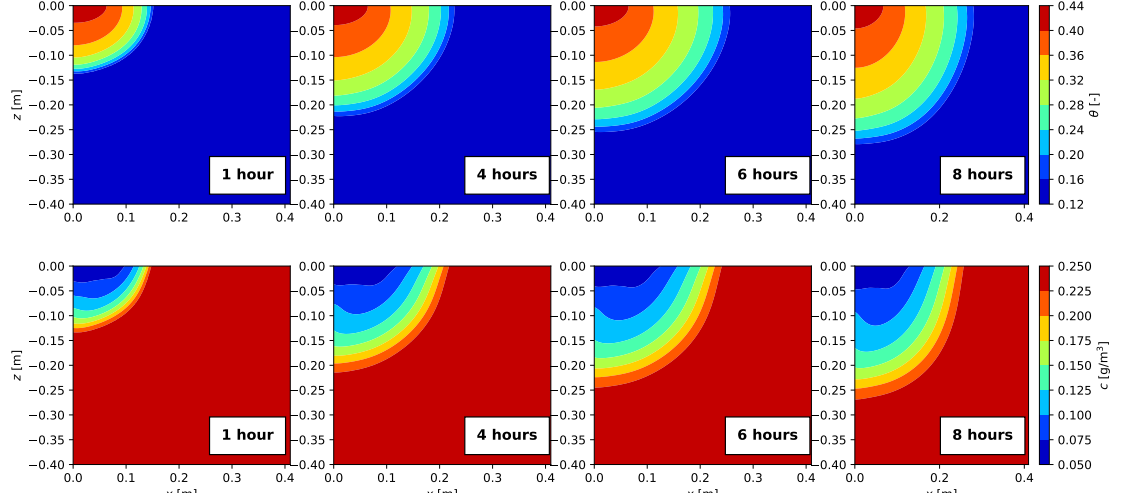


Figure 4.15 Test case D - Predicted 2D profiles of water content (top) and nitrate concentration (bottom) at various time snapshots using the S-PINN solver with adaptive activation

4.5 Limitations

This study explores the application of different PINN architectures for solving the coupled model of water flow and solute transport in unsaturated soils. We assume a lack of available data for the coupled model, except for the inverse problem, where scarce data were utilized to determine the kinematic ammonium nitrification rate. Therefore, in contrast to previous studies (Bandai and Ghezzehei, 2021; Chen *et al.*, 2023b; Depina *et al.*, 2022; Haruzi and Moreno, 2023a), the objective of this study was to assess the capability of the PINN approach in the forward problem of solving the highly non-linear Richards equation coupled with the transport of single or multi-species solutes. We demonstrated the efficiency and accuracy of the PINN approach, particularly when employing sequential training with adaptive activation functions to solve the model (referred to as S-PINN). One of the primary drawbacks of the PINN technique is its high computational cost during the training phase (Bandai and Ghezzehei, 2021), although predictions can be made in less than one second. If we modify the initial or boundary conditions, or any parameter defining the physical problem, it necessitates retraining the PINN solver. This drawback of PINN is also shared with other standard numerical methods, such as the finite element method. An alternative solution to this problem is the neural operator learning approach (Lu *et al.*, 2021a; Wang *et al.*, 2021b). In

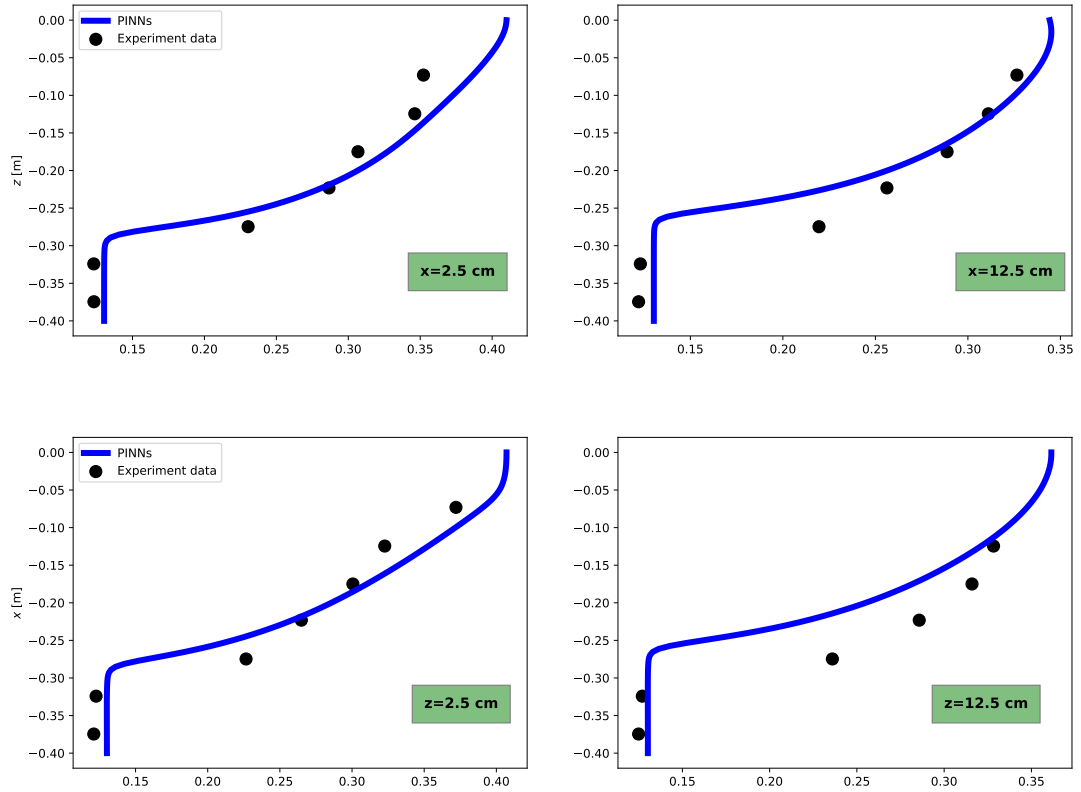


Figure 4.16 Cross-sections of the water content at coordinates $x = 2.5$ cm, $z = 2.5$ cm (left), and $x = 12.5$ cm, $z = 12.5$ cm (right), as predicted using the S-PINN solver with adaptive activation. The predicted results are compared with experimental data sourced from (Li *et al.*, 2005).

the neural operator technique, the capability of a neural network to approximate any nonlinear operator (Chen and Chen, 1993) is utilized to perform a regression between two infinite spaces. In the case of the coupled model of water flow and solute transport equations, one can learn the operator linking between initial/boundary conditions and the solution of the full system, while incorporating the physics encoded in the PDEs. Consequently, if we modify those conditions, the learned operator will promptly predict the desired solutions without requiring retraining. A comprehensive comparison between a deep neural operator and the S-PINN solver is provided in Appendix 6.6. The S-PINN solver has exhibited good accuracy in predicting the ammonium nitrification rate with sparse measurements. Enhancing the capability of the S-PINN approach, particularly when dealing with an inverse problem with unknown initial/boundary conditions, can be made more convenient by employing a more efficient method to handle the residual term in the loss function. Recent techniques proposed in (Anagnostopoulos *et al.*, 2024; McClenny and Braga-Neto, 2023) that locally adapt the weights associated

with each residual collocation point have shown promise. This technique is expected to be more effective for training the PINN solver in scenarios where there is no explicit description of initial or boundary conditions. In this study, we fine-tuned the weights for each loss term to balance accuracy and ensure stable convergence. For each test case, the weights for the initial and boundary condition loss terms were adjusted, while the residual loss weight was fixed at 1. Future work could automate this process by exploring adaptive weight schemes, such as the principled loss function (Chen *et al.*, 2023b), to enhance the S-PINN robustness for modeling water flow and solute transport in unsaturated soils. Moreover, enhancing the PINN framework using additional techniques, such as adaptive collocation point strategies (Wu *et al.*, 2023; Li *et al.*, 2023) and modified network architectures, could further improve the model’s accuracy and training efficiency.

In all numerical tests presented, the focus was on homogeneous soils. However, extending the S-PINN approach to heterogeneous soils presents additional challenges. While the XPINN framework (Jagtap and Karniadakis, 2020) offers a way to handle discontinuities by assigning separate neural networks to each sub-layer with continuity enforcement at interfaces, this approach has been applied mainly to 1D soils with a limited number of interfaces, as demonstrated in (Bandai and Ghezzehei, 2022). However, in highly heterogeneous environments, the number of required DNNs can increase significantly, making training more complex. In 2D and 3D domains, enforcing continuity across multiple interfaces further amplifies the computational demands. An alternative approach is the mixed-PINN formulation, which is based on the mixed form of governing equations. This method has been successfully applied to heterogeneous porous media with fewer DNNs, as shown in (Kamil *et al.*, 2024e; Lehmann *et al.*, 2023).

The S-PINN approach is inherently suited for one-way PDE systems, such as the water flow and nitrogen transport studied here, allowing separate training for each physical process. However, in strongly coupled systems, a modified algorithm is required, as the standard PINN framework struggles with multiple interacting processes. Furthermore, for inverse problems requiring simultaneous data from water content and multiple species concentrations, sequential training may fail to converge to realistic profiles.

Finally, we want to emphasize that the current work focuses on modeling water and solute dynamics in unsaturated conditions. A future extension of the PINN solvers would incorporate saturated conditions to better capture the transition between unsaturated and saturated zones.

4.6 Conclusion

This study investigates the efficacy of various architectures of physics-informed neural networks (PINNs) in modeling the coupled dynamics of water flow and solute transport in unsaturated soils. Three solvers, namely PINN, MPINN, and S-PINN, are proposed, each incorporating special enhancements such as learning rate scheduling, mini-batch

gradient descent, and an adaptive activation function. A series of 1D and 2D numerical experiments were conducted to assess the solvers in terms of training efficiency and accuracy. The introduction of the adaptive activation function notably improved the accuracy and training performance of all solvers. The primary advantage of the S-PINN solver over others is its sequential training approach. This method involves training a single neural network to predict the desired solution of each PDE, starting with the water flow equation and then moving on to other solute equations. This approach has demonstrated high accuracy in both forward and inverse problems. While prior studies have predominantly utilized the MPINN solver for coupled models (e.g., Haruzi and Moreno (2023a)), our main contribution is demonstrating the simplicity and efficiency of the S-PINN solver when combined with an adaptive activation function. Additionally, the transfer learning PINN approach proposed in (Kamil *et al.*, 2024c) can be adopted to further enhance the efficiency of the S-PINN solver.

The efficiency of the S-PINN solver was also evaluated for 2D problems, yielding satisfactory results when compared to a finite element reference solution or experimental data. It is worth noting that the methodology presented in this paper is adaptable for use with other physics-based machine learning algorithms.

In our future study, we aim to enhance the performance of S-PINN solver by focusing on realistic experiments. These experiments will encompass various scenarios, including atmospheric boundary conditions such as precipitation and evaporation, transpiration, plant absorption, different fertigation strategies and irrigation systems. The scientific community is encouraged by this initiative to incorporate the use of PINNs as an effective hybrid strategy that combines numerical and data-driven schemes. This method performs well for problems involving infiltration and transport phenomena in subsurface zones.

So far, different numerical models for water and solute transport in unsaturated soils have been developed. A standard FEM-based model was presented in Chapter 2, followed by the development of a PINN in Chapter 3, which was validated against the FEM solver. In this chapter, we demonstrated that the sequential training approach, referred to as S-PINN, is the most effective for both forward and inverse modeling. In Chapter 5, we further improve the performance of S-PINN in terms of accuracy, training stability, and inverse modeling by introducing a transfer learning strategy.

Code availability. The source code used in this study is publicly available from the GitHub repository <https://github.com/Hamza-Kamil/PINN-MPINN-Water-Solute>.

Appendix A. Pseudocode algorithms for the PINN solvers

In the following, we summarize the algorithms for the proposed PINN solvers to model water flow and single or multi-species solute transport in unsaturated soils.

Appendix B. Supplementary results

Hyperparameters vs. Accuracy. We investigated the impact of PINN solver settings on accuracy for modeling water infiltration in unsaturated soils. Using Test case A, we explored neural network architectures with widths of 10, 20, 30, and 50

Algorithm 1 PINN solver for coupled water flow and solute transport

Require: Training dataset (collocation points, measured data), Number of solutes n_s

Ensure: Predicted matric head, solute concentrations, water content, water flux, and mass changes

- 1: **Initialize network** $\mathcal{N}(\Theta)$:
 - 2: Set input dimension to time-space domain, output dimension to $1 + n_s$
 - 3: Define depth and width as per user specifications
 - 4: Initialize weights (Xavier scheme) and biases (zero)
 - 5: Set $\alpha = 0.05$, $s = 20$ for adaptive activation (if used)
 - 6: Apply tanh activation to hidden layers
 - 7: Set output activations: $-\exp(x)$ for matric head, identity for solutes
 - 8: **Train:**
 - 9: Minimize objective function J (Eq. (5.16)) using optimization algorithm from Section 8.3.1.5
 - 10: **Output:**
 - 11: Predict matric head and solute concentrations
 - 12: Compute water content and flux using Eqs. (5.1)-(8.4a)
 - 13: Calculate temporal changes in water and solute mass using Eq. (4.19)
-

neurons, and depths of 2, 4, and 5 layers. After identifying the optimal architecture, we fine-tuned the weights for initial and upper boundary conditions, while keeping residual and bottom boundary weights fixed at 1. We tested α_{ic} values of 1, 5, 10, and 50, and α_{up} values of 1, 10, and 100. Accuracy was measured using the relative L^2 -error of the matric head variable. Figure 4.17 shows results for the S-PINN solver with adaptive activation, reporting mean and standard deviation from 10 runs of 10,000 iterations each using the Adam optimizer.

Figure 4.17 demonstrates that the architecture and weights for the loss terms significantly affect the accuracy of the S-PINN when solving the Richards equation. Shallow networks with a depth of 2 do not perform well, as the relative L^2 -error remains above 5%, even with increased layer width. For a fixed width, deeper networks produce lower errors, with the 5-layer network yielding the best performance. Specifically, a network with 50 neurons per layer and a depth of 5 achieved the minimum error of 0.9%. We observed that using very deep architectures (more than 8 layers) does not improve the accuracy of S-PINN. This may be due to the increased difficulty of training deeper networks effectively, potentially leading to overfitting or gradient vanishing problems. The influence of the weights in the loss function is more pronounced in this test case. Small values of α_{up} result in an incorrect matric head profile regardless of the chosen value of α_{ic} , with the relative L^2 -error remaining above 60%. This occurs due to the gradient pathology, where the solver becomes biased toward satisfying the residual loss, leading to convergence to a poor local minimum. A large weight associated with the soil surface boundary (α_{up}) is necessary, with the optimal value being 100 in this test case. For $\alpha_{up} = 100$, increasing α_{ic} reduces accuracy, and when $\alpha_{ic} = 50$, the resulting

Algorithm 2 MPINN solver for coupled water flow and solute transport

Require: Training dataset (collocation points, measured data), Number of solutes n_s

Ensure: Predicted matrix head, solute concentrations, water content, water flux, and mass changes

- 1: **Initialize networks:**
 - 2: **for** $i = 0$ to n_s **do**
 - 3: Define $\mathcal{N}(\Theta_i)$: $\mathcal{N}(\Theta_0) = \mathcal{N}(\Theta_\Psi)$, $\mathcal{N}(\Theta_{i>0}) = \mathcal{N}(\Theta_{c_i})$
 - 4: Set input dimension to time-space domain, output dimension to 1
 - 5: Define depth and width as per user specifications
 - 6: Initialize weights (Xavier scheme) and biases (zero)
 - 7: Set $\alpha = 0.05$, $s = 20$ for adaptive activation (if used)
 - 8: Apply tanh activation to hidden layers
 - 9: Set output activation: $-\exp(x)$ for $i = 0$, identity for $i > 0$
 - 10: **end for**
 - 11: **Train:**
 - 12: **for** $i = 0$ to n_s **do**
 - 13: Compute individual loss J_i (residual, data, initial, boundary conditions)
 - 14: **end for**
 - 15: Compute total loss $J = \sum_{i=0}^{n_s} J_i$
 - 16: Minimize J using optimization algorithm from Section 8.3.1.5
 - 17: **Output:**
 - 18: Predict matrix head and solute concentrations
 - 19: Compute water content and flux using Eqs. (5.1)-(8.4a)
 - 20: Calculate temporal changes in water and solute mass using Eq. (4.19)
-

profile is incorrect, with a relative L^2 -error close to 70%. The optimal values for the loss weights were $\alpha_{ic} = 1$ for the initial condition and $\alpha_{up} = 100$ for the upper boundary condition, resulting in a relative L^2 -error of approximately 1%.

The number of collocation points within the time-space domain significantly affects S-PINN accuracy for solving the Richards equation (results omitted). Insufficient points ($\leq 5,000$) lead to inaccurate matrix head profiles, while excessive points ($> 50,000$) cause training instability. Approximately 10,000 points are required to achieve an L^2 -error of around 1%. A mini-batch approach is recommended for handling large collocation point sets.

Sequential optimization efficiency. In the following, we present additional results on training the S-PINN solver using Adam and L-BFGS optimizers. The aim is to assess if this sequential optimization improves the training efficiency and accuracy of the S-PINN solver for modeling the coupled flow-transport system in unsaturated soils. The other PINN solvers were excluded as they yielded similar conclusions.

Test case A is repeated using the S-PINN solver with the adaptive activation technique, maintaining the same hyper-parameters. During training, the total loss is monitored for each epoch using the Adam optimizer, followed by the L-BFGS optimizer. Additionally,

Algorithm 3 S-PINN solver for coupled water flow and solute transport

Require: Training dataset (collocation points, measured data), Number of solutes n_s

Ensure: Predicted matrix head (Ψ) and solute concentrations (c_i)

- 1: **Initialize:**
 - 2: Define neural networks $\mathcal{N}(\Theta_\Psi)$ for water flow and $\mathcal{N}(\Theta_{c_i})$ for each solute $i = 1, \dots, n_s$
 - 3: Set network parameters:
 - 4: Input: time-space domain dimension; Output: 1
 - 5: Hidden layers: tanh activation
 - 6: Output activation: $-\exp(x)$ for water flow; identity for solutes
 - 7: Xavier weight initialization; zero biases
 - 8: For adaptive activation: $\alpha = 0.05$, $s = 20$
 - 9: **Train:**
 - 10: Train $\mathcal{N}(\Theta_\Psi)$:
 - 11: Minimize loss J_Ψ (residual, data, initial, boundary conditions)
 - 12: **for** $i = 1$ to n_s **do**
 - 13: Train $\mathcal{N}(\Theta_{c_i})$:
 - 14: Minimize loss J_{c_i} (residual, data, initial, boundary conditions)
 - 15: Use $\mathcal{N}(\Theta_\Psi)$ for water content and flux in solute PDE residual
 - 16: If applicable, use $\mathcal{N}(\Theta_{c_i})$ for concentration in solute PDE residual
 - 17: **end for**
 - 18: **Output:**
 - 19: Predict matrix head and solute concentrations
 - 20: Compute water content and flux using Eqs. (5.1)-(8.4a)
 - 21: Calculate temporal changes in water and solute mass using Eq. (4.19)
-

the individual loss components, residual, initial condition, and boundary condition losses, are tracked. The S-PINN solver is trained for 2×10^4 epochs with Adam, followed by L-BFGS until convergence. The results are presented in Figure 4.18.

The S-PINN solver begins by addressing the highly nonlinear Richards equation. The Adam optimizer reduces the loss across all parts of the total loss function. The most difficult component to minimize is the residual loss (green curve), derived from the Richards equation operator. The oscillations observed in the loss curve in Figure 4.18 underscore the challenges of training a PINN loss function. This behavior reflects the complexity of the loss landscape in this problem, where significant non-convexity causes the Adam optimizer to face difficulties during training. A detailed theoretical analysis of this challenge is provided in (Rathore *et al.*, 2024), which demonstrates that the PINN loss function often becomes ill-conditioned for various PDE benchmarks due to the characteristics of the PDE operator. This issue is also evident in the case of the Richards equation.

The oscillations observed with the Adam optimizer are not present with the L-BFGS optimizer, which shows a smooth decrease in loss. The L-BFGS optimizer further

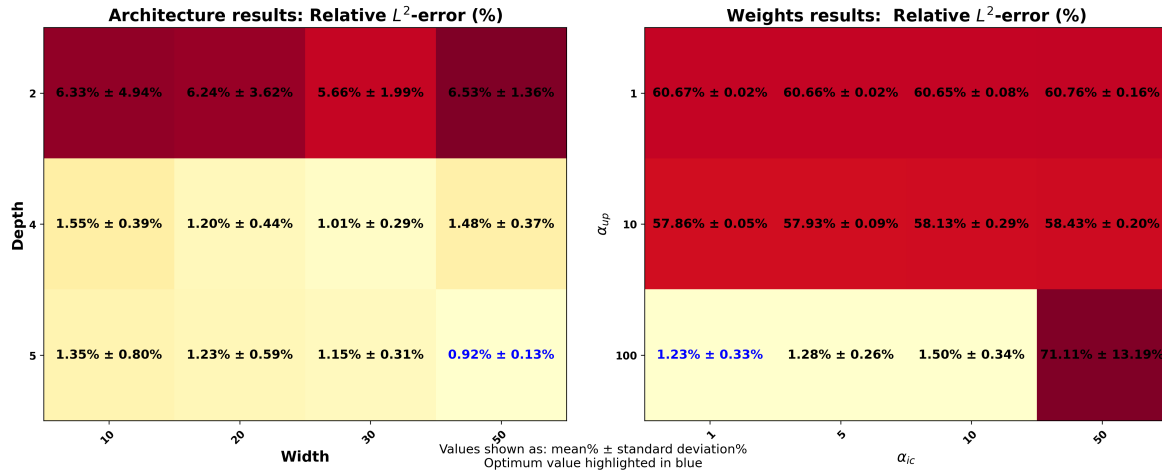


Figure 4.17 **Impact of hyperparameters on accuracy: Relative L^2 -error for different hyperparameter settings.** The left panel shows the analysis of neural network architecture, while the right panel illustrates the influence of loss term weights on accuracy. The parameter α_{up} represents the weight for the soil surface boundary condition, where water infiltrates the soil domain with an inflow q_0 (see Test Case A). Results are reported for the S-PINN solver with adaptive activation.

reduces the total loss achieved by the Adam optimizer in training the S-PINN solver. In terms of relative L^2 -error of the matric head, the L-BFGS optimizer improves the accuracy achieved by the Adam optimizer by 2.5%. Similar conclusions were reached when applying the S-PINN solver to the solute transport equation. These results suggest that the two-step optimization approach, utilizing the Adam and L-BFGS optimizers, yields favorable outcomes when applied to the S-PINN solver for addressing the coupled flow-transport system in unsaturated soils.

S-PINN vs Deep neural operators. In the following, we compare the performance of the S-PINN solver with deep neural operators. The aim is to demonstrate that the computational cost of the PINN approach can be reduced by leveraging deep operator learning-based methods, particularly for parameterized PDEs or multitask learning problems. To this end, we employ the DeepONet architecture proposed in (Lu *et al.*, 2021a) to solve various scenarios of the Richards equation by varying the initial conditions in terms of the matric head.

The core concept of DeepONets is to learn a mapping between two Banach spaces: the space of PDE parameters, $\mathcal{P}$, and the space of corresponding solutions, $\mathcal{S}$. In our study, $\mathcal{P}$ represents a set of initial conditions sampled uniformly over the spatial domain, and $\mathcal{S}$ represents the corresponding solutions (Kamil *et al.*, 2024b).

The DeepONet architecture consists of two parallel neural networks: the branch network and the trunk network. The branch network encodes inputs from the parameter space $\mathcal{P}$, represented as vectors of function values measured at fixed sensor locations.

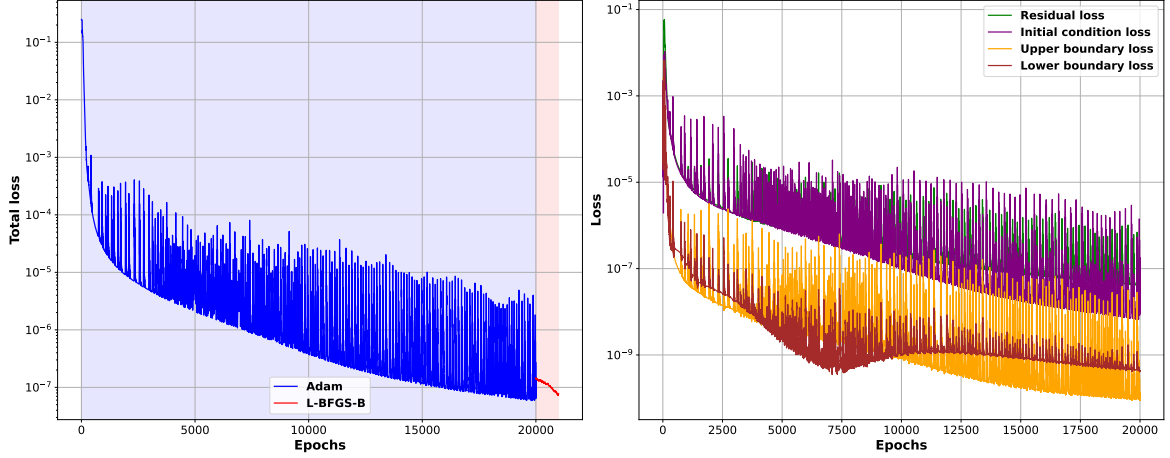


Figure 4.18 (Left) Comparison of total loss for S-PINN solver: Adam optimizer (blue) vs. L-BFGS (red). (Right) Breakdown of the residual, initial condition, and boundary losses using the Adam optimizer. The upper loss refers to the loss at the soil surface, while the lower loss corresponds to the loss at the soil bottom boundary.

The trunk network encodes inputs from the time-space domain $I \times \Omega$. The outputs of these two networks are then combined element-wise through a product operation to generate the final approximate solution to the PDE. For more details on the DeepONet architecture and training procedures, we refer to (Lu *et al.*, 2021a).

For the numerical experiment, we use the configuration from test case A. We generate 100 distinct initial conditions for the matric head, each defined at 100 spatial sensor points, resulting in 100 different versions of the Richards equation. These initial conditions are sampled uniformly within the spatial domain, corresponding to water content values within the interval (θ_r, θ_s) . For each initial condition, we use the FEM to solve the Richards equation and generate training data for the DeepONet model, consisting of matric head values at various time-space points. The DeepONet architecture includes branch and trunk networks, each with 4 layers and 100 neurons per layer, using tanh as the activation function in the hidden layers. The model is trained for 10^4 epochs using the Adam optimizer. In parallel, we solve the Richards equation using the S-PINN model for each initial condition as a baseline for comparison. Figure 4.19 presents the evolution of the total training loss for DeepONet and the computational costs of both DeepONet and the S-PINN solver.

Figure 4.19-(a) shows that the total training loss of DeepONet reaches an error on the order of 10^{-5} after 10^4 epochs of training with the Adam optimizer, indicating successful training convergence. Once trained, DeepONet can predict the solution for any new initial matric head condition, sampled uniformly in space, without any further training. In contrast, the S-PINN solver requires additional training for each new initial condition.

Figure 4.19-(b) shows the CPU time (in seconds) required for both DeepONet and the S-PINN solver to predict a new initial matrix head condition. For instance, with five different initial conditions, we measure the prediction time for DeepONet and the combined training and prediction time for S-PINN to solve the Richards equation for all five cases. The results in Figure 4.19-(b) show that DeepONet is significantly more computationally efficient, as it avoids the repeated training required by the S-PINN solver.

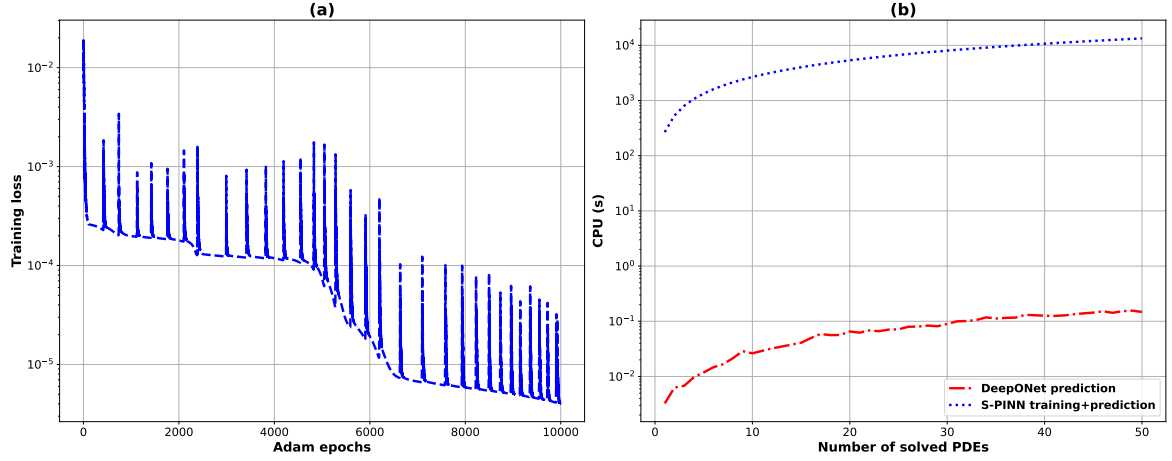


Figure 4.19 Comparison of S-PINN and DeepONet performance: (a) DeepONet loss function evolution over Adam epochs; (b) Computational cost comparison (in seconds) between S-PINN (including training and prediction) and DeepONet predictions, based on the number of PDEs to be solved (number of initial conditions).

Computational cost analysis. In the following, we present the CPU time for each PINN solver in the case of test A (see 4.4.1). Table 4.5 summarizes the training and prediction times for the different approaches.

Table 4.5 Comparison of training and prediction CPU time for different solvers

Solver	Training CPU time (s)	Prediction CPU time (s)
PINN	30.14	0.61
MPINN	48.95	0.56
S-PINN	55.02	0.81

The results show that the standard PINN approach requires the least training time, while the S-PINN approach is slightly more computationally intensive than PINN due to its sequential training process. However, this CPU time is acceptable given its superior accuracy. For prediction, all solvers exhibit similar performance, with negligible

differences in CPU time.

CHAPTER 5 A transfer learning physics-informed deep learning framework for modeling multiple solute dynamics in unsaturated soils

Abstract

Modeling subsurface flow and transport phenomena is essential for addressing a wide range of challenges in engineering, hydrology, and ecology. The Richards equation is a cornerstone for simulating infiltration, and when coupled with advection-dispersion equations, it provides insights into solute transport. However, the complexity of this coupled model increases significantly when dealing with multiple solute transports. Physics-informed neural networks (PINNs) offer a flexible technique that merges data-driven approaches with the underlying physics principles, enabling the direct incorporation of physical laws or constraints into the neural network training process. Nevertheless, employing PINNs for solving multi-physics problems can present challenges during training, particularly in achieving convergence to realistic concentration profiles. Our study introduces a transfer learning technique to tackle the challenge of modeling multiple species transport in unsaturated soils. This approach aims to improve the accuracy of the PINN framework by decoupling the training process and solving the governing partial differential equations (PDEs) sequentially. We incorporate various strategies to optimize and accelerate the training process. Specifically, we begin by solving the Richards equation and then transfer the acquired knowledge to subsequent solute PINN solvers. This strategy leverages the fact that these PDEs have some similarities in their structure as advection-diffusion equations. To rigorously validate our approach, we conduct 1D numerical experiments and extend our analysis to encompass 2D problems, and inverse problems for homogeneous soils, as well as numerical tests using layered soils. Our findings indicate that transferring learned features is more advantageous than utilizing random features, highlighting the effectiveness of the proposed strategy.

5.1 Introduction

Improving our understanding of the subsurface hydrological processes occurring in unsaturated zones is crucial for the development of efficient irrigation systems (Kandelous and Šimůnek, 2010; Kandelous *et al.*, 2011), controlling the fate of organic contaminants (Essaid *et al.*, 2015), and designing efficient drainage systems (Ritzema *et al.*, 1994; Stuyt and Dierickx, 2006). Soil moisture serves as a critical determinant, directly influencing plant growth, soil microbial activities, and nutrient cycling (Dougill *et al.*, 1998; Porporato *et al.*, 2003; Van Cleve and Yarie, 1986). In general, relying solely on soil moisture information is insufficient; therefore, it is crucial to model other pertinent physical processes. For example, the formulation of effective fertigation strategies relies on controlling the fate of applied water and fertilizers (Autovino *et al.*, 2018;

Provenzano, 2007).

Simulating physical, biological, or chemical phenomena often relies on established physical laws, which can be expressed mathematically as partial differential equations (PDEs). In the specific context of infiltration and transport in unsaturated soils, the Richards equation (Richards, 1931) is a fundamental governing equation for water flow, frequently coupled with advection-dispersion equations (Bear, 2013) to achieve a comprehensive modeling framework. The Richards equation presents a significant challenge due to its highly nonlinear nature. This complexity arises from the dependence of both the unsaturated hydraulic conductivity and the water content on the pressure head, often requiring the use of efficient numerical solvers to obtain accurate solutions. In addition, the advection-dispersion equations rely directly on the water content and water flux derived from the Richards equation, which are involved in driven forces. Consequently, inaccurate modeling of the water flow equation can significantly impact the numerical solution of the advection-dispersion equations.

Researchers have developed various numerical schemes for solving the Richards equation alone (Bergamaschi and Putti, 1999b; Boujoudar *et al.*, 2021a, 2023b; Keita *et al.*, 2021c), and the coupled model of water flow and solute transport (Chen *et al.*, 2019b; Illiano *et al.*, 2021; Russo, 2012a). Practitioners frequently use commercial software (Hanson *et al.*, 2006; Shafeeq *et al.*, 2020) developed based on these numerical schemes, due to its user-friendliness and specialized features. Numerical methods for solving PDEs typically involve mesh generation, dividing the computational domain into small elements, and approximating solutions at their nodes or centers. Numerical methods face challenges to deal with the issues of the instability associated with the sources of stiffness in the numerical solutions.

Data-driven approaches, empowered by deep learning solvers, present great potential as surrogate tools for solving diverse PDEs. These methods harness the capabilities of neural networks, whether employed individually or collaboratively, to approximate systems of PDEs, despite their linear or nonlinear characteristics (Hornik *et al.*, 1989). The primary limitation of utilizing a standard deep learning solver is its requirement of large datasets for training, which can be impractical in scenarios where field measurements are typically sparse and costly to obtain (Raissi *et al.*, 2019). Recently, a more powerful hybrid approach has emerged, transitioning from being purely data-driven to incorporating physics-informed principles into surrogate modeling (Raissi *et al.*, 2019). This hybrid approach, known as Physics-Informed Neural Networks (PINNs), has garnered widespread attention within the scientific community, leading to the successful solution of various PDEs (Eivazi *et al.*, 2022; Rasht-Behesht *et al.*, 2022; Wight and Zhao, 2020). For a deeper understanding of PINN applications in fluid mechanics, we recommend the literature review in (Cai *et al.*, 2021a). In a PINN solver, the loss function is redefined to encompass the physical constraints encoded in the associated PDEs. Automatic differentiation techniques play a crucial role, facilitating the precise computation of derivatives for the neural network with respect to its inputs (Baydin *et al.*,

2018). Various other variants of the PINN approach have been introduced, including variational formulations (Kharazmi *et al.*, 2019), *hp*-PINNs (Kharazmi *et al.*, 2021), and Bayesian-PINNs (Yang *et al.*, 2021). The main advantages of the PINN technique include: (a) its mesh-free nature, eliminating the need for intricate mesh generation in the computational domain. Instead, it uses random sampling points within the physical domain; (b) Unlike other purely data-driven approaches, sparse measurements are only required in cases where the studied problem is ill-posed, i.e., involving unknown parameters, initial, or boundary conditions (Bandai and Ghezzehei, 2022); and (c) PINNs offer greater efficiency in solving the inverse problem compared to calibrating standard numerical methods, resulting in lower computational costs (Bandai and Ghezzehei, 2021).

The application of PINNs in the realm of subsurface flows has proven successful. In (Bandai and Ghezzehei, 2021, 2022), the authors developed a framework utilizing the PINN approach to solve the Richards equation for both forward and inverse problems in one-dimensional soil columns, addressing homogeneous and layered soils. Additionally, in (Wang *et al.*, 2023b), the PINN approach was employed to solve the coupled flow of water and heat transfer in unsaturated soils. In (Haruzi and Moreno, 2023a), the authors utilized the PINN approach to address a 2D problem involving water infiltration and pore water electrical conductivity transport, with unknown initial conditions. Several other studies using this approach have also been conducted (Chen *et al.*, 2023b; Depina *et al.*, 2022; Elmorsy *et al.*, 2023; He *et al.*, 2020; Tartakovsky *et al.*, 2020). When tackling the coupled model of water flow and solute transport, existing PINN solvers are designed to simultaneously solve both equations. This poses a challenge for PINNs, as they endeavor to approximate two distinct PDEs without leveraging the interdependence inherent in the fact that the solute transport equation relies on the outputs of the Richards equation. This challenge becomes more pronounced when studying multiple solute transports concurrently. Directly implementing the PINN approach with multiple solutes in the coupled system tends to be unsuccessful (Laubscher, 2021).

To address the challenges associated with coupled modeling, we propose an efficient PINN-based framework for the dynamic of water flow and the transport of multiple solute species in unsaturated soils. This approach employs a decoupling strategy, initially addressing the Richards equation with a specialized PINN water solver. This sequential training approach aligns with the strategy adopted in other studies (e.g., Amini *et al.* (2022); Harandi *et al.* (2024); Wang *et al.* (2023b)). Additionally, we integrate enhancements such as the adaptive activation technique proposed in (Jagtap *et al.*, 2020). Subsequently, we apply transfer learning techniques (Chen *et al.*, 2015, 2021a; Tarbiyati and Nemati Saray, 2023) to leverage insights gained from the PINN water solver when addressing the first solute transport equation. Transfer learning strategies with PINNs have been explored in prior studies (Chen *et al.*, 2023a; Goswami *et al.*, 2020; Tang *et al.*, 2022; Xu *et al.*, 2023a). *This approach ensures the retention and*

reuse of knowledge obtained from solving the Richards equation. We therefore imply that the subsequent PINN solver is initiated using the parameters obtained in the PINN water solver, avoiding random initialization. In this setup, the characteristics learned by the PINN water solver are transferred to the other species' transport solvers. This transfer is justified by the commonality of all the PDEs in the class of advection-diffusion equations. For species involving chemical reactions, such as nitrogen species, after solving for the first species, the subsequent species utilize its features to initialize the corresponding PINN solver. This sequential process continues until the entire system of PDEs is addressed. In the proposed transfer learning approach, we adopt a freezing technique to alleviate the computational cost. It is worth noting that our methodology remains applicable even when substituting the PINN approach with any other physics-informed deep-learning algorithm. The main advantage of the proposed transfer learning PINN method is that it expends most of its optimization efforts on minimizing the total loss function of the PINN solver applied to the Richards equation. This, in turn, allows the PINN solute solvers that follow to achieve high accuracy with just a few iterations. We will show that, compared with a PINN with random initialization, a pre-trained PINN provides a better initialization, which makes the training much faster.

The standard PINNs have limitations in handling soil heterogeneity, and several techniques have been proposed to improve their efficiency (Alhubail *et al.*, 2022; Fraces and Tchelepi, 2021). To include some aspects of heterogeneity of the medium, we investigate how the proposed transfer learning PINN approach can be extended to take into account sharp gradients and discontinuities of the physical properties of soils. The proposed strategy is based on the mixed form of PDEs.

The rest of the paper is organized as follows: In Section 5.2, we outline the mathematical models governing water flow, solute transport, and nitrogen multiple species transport in unsaturated soils. Section 5.3 details the numerical experiment conducted in this study. The methodology for the proposed PINN solvers is explained in Section 5.4. In Section 5.5, we present and discuss the numerical results. Finally, Section 5.7 summarizes the study, presents key conclusions, and outlines future perspectives.

5.2 The mathematical models

5.2.1 Water flow model

In this study, the Richards equation is employed to simulate the infiltration of water into unsaturated soils. The Darcy-Buckingham law and the mass conservation principle can be combined to derive this equation (Buckingham, 1907). The equation is given in its mixed form as follows (Celia *et al.*, 1990b):

$$\begin{cases} \frac{\partial \theta}{\partial t} + \nabla \cdot \mathbf{q} = 0, \\ \mathbf{q} = -K \nabla (\Psi + z), \end{cases} \quad (5.1)$$

where θ is the volumetric water content, Ψ is the pressure head, $\mathbf{q}$ is the water flux, $K = K_s k_r$ is the unsaturated hydraulic conductivity, in which K_s is the saturated hydraulic conductivity and k_r is the water relative permeability. The time is denoted by t , and the spatial coordinates are given by $\mathbf{x} = (x, z)$, where x represents the horizontal dimension (in two-dimensional cases), and z represents the vertical dimension assumed positive upward.

Several empirical models have been developed to provide explicit functions for K and θ with respect to Ψ . Notably, models proposed by van Genuchten (Van Genuchten, 1980), Brooks and Corey (Brooks and Corey, 1966), Gardner (Gardner, 1958), and Haverkamp (Haverkamp *et al.*, 1977) are commonly utilized in the literature. In this study, the water content θ as a function of Ψ is described using the van Genuchten model, while the unsaturated hydraulic conductivity K follows Mualem's model (Mualem, 1976). The complete expressions for both are as follows:

$$\theta = \begin{cases} \theta_r + (\theta_s - \theta_r) \left(1 + (\alpha_v |\Psi|)^{n_v} \right)^{-m_v} & \text{if } \Psi \leq 0, \\ \theta_s & \text{if } \Psi > 0, \end{cases} \quad (5.2a)$$

$$K = \begin{cases} K_s \frac{\left[1 - (\alpha_v |\Psi|)^{n_v-1} \left(1 + (\alpha_v |\Psi|)^{n_v} \right)^{-m_v} \right]^2}{\left(1 + (\alpha_v |\Psi|)^{n_v} \right)^{m_v/2}} & \text{if } \Psi \leq 0, \\ K_s & \text{if } \Psi > 0, \end{cases} \quad (5.2b)$$

$$m_v = 1 - \frac{1}{n_v}, \quad n_v > 1, \quad (5.2c)$$

where θ_s and θ_r represent the saturated and residual volumetric water contents, respectively, and n_v , m_v and α_v are the van Genuchten empirical parameters.

5.2.2 Solute transport model

In unsaturated soils, the coupling of the Richards equation with the advection-dispersion equation is employed to model the interconnected subsurface infiltration-transport phenomena. The model takes the following form (Bear, 2013; Šimůnek *et al.*, 2005):

$$\frac{\partial \theta c}{\partial t} + \frac{\partial \rho s}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c - c \mathbf{q}), \quad (5.3)$$

where c is the concentration of the solute in the liquid phase, s is the concentration of the solute in the solid phase, ρ is the soil bulk density, $\mathbf{D}$ is the dispersion tensor, which incorporates effects of molecular diffusion and hydrodynamic dispersion, and as already mentioned, θ and $\mathbf{q}$ are the water content and the water flux, respectively, given by the Richards equation (5.1).

The one-way coupled model (5.1)-(5.3) in rigid medium, used in this study, is extensively utilized in various real-world applications ((Hanson *et al.*, 2006; Khashaei *et al.*, 2024; Ebrahimian *et al.*, 2013; Liu *et al.*, 2021; Chen *et al.*, 2014; Caiqiong and Jun, 2016; Nielsen *et al.*, 1986)). The Freundlich equation can be used to represent the interaction between the concentrations s and c (Šimůnek *et al.*, 2005):

$$s = K_d c^p, \quad (5.4)$$

where p is the Freundlich constant, which is equal to 1 when the model is assumed to be linear, and K_d is an empirical coefficient, which is equal to zero if no interaction between soil and solute is considered.

By substituting the Freundlich equation (5.4) into the solute transport equation (5.3), we can derive the following compact form:

$$\frac{\partial R \theta c}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c - c \mathbf{q}), \quad (5.5)$$

where the retardation factor R is determined by the following formula and accounts for the interaction between the solute and the soil solid phase:

$$R = 1 + \rho K_d \frac{c^{p-1}}{\theta}. \quad (5.6)$$

When the gaseous phase is neglected, the dispersion tensor's components are computed as follows (Bear, 2013):

$$\theta \mathbf{D}_{ij} = (D_T \|\mathbf{q}\| + \theta D_w \tau) \delta_{ij} + (D_L - D_T) \frac{\mathbf{q}_i \mathbf{q}_j}{\|\mathbf{q}\|}, \quad (5.7)$$

where D_w is the molecular diffusion constant, D_L and D_T are the longitudinal and transversal dispersivities, δ_{ij} is the Kronecker symbol, τ is the tortuosity factor, and $\|\mathbf{q}\|$ denotes the Euclidean norm of the water flux vector, having the components $\mathbf{q}_i$ and $\mathbf{q}_j$ in the respective spatial directions x and z . The tortuosity factor is calculated by using the following formula (Bear, 2013; Millington and Quirk, 1961):

$$\tau = \frac{\theta^{7/3}}{\theta_s^2}. \quad (5.8)$$

5.2.3 Nitrogen species transport model

For situations where we have multiple solute transport equations, which is the main focus of this study, zero and first-order decay coefficients can be used to account for the interactions between solutes (Šimůnek *et al.*, 2005). The dynamics of multiple solutes can be modeled using the following system of PDEs (Kaluarachchi and Parker, 1988;

Šimůnek *et al.*, 2005):

$$\begin{cases} \frac{\partial R\theta c_1}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c_1 - c_1 \mathbf{q}) - \mu_1 \theta c_1, \\ \frac{\partial \theta c_2}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c_2 - c_2 \mathbf{q}) + \mu_1 \theta c_1 - \mu_2 \theta c_2, \\ \frac{\partial \theta c_3}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c_3 - c_3 \mathbf{q}) + \mu_2 \theta c_2 - \mu_3 \theta c_3, \end{cases} \quad (5.9)$$

where the concentration and the first-order decay coefficient of solute k are indicated respectively by c_k and μ_k . The system of PDEs given by (6.27) is utilized for modeling nitrogen species transport. Specifically, ammonium NH_4^+ and nitrate NO_3^- are considered the initial nitrogen sources. Autotrophic bacteria catalyze the nitrification of ammonium, producing nitrite (NO_2^-) at a rate of μ_1 . Subsequently, further nitrification occurs, with nitrite transforming into nitrate (NO_3^-) at a rate of μ_2 . Finally, denitrification of nitrate takes place, leading to the production of dissolved nitrogen (N_2) and (N_2O) at a rate of μ_3 (refer to Figure 5.1).

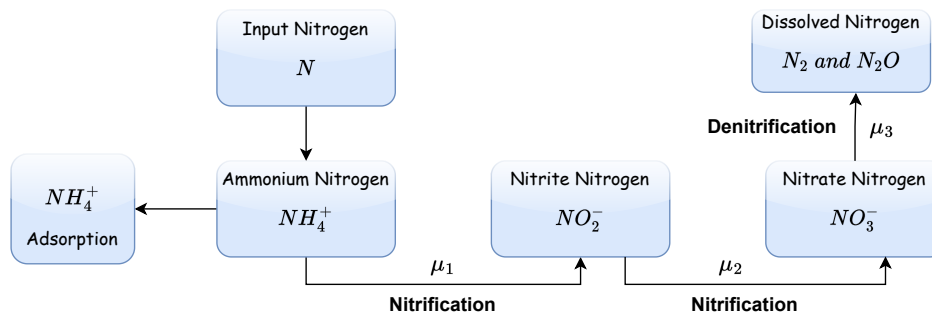


Figure 5.1 Representation of the first-order decay chain of nitrogen species.

For consistency with system (6.27), we use c_1 , c_2 , and c_3 to represent the concentrations of NH_4^+ , NO_2^- , and NO_3^- , respectively. Additionally, μ_1 , μ_2 , and μ_3 are employed to denote the first-order rate coefficients for the nitrification of NH_4^+ , the nitrification of NO_2^- , and the denitrification of NO_3^- , respectively.

The model (6.27) incorporates exclusively first-order degradation processes, such as nitrification, denitrification, and adsorption. Specifically, the assumption is that only ammonium undergoes linear adsorption onto the solid phase of the soil, characterized by the retardation factor R ($p = 1$). The nitrogen species transport model can consider additional processes; for instance, extra species such as potassium or phosphorus could be incorporated into the coupled system to represent their dynamics when using an NPK fertilizer (Hanson *et al.*, 2006). Importantly, it should be noted that the inclusion of these additional species does not impact the nitrogen decay chain model. For a comprehensive understanding of the nitrogen cycle, we recommend consulting the study by Kim *et al.* (Kim *et al.*, 2008).

5.2.4 Initial and boundary conditions

Here, we outline the initial and boundary conditions added to equations (5.1) and (6.27). The set of initial and boundary conditions employed for the Richards equation are expressed as follows:

$$\begin{cases} \Psi(0, x) = \Psi_0(x) & \text{in } \Omega, \\ \Psi = \Psi_D & \text{in } I \times \Gamma_D \subset I \times \partial\Omega, \\ \mathbf{q} \cdot \mathbf{n} = q_0, & \end{cases} \quad (5.10)$$

where $I \times \Omega$ is the time-space domain ($I = [0, T]$), Ψ_0 , Ψ_D , and q_0 represent the initial pressure head, Dirichlet and Neumann boundary conditions for the pressure head and the water flux respectively, $\partial\Omega$ denotes the boundary of Ω , and Γ_D and Γ_N refer to the Dirichlet and Neumann boundaries, respectively. The outward unit normal vector to the boundary Γ_N is represented by $\mathbf{n}$. We note that the initial condition can be specified either as a function of the water content θ , the effective saturation S , or the pressure head Ψ .

For the nitrogen transport equations (6.27), the following initial and boundary conditions can be used:

$$\begin{cases} c_i(0, x) = c_i^0(x) & \text{in } \Omega, \\ c_i = c_i^D & \text{in } I \times \Gamma_D \subset I \times \partial\Omega, \\ -(\theta \mathbf{D} \nabla c_i - \gamma c_i \mathbf{q}) \cdot \mathbf{n} = \bar{c}_i^0 q_0 & \text{in } I \times \Gamma_N \subset I \times \partial\Omega, \end{cases} \quad (5.11)$$

where c_i^0 denotes the initial concentration of solute i , c_i^D represents the specified value of solute i at the Dirichlet boundary, and $\bar{c}_i^0$ is the applied inlet concentration of solute i . The third equation in (5.11) corresponds to a second-type (Neumann) boundary condition when $\gamma = 0$, and a third-type (Cauchy) boundary condition when $\gamma = 1$.

5.3 Studied cases

To simulate the complete coupled dynamics of water flow and nitrogen species transport in unsaturated soils, we conducted the following 1D and 2D numerical experiments based on loamy soil with the following parameters:

$$K_s = 0.2496 \text{ m/day}, \alpha_v = 3.6 \text{ m}^{-1}, \theta_s = 0.43, \theta_r = 0.078, n_v = 1.56, L = 1 \text{ m}, \quad (5.12)$$

where L represents the depth of the soil.

For the water flow equation, the initial pressure head is set to -1.0 m, the soil surface boundary condition is considered as a flux boundary of specific discharge $q_0 = -0.05$ m/day, and free drainage is assumed at the bottom of the soil column, which can be

expressed as $\left. \frac{\partial \Psi}{\partial z} \right|_{z=L} = 0$.

For the nitrogen species model, the following parameters are used:

$$D_L = 0.04 \text{ m}, D_w = 0.288 \text{ cm}^2/\text{day}, \rho = 1.6 \times 10^{-6} \text{ mg/l},$$

$$K_d = 0.34 \times 10^{-6} \text{ l/mg}, \mu_1 = 0.12 \text{ day}^{-1}, \mu_2 = 0.048 \text{ day}^{-1}, \mu_3 = 0.012 \text{ day}^{-1}. \quad (5.13)$$

The nitrogen kinetics parameters μ_i are obtained from (Bailey *et al.*, 2013), while other parameters are sourced from (Berlin and Suresh Kumar, 2018).

The initial state of the soil is considered free from any nitrogen species ($c_i = 0$, $i = 1, 2, 3$). The soil surface is treated as a flux boundary condition, characterized by a third-type Cauchy boundary condition (5.11) with inlets of 50, 0, and 20 mg/l for ammonium, nitrite, and nitrate, respectively. This implies that only ammonium and nitrate are applied as sources, and no nitrite is introduced. Ammonium-nitrate fertilizer is suitable for this scenario. Zero-concentration gradients are imposed on the bottom boundary for all nitrogen species ($\left. \frac{\partial c_i}{\partial z} \right|_{z=L} = 0$, $i = 1, 2, 3$). In the 2D numerical test, we maintained the same initial and boundary conditions as in the 1D scenario, with no flux boundary condition at the lateral boundaries within the spatial domain $[0, L]^2$. For the transversal dispersivity, we set $D_T = D_L/100$. Figure 5.2 depicts the studied cases for the coupled water flow and nitrogen species transport model in homogeneous soil. The studied cases for the layered soil will be given in detail in Section 5.5.8.

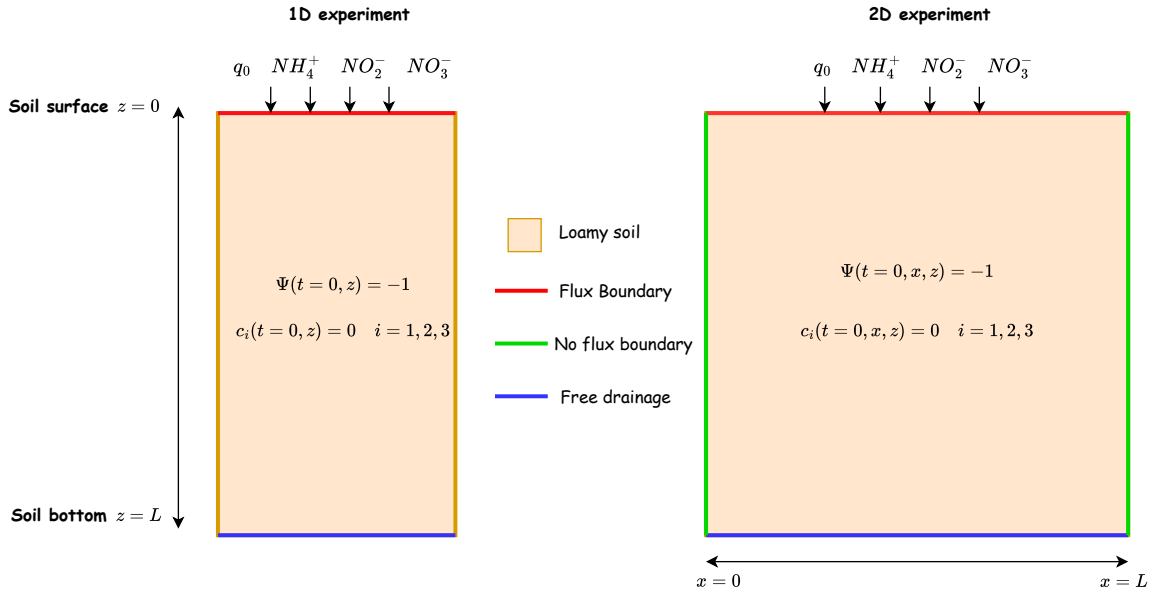


Figure 5.2 Configuration of 1D and 2D numerical experiments for water flow and nitrogen species transport.

5.4 Methods

5.4.1 Physics-Informed Neural Network: PINN

In this section, we provide a summary of the vanilla PINN method initially introduced in (Raissi *et al.*, 2019). For more details, readers are encouraged to explore the literature review presented in (Cai *et al.*, 2021a). PINNs represent a unique category of feedforward neural networks (FNNs). In these networks, the input consists of spatiotemporal data gathered from the domain $I \times \overline{\Omega}$, where $\overline{\Omega} = \Omega \cup \partial\Omega$. The output of these networks is the solution to the PDE that needs to be solved. These points can be sampled either uniformly or randomly, employing a specific probability distribution function (Zhang *et al.*, 2022). As an illustration, consider a random data point $(t, \mathbf{x})$ sampled from $I \times \Omega$. The FNN linearly transforms this data using weights W and a bias b . Subsequently, the FNN applies a nonlinear activation function and a specific output function. The final results for L layers are represented as follows:

$$\begin{cases} y_0 = (t, \mathbf{x})^T, \\ y_l = \sigma(W_l y_{l-1} + b_l), \quad \text{for } l = 1, \dots, L \\ \hat{y} = o(y_L). \end{cases} \quad (5.14)$$

Here, L denotes the number of layers in the FNN, and W_l and b_l represent the linear transformation parameters (weights and biases) of layer l . The function σ denotes the activation function applied at each hidden layer, and o is the output function of the FNN.

The primary distinction between FNNs and PINNs lies in their loss functions. In the case of PINNs, the loss function integrates the physical laws along with the associated initial and boundary conditions. This inclusion has demonstrated significant enhancement compared to employing a conventional FNN for solving a PDE. The primary advantage lies in the ability to train a PINN solver effectively with sparse data, eliminating the requirement for an extensive dataset, which can be challenging to obtain for real-world problems in engineering. For a clearer representation, the standard PINN framework can be illustrated as follows:

Once the dataset (time-space collocation points) for the PINN solver is collected, it is fed into the FNN to compute the corresponding solution. In this study, a separate network is employed for each PDE solution ($\Psi, c_k, i = 1, \dots, n_s$, where n_s represents the number of solutes under investigation). Therefore, we calculate the following residual functions:

$$\begin{cases} \hat{f}_R(t_i^r, \mathbf{x}_i^r) = \frac{\partial \hat{\theta}}{\partial t} - \nabla \cdot \left[\hat{K} \nabla (\hat{\Psi} + z) \right], \\ \hat{f}_{c_k}(t_i^r, \mathbf{x}_i^r) = \frac{\partial \hat{R} \hat{\theta} \hat{c}_k}{\partial t} - \nabla \cdot \left(\hat{\theta} \hat{\mathbf{D}} \nabla \hat{c}_k - \hat{c}_k \hat{\mathbf{q}} \right) + \hat{\Phi}_k; \quad k = 1, \dots, n, \end{cases} \quad (5.15)$$

where the hat notation is employed to represent the predictions made by the PINNs solvers, the first residual function corresponds to the residual loss in the Richards equation, while the subsequent ones pertain to each solute transport equation. The derivatives are computed using automatic differentiation. The points $(t_i^r, \mathbf{x}_i^r)$ represent residual data within the time-space domain $I \times \Omega$.

The total loss function of a PINN solver for multiple species transport, which enforces initial conditions, boundary conditions, and incorporate data measurements, can be defined as follows:

$$J(\Theta) = \alpha_d J_d(\Theta) + \alpha_r J_r(\Theta) + \alpha_b J_b(\Theta) + \alpha_{ic} J_{ic}(\Theta), \quad (5.16)$$

where

$$J_r(\Theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \left[\left(\hat{f}_R(t_i^r, \mathbf{x}_i^r) \right)^2 + \sum_{k=1}^{n_s} \left(\hat{f}_{c_k}(t_i^r, \mathbf{x}_i^r) \right)^2 \right], \quad (5.17a)$$

$$J_d(\Theta) = \frac{1}{N_d} \sum_{i=1}^{N_d} \left[\left(\Psi(t_i^d, \mathbf{x}_i^d) - \hat{\Psi}(t_i^d, \mathbf{x}_i^d) \right)^2 + \sum_{k=1}^{n_s} \left(c_k(t_i^d, \mathbf{x}_i^d) - \hat{c}_k(t_i^d, \mathbf{x}_i^d) \right)^2 \right], \quad (5.17b)$$

$$J_b^D(\Theta) = \frac{1}{N_b^D} \sum_{i=1}^{N_b^D} \left[\left(\Psi(t_i^D, \mathbf{x}_i^D) - \hat{\Psi}(t_i^D, \mathbf{x}_i^D) \right)^2 + \sum_{k=1}^{n_s} \left(c_k(t_i^D, \mathbf{x}_i^D) - \hat{c}_k(t_i^D, \mathbf{x}_i^D) \right)^2 \right], \quad (5.17c)$$

$$J_b^N(\Theta) = \frac{1}{N_b^N} \sum_{i=1}^{N_b^N} \left[\left(\mathcal{N}_\Psi(t_i^N, \mathbf{x}_i^N) - \mathcal{N}_{\hat{\Psi}}(t_i^N, \mathbf{x}_i^N) \right)^2 + \sum_{k=1}^{n_s} \left(\mathcal{N}_{c_k}(t_i^N, \mathbf{x}_i^N) - \mathcal{N}_{\hat{c}_k}(t_i^N, \mathbf{x}_i^N) \right)^2 \right], \quad (5.17d)$$

$$J_b(\Theta) = J_b^D(\Theta) + J_b^N(\Theta), \quad (5.17e)$$

$$J_{ic}(\Theta) = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} \left[\left(\Psi(0, \mathbf{x}_i^{ic}) - \hat{\Psi}(0, \mathbf{x}_i^{ic}) \right)^2 + \sum_{i=k}^{n_s} \left(c_k(0, \mathbf{x}_i^{ic}) - \hat{c}_k(0, \mathbf{x}_i^{ic}) \right)^2 \right]. \quad (5.17f)$$

In the expressions above, $\Theta = (W, b)$ represents the parameters of the FNN, and J_r , J_b^D , J_b^N , J_b , J_d , and J_{ic} correspond to the residual, Dirichlet boundary, Neumann boundary, measurement data, and initial condition losses, respectively. The terms N_r , N_b^D , N_b^N , N_d , and N_{ic} denote the total number of residual, Dirichlet boundary, Neumann-Cauchy boundary, data, and initial collocation points, respectively. The operator $\mathcal{N}$ signifies the boundary operator defined by the Neumann-Cauchy boundary conditions. The parameters α_d , α_r , α_b , and α_{ic} are weights assigned to the training measurement data, residual, boundary conditions, and initial conditions losses, respectively. We refer to Figure 5.3 for an illustration of the standard PINN solver for each PDE in the coupled system.

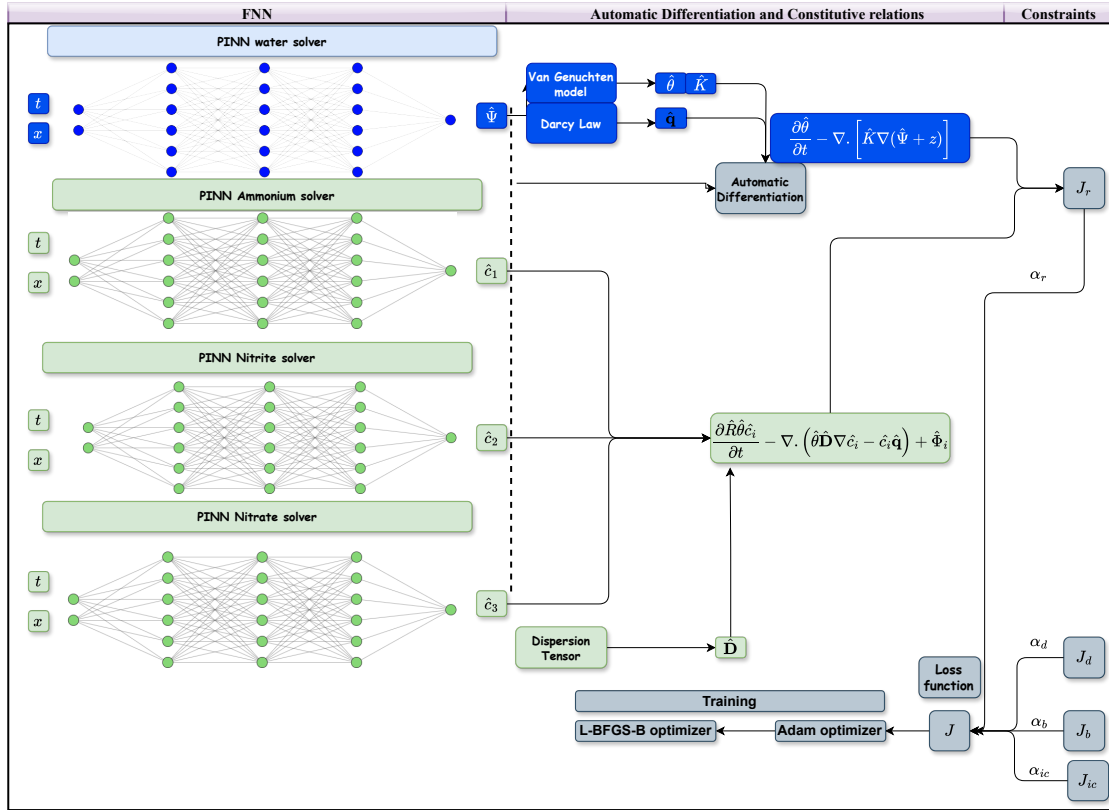


Figure 5.3 Diagram illustrating a standard PINN solver for the coupled water flow and nitrogen species transport model.

5.4.2 Improved Physics-Informed Neural Network

Applying the standard PINN method directly to solve highly nonlinear equations may lead to training failures or difficulty in converging to the accurate profile. Therefore, in this Section, we present several recent developments from the literature aimed at improving the training of the PINN solvers. Our primary contribution will be introduced in Section 5.4.3.

5.4.2.1 Adaptive activation function technique

Various studies have sought to speed up the training of PINN solvers. A recent technique, proposed in (Jagtap *et al.*, 2020), and explored in the context of the Richards equation in (Bandai and Ghezzehei, 2022), introduces an adaptive activation function. This method enhances the PINN solver's efficiency in capturing sharp fronts in the solution compared to the PINN using a standard activation function.

The key concept involves adding a trainable parameter, denoted as α , to the activation function of the FNN. For example, if the hyperbolic tangent function is selected, the

modified function becomes $\tanh(s\alpha x)$, with s as a scaling factor. The parameter α plays a crucial role in adjusting the slope of the hyperbolic function, enabling it to effectively capture sharp fronts and steep gradients of the solution. Figure 5.4 illustrates how modifying the parameter $s\alpha$ influences the slope of hyperbolic tangent and sigmoid functions.

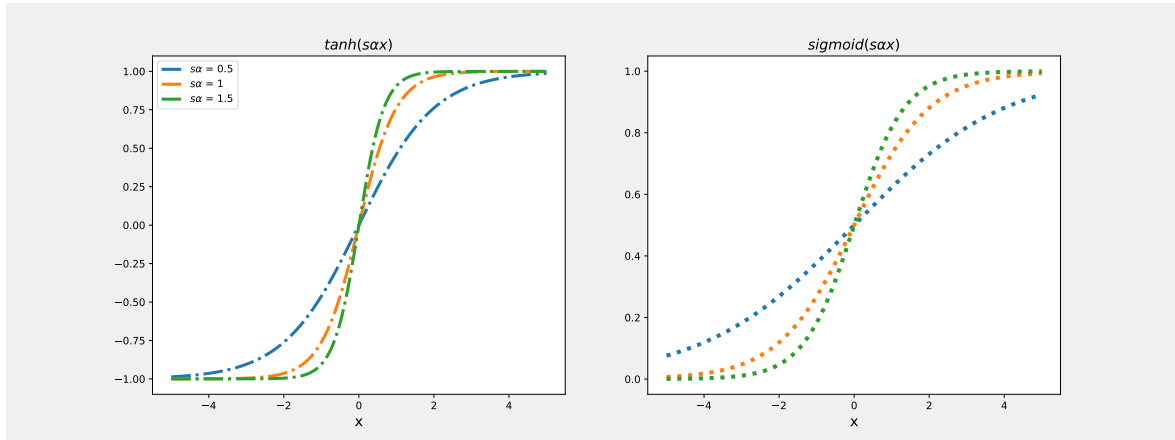


Figure 5.4 Activation functions: Hyperbolic tangent (left) and sigmoid (right) with varied $s\alpha$ values.

5.4.2.2 Mini-batch sizing technique

The most commonly used FNNs generally employ the entire dataset for calculating the loss value at each iteration or epoch, introducing various drawbacks from an optimization standpoint. This method may encounter convergence challenges, such as getting stuck in local minima. To address these issues, the deep learning community has introduced the mini-batch sizing technique to improve the training of deep learning solvers.

Mini-batch sizing involves dividing the entire dataset into smaller batches, and the loss is computed based on each mini-batch during training. This approach mitigates the risk of convergence issues and improves the optimization, especially in the PINN approach when solving stiff PDEs (McClenny and Braga-Neto, 2023). One explanation for the observed superiority of stochastic gradient descent (SGD) over large-batch training lies in the loss surface landscape (McClenny and Braga-Neto, 2023). Large-batch training can converge to sharp minima, which often leads to poorer network performance. Conversely, SGD calculations with small batches are more likely to find flat minima, which tend to exhibit better generalization capabilities (McClenny and Braga-Neto, 2023).

5.4.2.3 Learning rate schedule

A robust optimization algorithm is crucial to optimize the highly nonlinear and non-convex function J . Many PINN studies have employed the Adam optimizer (Kingma

and Ba, 2014), a widely-used stochastic gradient descent variant in deep learning algorithms, and L-BFGS-B (Byrd *et al.*, 1995), which belongs to the quasi-Newton methods family.

Employing a fixed learning rate in an optimization algorithm can lead to delayed convergence, oscillations, and the risk of becoming trapped in undesirable local minima. Therefore, an adaptive learning rate is recommended. Several adaptive learning rates have been proposed in the literature, including step decay, exponential decay, inverse decay, polynomial decay, and piecewise constant. In this study, we choose exponential decay, defined as follows:

$$\eta_t = \eta_0 \exp(-dt), \quad (5.18)$$

where η_t denotes the learning rate at iteration t , and d is the decay factor.

5.4.3 Transfer learning Physics-Informed Neural Network: TL-PINN

The main contribution of this study, as outlined in this section, is to introduce an initialization transfer learning technique to enhance the efficiency of the PINN method's training for solving multiple solute transport species in unsaturated soils. For a comprehensive understanding of the transfer learning technique, we recommend referring to the recent survey by Zhuang *et al.* (Zhuang *et al.*, 2020).

The transfer learning approach involves using knowledge from other deep learning solvers and then applying it to address a new task. While a few studies have explored this technique in the context of PINNs (e.g., Tarbiyati and Nemati Saray (2023)), our approach takes advantage of the one-way coupling within the system. We propose to decouple the studied PDEs in the following manner:

1. Initially, our attention is directed towards solving the first equation in the system of PDEs, the Richards equation. This computation is crucial as it provides essential outputs for the other equations, encompassing those for water content, pressure head, and the water flux vector. We refer to the corresponding model as the PINN water solver.
2. Next, we solve the PDE governing ammonium using the corresponding model, referred to as the PINN ammonium solver. More broadly, if there are other solutes independent of the nitrogen species, we address their respective PDEs as well. In this step, we proceed to initialize the PINN ammonium solver by the optimized parameters obtained from the PINN water solver. This procedure is referred to as transfer learning, transitioning from the Richards equation to the solute transport equations.
3. After completing the training of the PINN ammonium solver in step 2, we proceed to initialize the parameters for the next solute concentration. This initialization is performed using the parameters obtained from the previous PINN solver, specifi-

cally corresponding to the decay chain. This iterative process continues until the last solute in the decay chain is addressed.

The proposed methodology offers several key advantages:

1. It utilizes realistic water content, pressure head, and water flux profiles for training other solute concentrations.
2. This methodology transfers information from previous profiles to expedite the training of subsequent desired PDEs. This is justified by the fact that all PDEs in the studied system belong to the same class as advection-diffusion equations.
3. As demonstrated in the subsequent numerical tests, this approach dedicates the majority of iterations to training the Richards equation and only a few iterations for solving other solute concentrations. Additionally, while specifically developed for this coupled system, this approach can be adapted to any directed decay chain system of PDEs.

In cases where field or laboratory data are available, we suggest training an ordinary FNN with a few iterations. The final weights of this FNN can then be used to initialize the PINN water solver. Alternatively, in the absence of available data, the Xavier initialization stochastic algorithm (Glorot and Bengio, 2010a) can be employed. It is noteworthy that the initialization of a PINN solver with measured data or data from a high-resolution solver (e.g., the finite element method) has been proposed in other studies (Tarbiyati and Nemati Saray, 2023). Using this approach, one can gather data from simulations based on numerical techniques or available software such as HYDRUS (Šimůnek *et al.*, 2005), and employ a portion of this data to train a standard FNN. We refer to Algorithm (4) for a more comprehensive understanding of the transfer learning PINN approach.

To optimize the computational efficiency of the transfer learning algorithm, a parameter freezing technique can be applied. Specifically, after resolving the Richards equation, the resulting parameter Θ_w can be utilized to initialize the parameters of the PINN solver for each initial solute species in a decay process, as outlined in Algorithm (4). The parameter freezing technique involves selectively freezing certain parameters within Θ_{c_i} while training others. Specifically, for a PINN solver for solute i , where the parameters of layer l are denoted by Θ_l , freezing k_{c_i} layers implies that the only trainable parameters are Θ_l for $l = k_{c_i}, \dots, L$, while Θ_l for $l = 0, \dots, k_{c_i} - 1$ remains non-trainable. A comprehensive description of the freezing technique is provided in Algorithm (5).

A key insight lies in the extensibility of the TL-PINN technique to diverse neural network architectures. In particular, distinct PINN architectures may be applied to address each PDE in the studied system, with transfer learning entailing a transition from one neural network to a wider or deeper one. To facilitate this process, the Net2Net technique, as proposed in (Chen *et al.*, 2015), can be employed.

Algorithm 4 Transfer Learning PINN for solving multiple solute species transport in unsaturated soils

Procedure 1: Available water content data $\mathcal{D}$

1. Train a standard FNN using data $\mathcal{D}$ to obtain its final parameters Θ_1 .
2. Initialize the parameters of the PINN water solver with Θ_1 .
3. Train the PINN water solver to obtain the final parameters Θ_w , along with corresponding solutions $(\theta, \Psi, \mathbf{q})$.
4. Initialize the parameters of each PINN solute solver, either corresponding to the initial solute species in a decay chain or an independent solute, with the parameters Θ_w .
5. Train the PINNs' solute solvers to obtain the final parameters Θ_c for each solute.
6. Train the next solute in a decay chain, initializing its parameters with the last PINN's final parameters.

Procedure 2: No available water content data $\mathcal{D}$

Replace step 1 in Procedure 1 by initializing the PINN water solver with a suitable random initialization algorithm (e.g., the Xavier initialization).

Algorithm 5 Transfer Learning PINN for solving multiple solute species transport in unsaturated soils with the parameter freezing technique

Require: 1. List containing the parameters in each layer of the PINN solver.
 2. **freeze- c_i** : Boolean variable indicating whether or not to freeze the parameters of the PINN solver of solute c_i (True for freeze, False for trainable).

1: **Initialization Steps:**

2: **if** **freeze- c_i** is True **then**

3: **for** the first k_{c_i} layers **do**

4: Create non-trainable variables for parameters.

5: **end for**

6: **for** the remaining layers (starting from k_{c_i} or 0 if not frozen) **do**

7: Create trainable variables for parameters.

8: **end for**

9: **end if**

10: Perform the Transfer Learning PINN Algorithm (4).

Ensure: A list of variables representing the parameters for each layer.

The overall flowchart of the transfer learning PINN solver for modeling water flow and multiple species transport in unsaturated soils is illustrated in Figure 5.5.

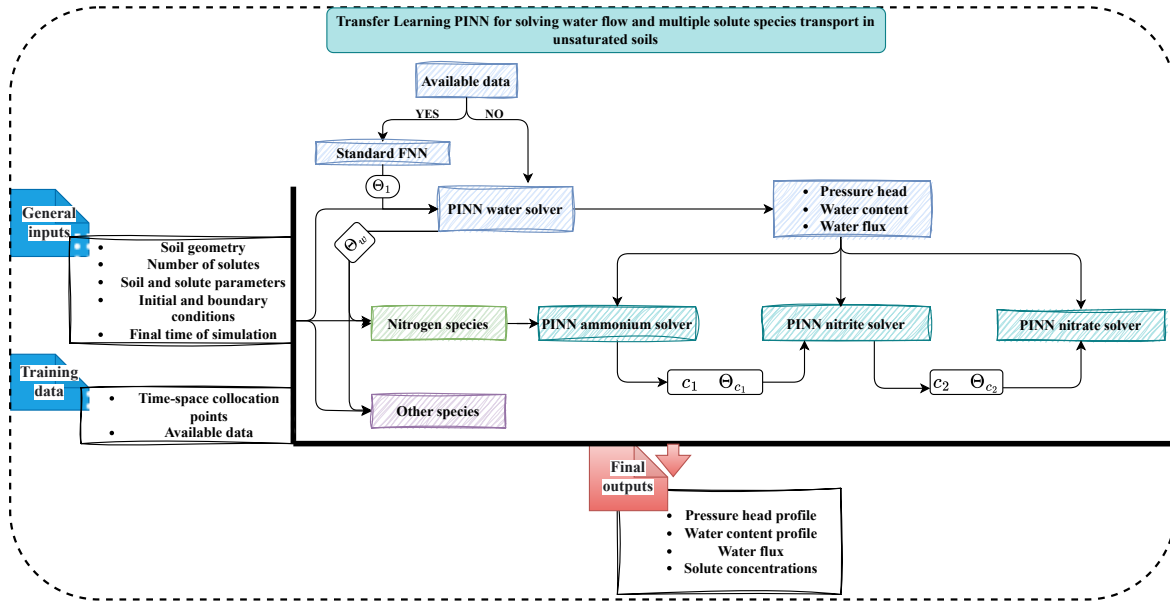


Figure 5.5 The overall flowchart of the transfer learning PINN solver for modeling water flow and multiple species transport in unsaturated soils.

5.4.4 Transfer learning with Mixed-PINN

Here, we investigate how the proposed transfer learning PINN approach can be extended to take into account sharp gradients and discontinuities in the physical properties of soils. Developing efficient techniques for solving problems where the solutions of PDEs exhibit shocks or discontinuities remains a significant challenge. The standard PINNs have limitations in effectively handling these cases, and several techniques have been proposed to improve their efficiency (Alhubail *et al.*, 2022; Fraces and Tchelepi, 2021).

We note that we can use the XPINNs proposed in (Jagtap and Karniadakis, 2020) for layered soils, where the time-space domain is decomposed into non-overlapping sub-domains, each associated with a PINN solver. This approach was adopted in (Bandai and Ghezzehei, 2022) to solve the Richards equation in a soil with two layers. In our case, the PINN solvers for the Richards equation will interact with each other by enforcing the continuity of the pressure head, water flux, and the residual of the governing PDE. For the transfer learning PINN, after solving the Richards equation, each PINN solute solver for each sub-layer will transfer its parameters to the next PINN solver in the same layer, and the process continues until all PDEs for solute transport are solved. In this study, we propose another method, as described below.

To develop a transfer learning PINN approach for the coupled model of water flow and multiple solute transport in the presence of discontinuities in the physical properties of soils, we extend our techniques by using the mixed form of the PDEs. Similarly to the homogeneous case, the governing PDEs for water flow and solute transport species are solved sequentially. However, for each PDE, we employ two neural networks: $\mathcal{N}_s$, used to predict the desired solutions such as pressure head for the Richards equation and concentration for each nitrogen species; and, $\mathcal{N}_q$ serve as the surrogate for the associated flux. The networks communicate with each other to ensure the compatibility conditions in terms of water and solutes fluxes. As an illustration, the compatibility conditions for water flow are specified as follows:

$$\begin{cases} \mathcal{N}_{q_x}(t, \mathbf{x}) = -K(\mathcal{N}_\Psi(t, \mathbf{x})) \frac{\partial \mathcal{N}_\Psi(t, \mathbf{x})}{\partial x}, \\ \mathcal{N}_{q_z}(t, \mathbf{x}) = -K(\mathcal{N}_\Psi(t, \mathbf{x})) \left(\frac{\partial \mathcal{N}_\Psi(t, \mathbf{x})}{\partial z} + 1 \right), \end{cases} \quad (5.19)$$

where $\mathcal{N}_\Psi$ is the network associated to the pressure head, $\mathcal{N}_{q_x}$ and $\mathcal{N}_{q_z}$ denote the fluxes in the x and z directions, respectively.

The training process for the mixed-PINN for each PDE follows the standard PINN methodology, involving the enforcement of the PDE residual along with initial and boundary conditions, as well as the compatibility conditions (5.19) for the fluxes. Among the advantages of the proposed mixed-PINN formulation is that it reduces the derivative computation of PDEs from second-order to first-order in space for each network (Harandi *et al.*, 2024; Lehmann *et al.*, 2023; Rezaei *et al.*, 2022).

In the TL-PINN method within the proposed mixed-PINN approach, the workflow is as follows. After solving the Richards equation using the mixed-PINN, the parameters of each neural network are transferred to the subsequent PINN solute solver. The transfer includes the network solution for the pressure head and the network fluxes. This iterative process continues until the last PDE for solute transport is solved. The transfer learning approach within the mixed-PINN method is detailed in Algorithm (6). We refer to Appendix 6.6 for more details about the training procedure of the mixed-PINN.

5.4.5 Algorithm implementation

The implementation of both the standard and transfer learning PINN solvers utilized the open-source software library TensorFlow (Abadi *et al.*, 2016). The activation function tanh is chosen because the governing PDEs of the coupled model of water and solute transport are second-order in space, and tanh is infinitely differentiable. The negative exponential function ($-\exp$) is used as the output activation for the PINN water solver.

In procedure 2 of Algorithm (4), Xavier random initialization is applied, and the initial value of the bias parameters b is set to zero. In the adaptive activation function

Algorithm 6 Transfer Learning Mixed-PINN for solving multiple solute species transport

1. Initialize the parameters (Θ_Ψ, Θ_q) of the PINN water solver. This includes two networks, $\mathcal{N}_\Psi(\Theta_s)$ for pressure head and $\mathcal{N}_q(\Theta_q)$ for water flux. Initialization can follow procedure 1 or 2 as shown in Algorithms (4) and (5).
 2. Train the PINN water solver based on the PDE residual, initial and boundary conditions, as well as the compatibility conditions (5.19) for the water flux. Obtain the final parameters (Θ_Ψ, Θ_q) , along with the corresponding solutions $(\theta, \Psi, \mathbf{q})$.
 3. Initialize the parameters (Θ_c, Θ_{qc}) with (Θ_Ψ, Θ_q) of the PINN ammonium solver. Two networks are employed: $\mathcal{N}_c(\Theta_c)$ for solute concentration and $\mathcal{N}_{qc}(\Theta_{qc})$ for solute flux.
 4. Train the PINN ammonium solver based on the PDE residual, initial, and boundary conditions, as well as the compatibility conditions (5.19) for the solute flux. Obtain the final parameters (Θ_c, Θ_{qc}) .
 5. Train the next solute species in the decay chain, initializing its parameters with the final parameters from the previous solute solver.
-

technique, the hyper-parameter α is initially set to 0.05, and the scale factor s is set to 20.

To solve the optimization problem presented by the objective function J , a two-step training technique is implemented, sequentially employing both the Adam optimizer and the L-BFGS-B optimizer. The Adam optimizer utilized an exponential decay of the learning rate, with the initial learning rate set to $\eta_0 = 10^{-3}$, a decay rate of $d = 0.9$, a decay step of 1000 (adjusting the learning rate after every 1000 epochs), and other parameters set to their default values. The parameters for the L-BFGS-B optimizer are set as follows: $\text{ftol} = 10^{-10}$, $\text{gtol} = 10^{-8}$, $\text{maxcor} = 50$, $\text{maxls} = 50$, $\text{maxiter} = 50,000$, $\text{maxfun} = 50,000$, with default values for the remaining parameters. A mini-batch of size 500 is used for the Adam optimizer, while a full batch is employed for the L-BFGS-B optimizer.

A total number of 10,000 residual collocation points are randomly sampled from the interior of the time-space domain, with an additional 3,000 collocation points dedicated to initial and boundary conditions. The effectiveness of vanilla PINNs is known to be intricately linked to the choice of sampling method for generating training points. A robust sampling strategy is critical for PINN convergence (Chen *et al.*, 2024). Figure 5.6 depicts four distinct sampling methods commonly employed in previous studies using the PINN approach for the generation of training points. In this study, we employed a uniform random sampling approach, where training points are selected randomly once at the beginning, and remained fixed throughout the training process. In Section 5.5.3,

we investigated the influence of different numbers of residual collocation points on the accuracy and training behavior of the TL-PINN solver. For the weights in the loss function (5.16), we used $\alpha_r = \alpha_{ic} = 1$ and $\alpha_b = 100$. Additionally, we assume no available data is provided for training the standard PINN and TL-PINN solvers (except for the inverse problem), and thus we set $\alpha_d = 0$.

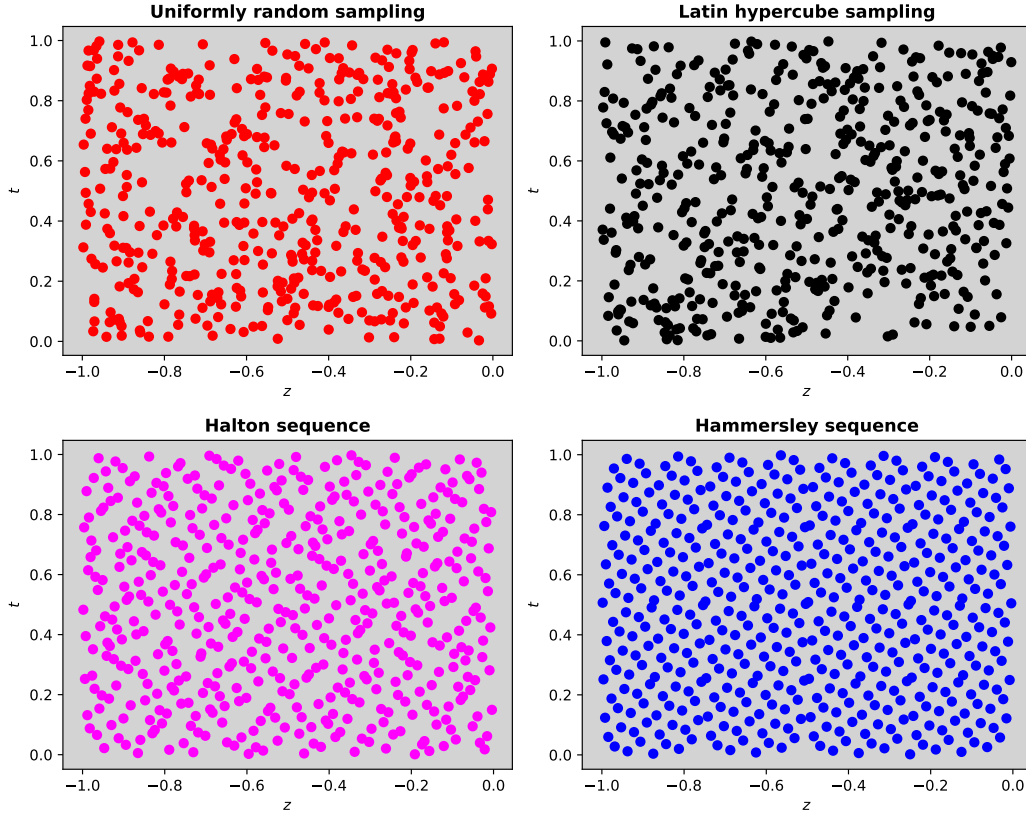


Figure 5.6 Various sampling methods for residual collocation points within the time-space domain $[0, 1] \times [-1, 0]$.

For the standard PINN solvers, we employed one network for each desired solution. Therefore, we trained four FNNs: one for the pressure head and one for the concentration of each nitrogen species. The architecture for the FNNs used for both standard PINNs and TL-PINNs is set as 5 hidden layers with 50 neurons in each layer. The impact of the FNN architecture on the TL-PINN solver is examined in Section 5.5.3. To minimize the objective function J in the standard PINN solver, we utilized 35,000 iterations with the Adam optimizer, followed by an additional 50,000 iterations using the L-BFGS-B optimizer. In contrast, for the TL-PINN solver, we initially trained the FNNs to solve the water flow equation with 20,000 Adam iterations. Subsequently, TL-PINNs are employed to address nitrogen transport PDEs, with only 5,000 Adam iterations for each solute transport equation.

The numerical experiments in this study were conducted using an NVIDIA V100 GPU. The source code is available on the GitHub repository:

<https://github.com/Hamza-Kamil/Transfer-Learning-PINN-Nitrogen>.

5.5 Results and discussions

5.5.1 A comparative analysis of transfer learning PINNs and standard PINNs

Here, we compare the accuracy of the standard PINN and the TL-PINN approach discussed in Section 5.4.3. We assess their performance in solving the numerical experiment outlined in Section 5.3. All the techniques detailed in Section 5.4.2 are employed for both standard PINNs and TL-PINNs to enhance the efficiency of their training. We conducted the same numerical experiment, using HYDRUS-1D software (Šimůnek *et al.*, 2005) to compare the predictions made by the two solvers. In this simulation, we did not employ the freezing technique outlined in Algorithm (5) for the TL-PINN solver. We refer to Section 5.4.5 for the configuration details of both solvers.

Figure 5.7 presents the results obtained using the TL-PINN solver in the time-space domain $[0, 1] \times [0, 1]$, where we show both the predicted profiles and the absolute error associated with pressure head and nitrogen species concentrations. To compare the TL-PINN with the standard PINN, we present cross-sections at the final simulation time $T = 1$ day from both solvers, and the results are depicted in Figure 5.8. Our predictions are in good agreement with the reference solution which confirm that the TL-PINN solver accurately captures the profiles of pressure head, water flux, and nitrogen species concentrations. In contrast, the standard PINN solver gets stuck at the first PDE in the system, the Richards equation, and fails to approximate the other profiles. Increasing the iterations of the Adam optimizer did not yield any improvement in the results for the PINN solver. Figure 5.9 depicts the evolution of the total loss function for the standard and TL-PINN solvers. The training process of the TL-PINN solver appears to be significantly simpler than the standard PINN approach. This simplicity arises from the fact that while the standard PINN solver needs to address the loss associated with solving the entire system of PDEs, the TL-PINN solver focuses solely on tackling the Richards equation. This supports our initial assertion that the loss function of standard PINNs is highly non-convex, making the solution of a large coupled system challenging. Handling such complexity with a standard PINN is not straightforward.

Another noteworthy observation regarding TL-PINNs is their ability to rapidly capture nitrogen species concentration in just a few iterations following the solution of the Richards equation. This demonstrates that the transfer learning approach, where knowledge is transferred from a highly nonlinear advection-diffusion equation to a linear one, significantly enhances the performance of the PINN solver compared to random initialization. The decoupling strategy, where each equation is solved independently,

serves as a cornerstone of the TL-PINN solver. In this strategy, solute transport equations utilize the final predicted results of water content and water flux to determine concentrations. Subsequent solute concentrations are then trained based on the preceding solute concentration profile, predicted earlier with the water content and flux. A central question arises: is this decoupling advantage unique to the TL-PINN solver, rendering knowledge transfer from the Richards equation unnecessary? The following section is dedicated to addressing this question.

Profiles of the obtained solutions

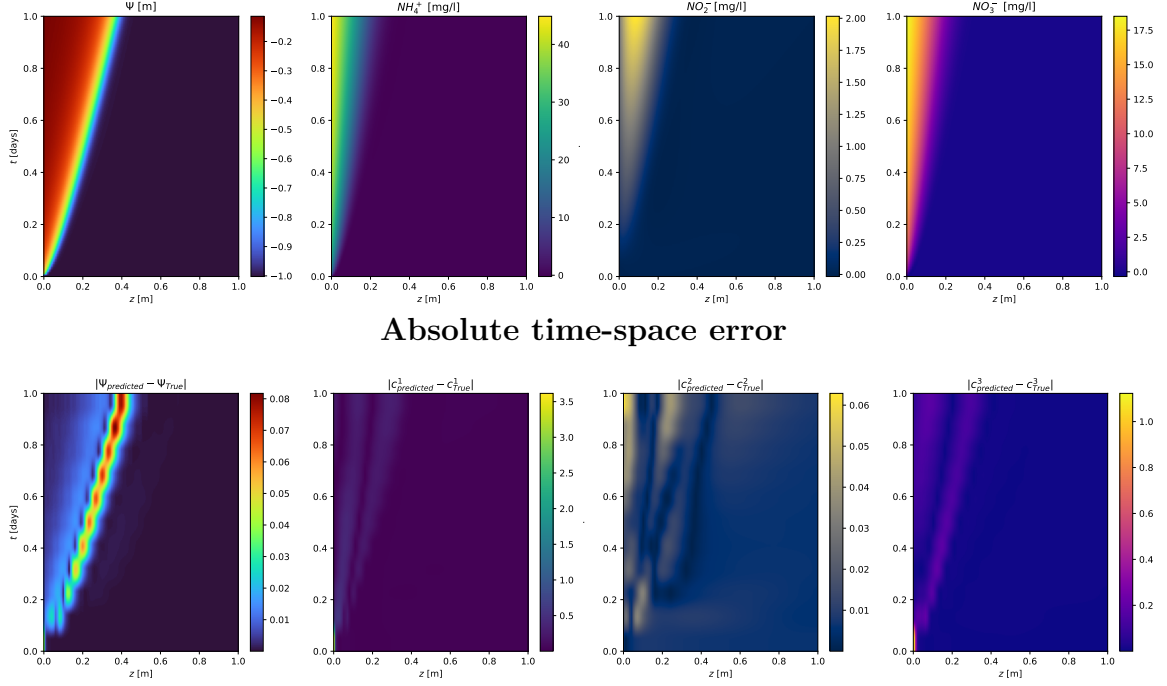


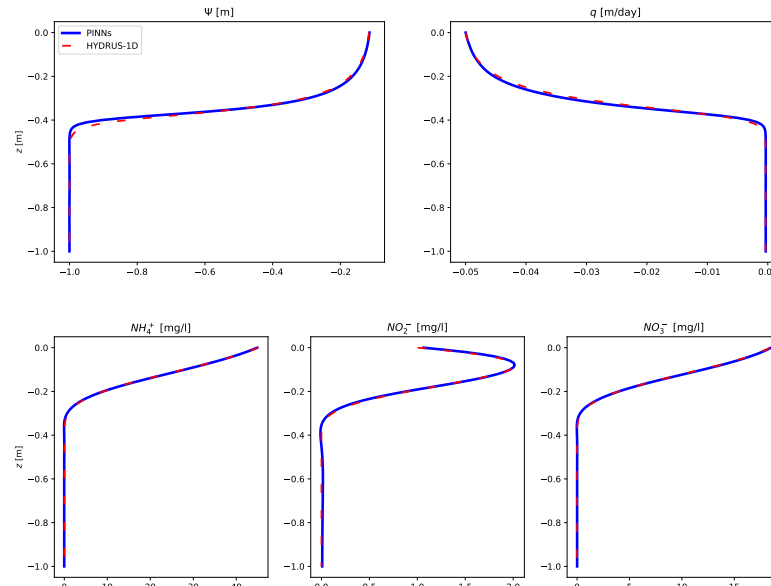
Figure 5.7 Profiles (top) and errors (down) of pressure head and nitrogen species concentrations obtained using TL-PINNs in the time-space domain $[0, 1 \text{ day}] \times [0, 1 \text{ m}]$.

5.5.2 Random initializer PINN vs transfer learning PINN

After establishing the superiority of the TL-PINN solver over the standard one, attributed to the decoupling concept introduced in Section 5.4.3, we extend our analysis to compare different sequential PINN solvers to assess whether transfer learning leads to more accurate results. Our aim is to compare the following approaches:

1. RD-PINN: Randomly initializing each PINN solver for every PDE. Instead of transferring parameters from the PINN water solver to the ammonium PINN solver and repeating the process for the subsequent nitrogen species, we opt for random initialization for each PINN solver.

Transfer Learning PINN



Standard PINN

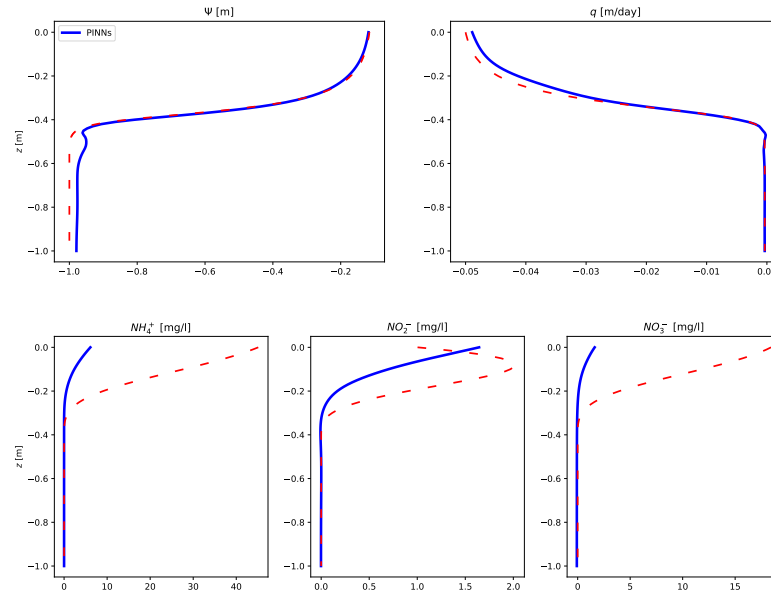


Figure 5.8 Pressure head, water flux and nitrogen species concentrations obtained using TL-PINNs and standard PINNs at time $T = 1$ day.

2. RD1-PINN: In this approach, parameters from the PINN ammonium solver are transferred to the PINN nitrite solver, while the solvers for ammonium and nitrate are randomly initialized instead of employing the transfer learning approach.

While there are other potential approaches to explore, our study specifically focuses on

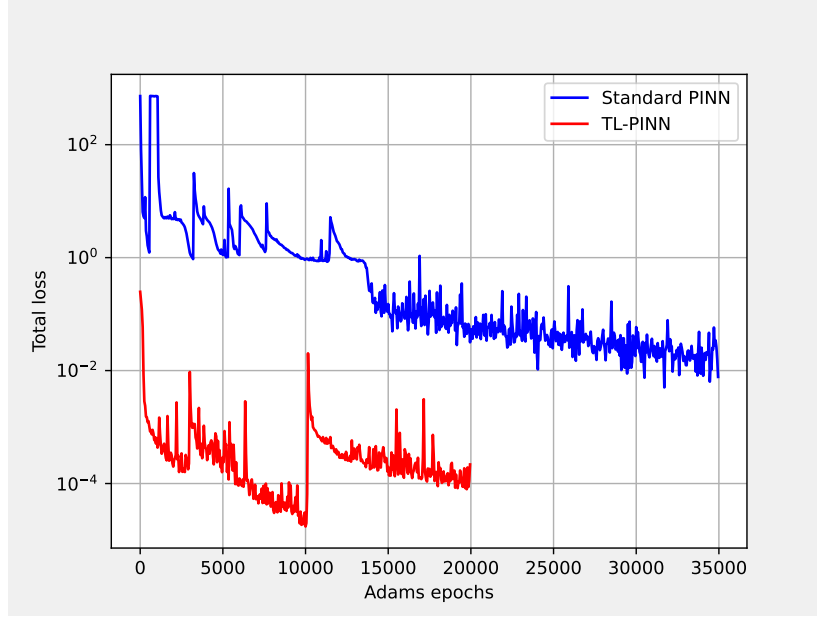


Figure 5.9 Total loss function of the standard PINN and the TL-PINN solver during Adam iterations.

the two methods mentioned above. It is important to note that the RD-PINN solver deviates from the standard PINN solver. The RD-PINN solver uses sequential training, whereas the standard PINN approach minimizes the loss of the coupled system of PDEs simultaneously.

Throughout this investigation, we aim to highlight the primary advantage of TL-PINNs in the context of decoupling the training of PINN solvers. The HYDRUS-1D software is used to generate a reference solution for calculating the relative L^2 -error for the solution predicted by the PINN solvers. The relative L^2 -error in the time-space domain $[0, T] \times [0, L]$ is defined as follows:

$$\tau^u = \frac{\sum_{t \in [0, T]} \sum_{z \in [0, L]} (\hat{u}(t, z) - u(t, z))^2}{\sum_{t \in [0, T]} \sum_{z \in [0, L]} u(t, z)^2}, \quad (5.20)$$

where $\hat{u}$ and u are the PINN and the HYDRUS-1D solutions, respectively.

In the simulations, the TL-PINN architecture and the number of iterations in the optimizers, as utilized in the previous section, are maintained for the other two approaches. The results in Figure 5.10 highlight the performance of TL-PINN, RD-PINN, and RD1-PINN. As anticipated, the TL-PINN solver emerges as the most efficient among the three. Randomly initializing all PINN solvers proves to be the least effective, resulting in a significant error for nitrogen species concentrations. Notably, in the RD1-PINN approach, random initialization of the PINN ammonium solver increases the error compared to the TL-PINN, which benefit from prior knowledge from the PINN water

solver. The RD1-PINN approach decreases the error of the PINN nitrite solver by leveraging knowledge from the PINN ammonium solver. However, the error increases again in the PINN nitrate solver due to its random initialization in the RD1-PINN approach. Figure 5.11 displays the loss function last values for ammonium and nitrite after training, illustrating the performance of TL-PINN and reinforcing the observations mentioned earlier. Once again, the TL-PINN approach demonstrates superior efficiency, emphasizing that decoupling the PDEs has a limited impact on accuracy unless previous knowledge from other solvers is utilized.

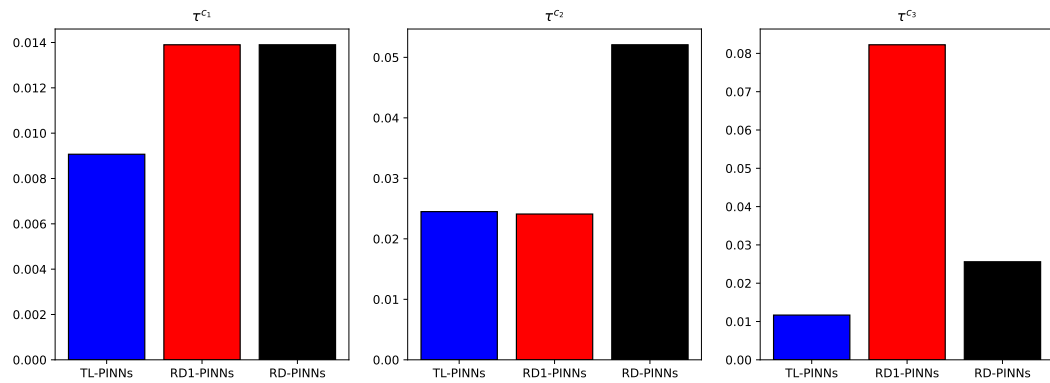


Figure 5.10 Relative L^2 -error of TL-PINN, RD-PINN, and RD1-PINN for each nitrogen species concentration.

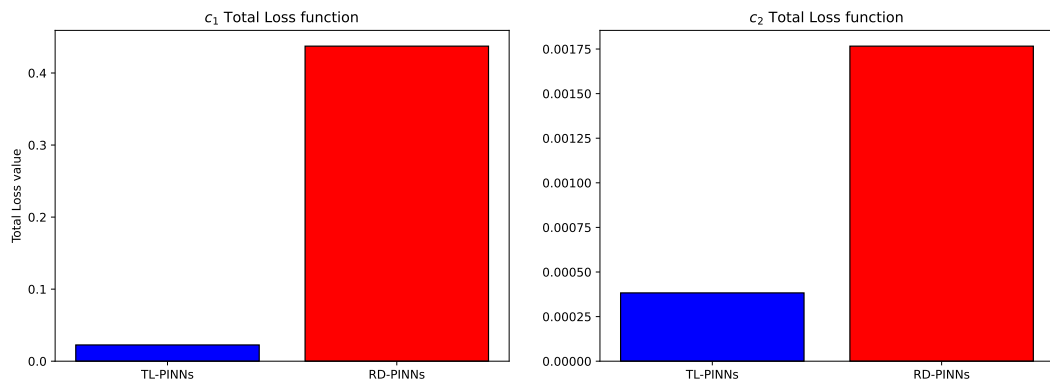


Figure 5.11 The total loss function values for c_1 and c_2 for TL-PINN and RD-PINN solvers. Ammonium and nitrite concentrations are denoted as c_1 and c_2 respectively.

5.5.3 Investigating the influence of FNN architecture and residual points number on TL-PINN solver performance

The performance of the TL-PINN solver is heavily influenced by the design of its FNN architecture. In this section, we explore how the number of hidden layers and neurons per layer in the FNN affect the solver’s accuracy. We also investigate the effects of varying the number of residual points used to calculate the residual losses.

To investigate the influence of the number of residual collocation points N_r , we examine its impact on the TL-PINN solver for 5 candidate values: 1000, 2000, 5000, 15000, and 20000. The TL-PINN solver architecture is set to 5 hidden layers and 50 neurons per layer. The number of iterations aligns with the specifications outlined in Section 5.4.5. Figure 5.12 displays the results for all desired solutions. Using fewer residual points led to a higher error across all variables. However, increasing the number of points beyond a certain threshold did not significantly improve the TL-PINN’s accuracy. This suggests that a fixed, large number of residual points might not be optimal for all cases. As proposed in (Lu *et al.*, 2021b), a residual-based adaptive refinement technique could potentially establish a more direct correlation between accuracy and the number of time-space points used. Figure 5.13 shows how the total loss function for each nitrogen species concentration in the TL-PINN solver evolves with different numbers of residual collocation points. Lower training loss is evident with a smaller number of residual collocation points, while higher numbers of residual collocation points can complicate the training phase.

The influence of the FNN architecture on the TL-PINN solver’s performance is investigated, exploring diverse configurations for the number of hidden layers (ranging from 1 to 5) and the number of neurons per layer (with options of 10, 20, and 50). Figure 5.14 displays the results for all the configurations evaluated. As expected, shallow FNNs with few neurons lead to high errors for all TL-PINN solvers. Interestingly, increasing the number of hidden layers beyond a certain point can negatively impact accuracy. The configuration with 50 neurons per layer and 5 hidden layers yields the optimal performance in this study.

5.5.4 Impact of the freezing technique on the transfer learning PINN

In this section, we investigate the impact of the freezing technique 5 on the accuracy of the TL-PINN solver. The parameters used for this solver remain the same as those employed previously. We recall that the hyperparameter k_{c_i} denotes the number of freezing layers counted from the first hidden layer in the PINN solver for solute c_i . To evaluate the influence of k_{c_i} on the accuracy of TL-PINNs, we utilized the relative L^2 -error (5.20), considering HYDRUS-1D as our reference solution. The results obtained are illustrated in Figure 5.15. For the PINN ammonium solver, the optimal number of hidden layers to freeze is $k_{c_1} = 3$; exceeding this threshold results in an increase in the error. Turning to the PINN nitrite solver, we observe the combined effect of

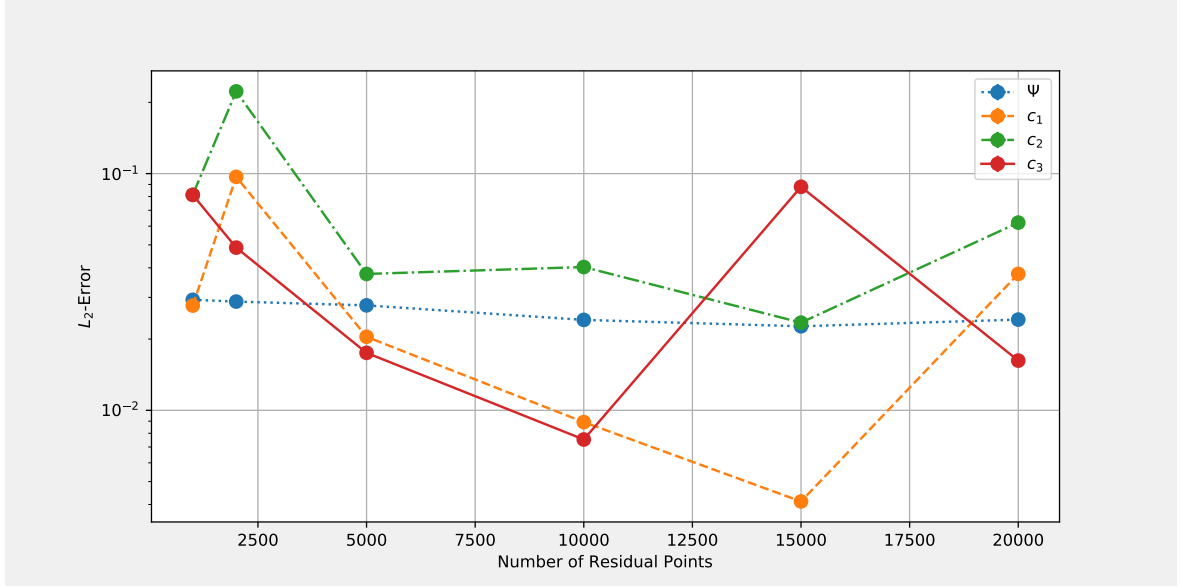


Figure 5.12 The relative L^2 -error as a function of the number of residual collocation points for each PDE solution (Ψ, c_i) .

parameters k_{c_1} and k_{c_2} on its accuracy. The optimal number of hidden layers to freeze is $(k_{c_1}, k_{c_2}) = (3, 4)$. Similar to the first solver, freezing half of the architecture proves to be the most suitable choice for training. Lastly, for the PINN nitrate solver, the combined effect of parameters k_{c_2} and k_{c_3} influences its accuracy. The optimal number of hidden layers to freeze is $(k_{c_2}, k_{c_3}) = (2, 3)$. As with the other solver, freezing half of the number of hidden layers is the most suitable choice for training. It can be concluded that, in the three solute PINNs solvers, freezing the first half of the hidden layers does not significantly affect the accuracy of TL-PINN. This is because these layers capture the general physical structure of the PDE. On the other hand, freezing the remaining half of the layers is not advisable, as these layers are essential for handling the initial and boundary conditions.

In conclusion, the freezing technique proves to be an efficient means of accelerating the training of TL-PINN, especially when the number of frozen hidden layers is selected based on a posterior error study, as demonstrated in this investigation. For our case, it is optimal to freeze 3 hidden layers for the the PINN ammonium solver, 4 for the the PINN nitrite solver, and 3 for the PINN nitrate solver.

5.5.5 Training stability in the transfer learning PINN

In this section, our objective is to monitor the loss of each PDE within the coupled system and observe the impact of knowledge transfer from one solver to the next. We employed the same configurations as in previous sections, training the PINN water solver for 5000 iterations and each PINN solute solver for 1000 iterations, totaling

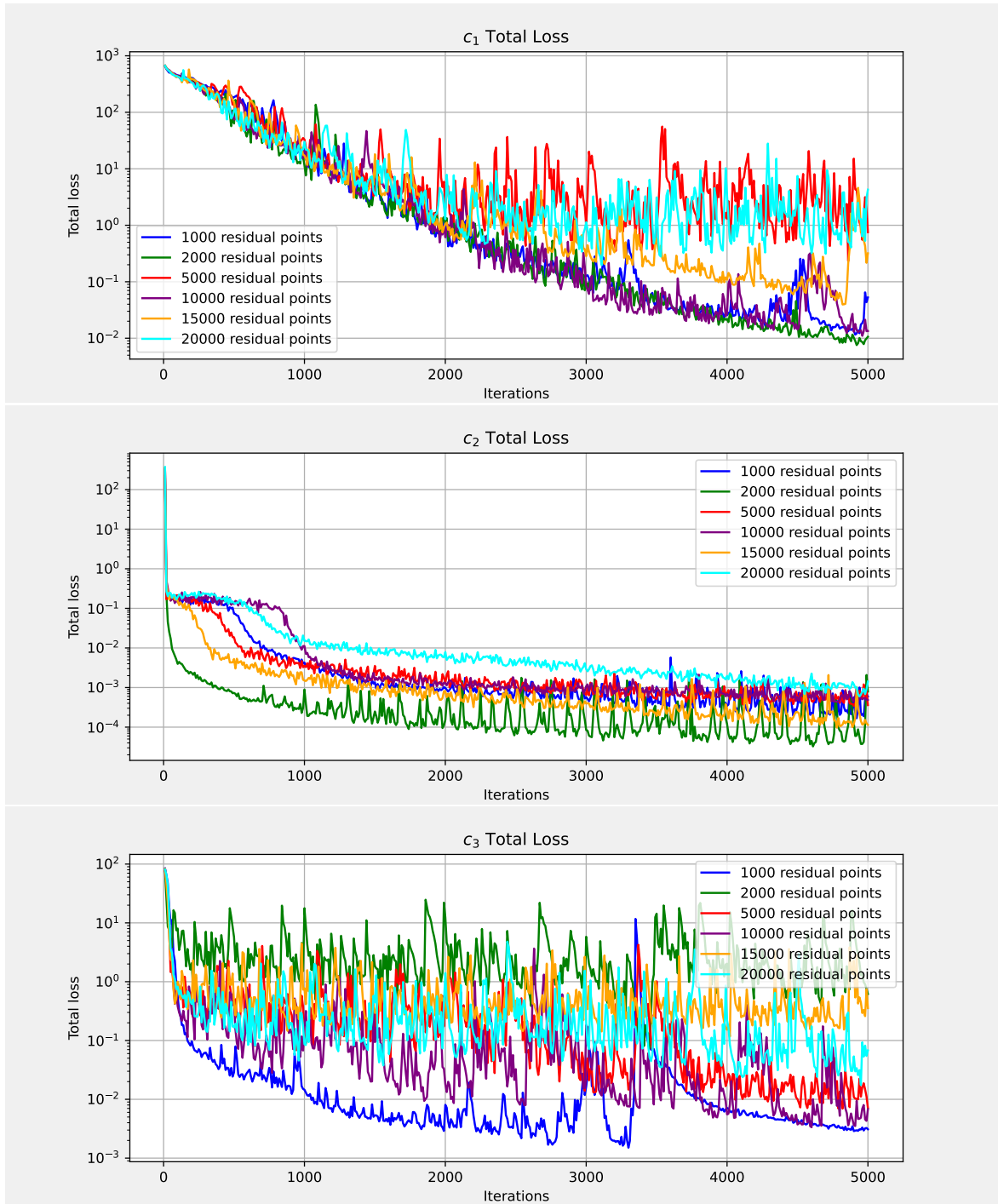


Figure 5.13 The c_1 , c_2 , and c_3 total loss function evolution as a function of the Adams iterations for different numbers of residual collocation points.

8000 iterations. We logged the loss for each Adam iteration and present the results in Figure 5.16, depicting the loss evolution for each PDE solver.

The training process begins with the Richards equation, incorporating its initial and

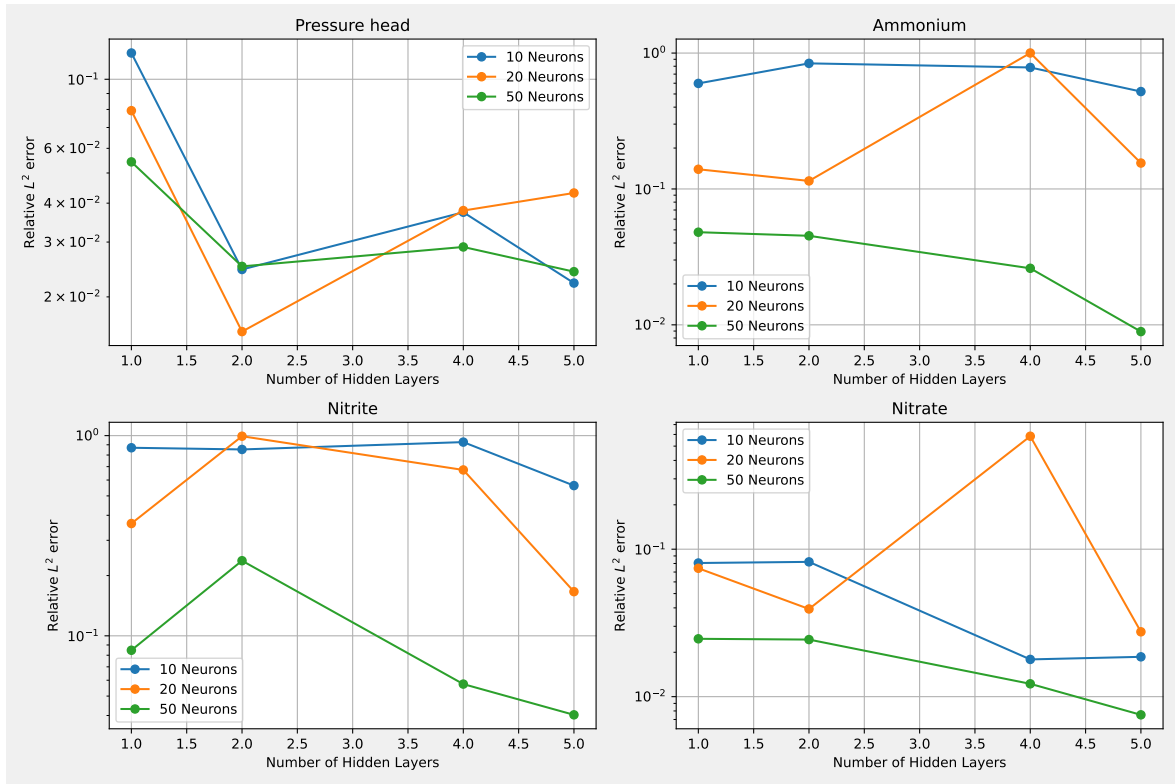


Figure 5.14 The relative L^2 -error with varying hidden layers and neurons for each PDE solution (Ψ, c_i) in the TL-PINN solver.

boundary conditions with a specific PINN water solver. After 5000 iterations, the solver achieves a training error of less than 10^{-4} . Subsequently, the knowledge acquired by the PINN water solver is transferred to the PINN ammonium solver. Initially, at the first iteration, a training error larger than 10^{-1} is observed. This discrepancy is expected since the parameters used in the Richards equation differ from those in the ammonium transport equation across all terms (residuals, initial, and boundary conditions). However, after 1000 iterations, the training error rapidly decreases to 10^{-3} , indicating that the knowledge gained from solving the Richards equation has significantly improved the training of the PINN ammonium solver. Similar observations hold for the other solute solvers, where lower training errors are achieved rapidly. All PDE losses achieved a training error of less than 10^{-4} within a total of 8000 iterations. We repeated this process with 10 different seeds for initializing the PINN water solver to account for stochasticity, and for all experiments, the training was stable, and no oscillations were observed. In summary, our extensive numerical simulations confirmed the stability of the proposed TL-PINN approach. The TL-PINN solver significantly boosts the efficiency of the PINN approach in modeling this system of PDEs. These findings are important and can be a subject of future theoretical analysis.

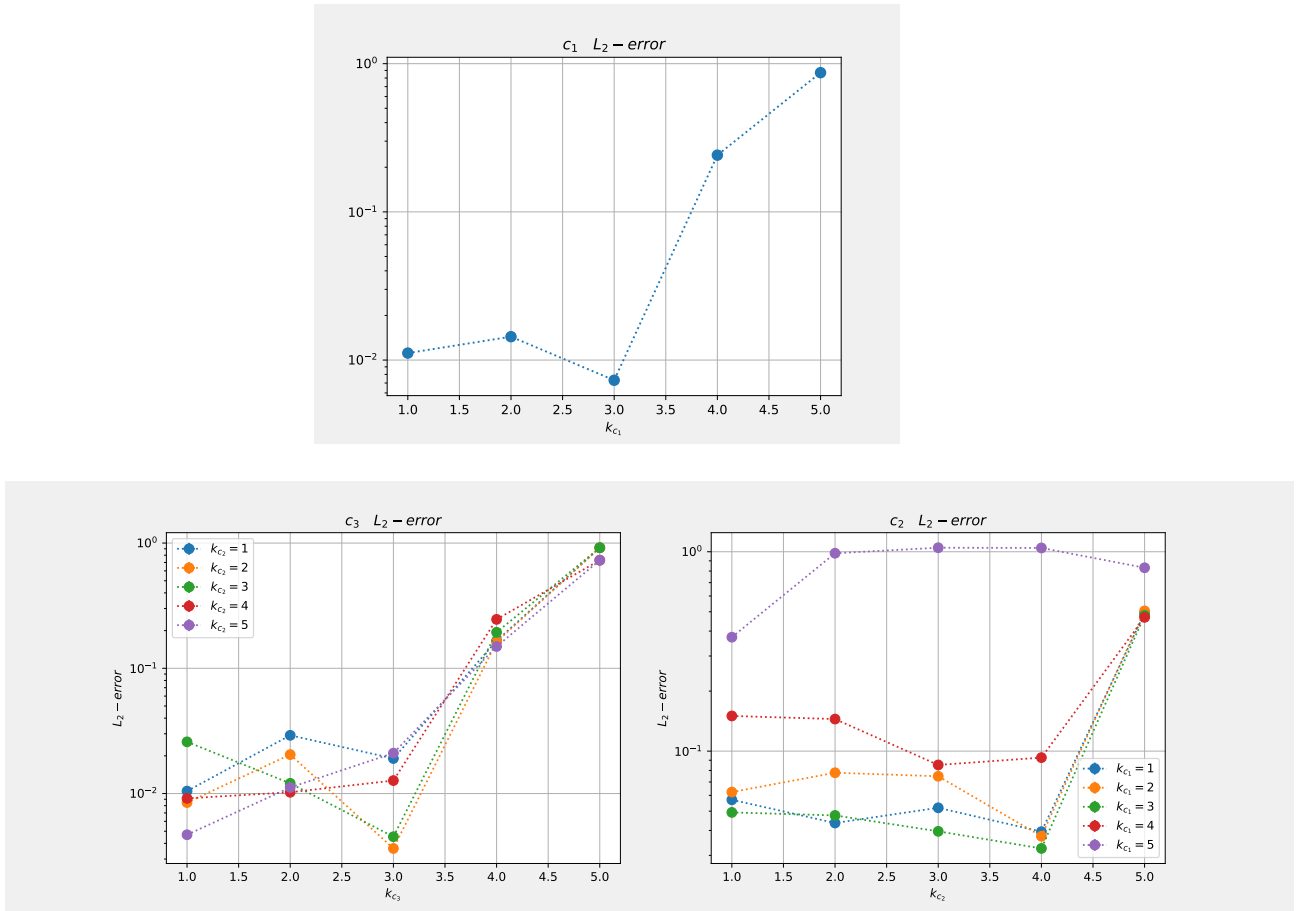


Figure 5.15 Influence of freezing hyper-parameters k_{c1} , k_{c2} , and k_{c3} on L_2 -error for ammonium (c_1), nitrite (c_2), and nitrate (c_3).

5.5.6 2D simulations using the transfer learning PINN solver

Training a PINN solver in higher dimensions may hinder the training process, especially for the coupled system examined in this study. To assess the reliability of the TL-PINN solver in addressing the infiltration process and nitrogen species dynamics within a 2D space geometry, we conducted a numerical experiment, illustrated in Figure 5.2. In this simulation, we used the same architecture as before, with 5 hidden layers for each solver. We trained the PINN water solver for 20,000 Adam iterations. Afterward, we conducted training for the nitrogen PINN solvers over 10,000 Adam iterations while keeping 3 hidden layers frozen. We also maintained consistency in the number of collocation points for both solvers, aligning with those employed in the 1D scenario. This approach allows us to evaluate the solvers' performance under the condition of sparse collocation points.

Figure 5.17 illustrates the final profiles of the pressure head (Ψ), ammonium (NH_4^+), nitrite (NO_2^-), and nitrate (NO_3^-) after 1 day. The application of the TL-PINN solver with the freezing technique has demonstrated effectiveness in this 2D scenario. The

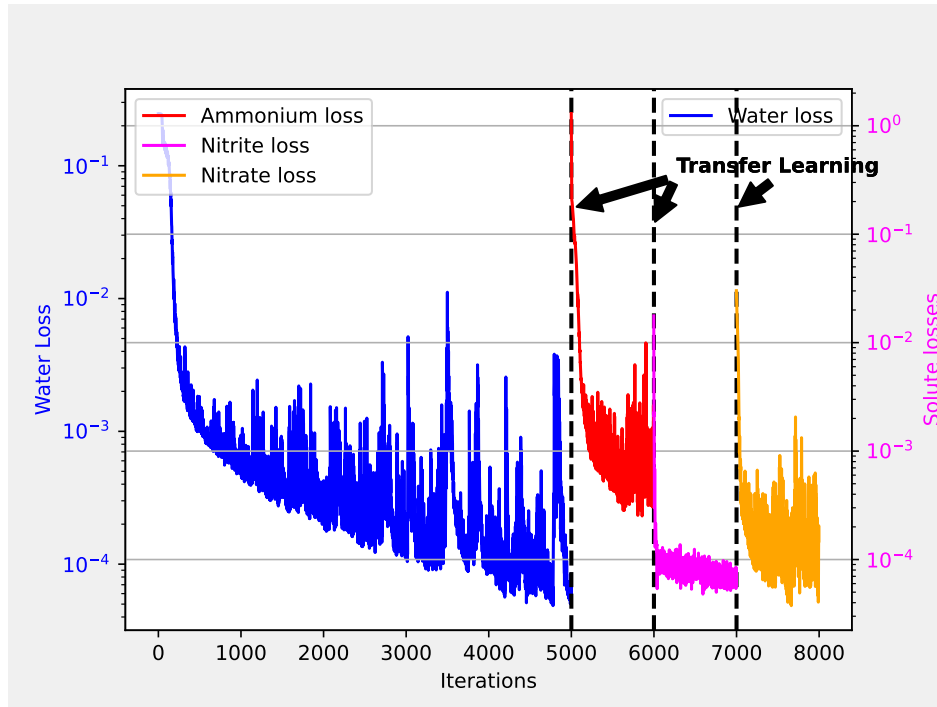


Figure 5.16 Evolution of total loss across Adam iterations for each PINN solver.

solver successfully maintains the negativity of the pressure head and the positivity of the other concentrations. This proves the reliability of the proposed transfer learning technique for solving the 2D problem.

5.5.7 Inverse problem with unknown initial and boundary conditions

The PINN approach has showcased remarkable efficacy in addressing inverse problems, as demonstrated in several studies (e.g., [Bandai and Ghezzehei \(2021\)](#)). In this section, we assess the performance of the proposed TL-PINN solver by addressing an inverse problem. Specifically, we aim to solve the coupled system of water flow and nitrogen species, where the initial and boundary conditions are unknown. This situation reflects the challenges encountered in practical applications, where accurately determining these conditions is often difficult. Here, we demonstrate how TL-PINN can efficiently address this inverse problem in a single training process, circumventing the need for multiple forward solutions typically required by standard numerical solvers, which can be impractical in real-world scenarios.

To guide the training of the TL-PINN solver in addressing this inverse problem, we incorporated limited measured data obtained using HYDRUS-1D software. In real-world scenarios, measuring nitrite concentrations is challenging due to its rapid conversion to nitrate ([Kim *et al.*, 2008](#)). Therefore, we impose initial and boundary conditions for nitrite, presuming these to be more manageable to enforce than directly using nitrite

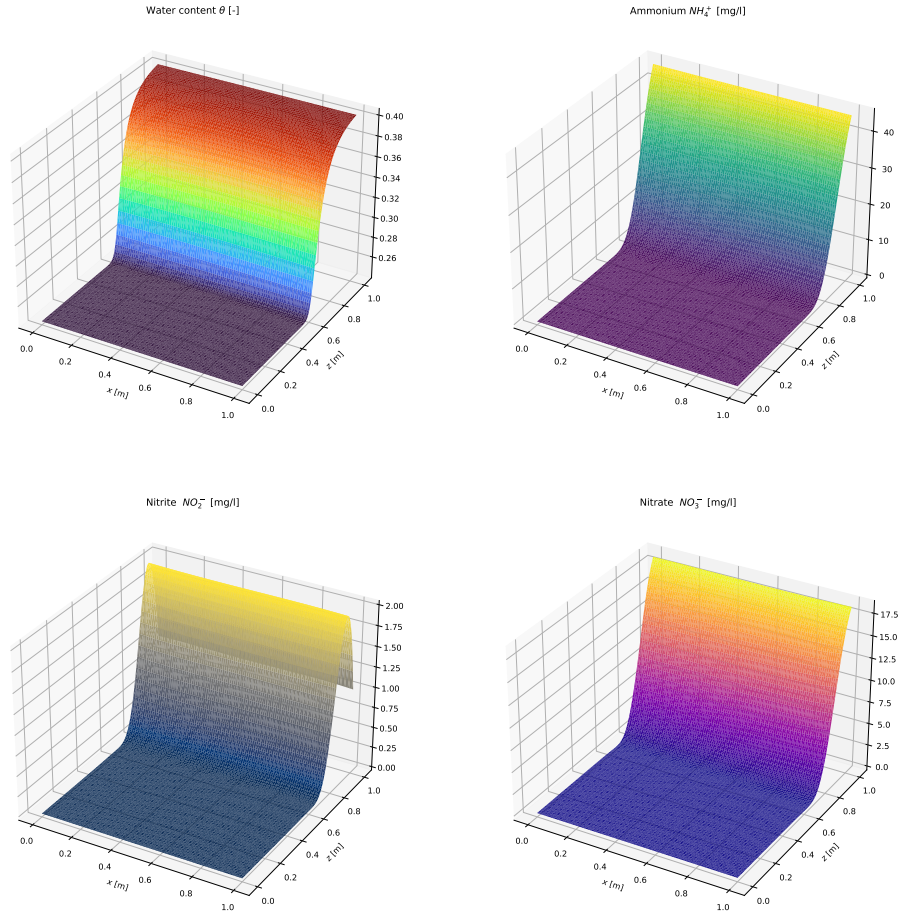


Figure 5.17 2D profiles of water content, ammonium, nitrite, and nitrate in the time-space domain $\{T = 1\} \times [0, 1]^2$.

concentration data. Two scenarios are examined and illustrated in Figure 5.18. In scenario 2, we decrease the number of measured data points utilized for training the TL-PINN solver to assess the impact of the number of measured data on its accuracy. We keep the parameters used in the forward problem for the nitrite PINN solver. For the other solvers, we adjust the weight parameters in the total loss function as follows: $\alpha_r = \alpha_d = 1$, and $\alpha_{ic} = \alpha_b = 0$. Moreover, we utilized the frozen settings as outlined in Section 5.5.4.

The profiles obtained for scenario 1 are depicted in Figure 5.19, along with the corresponding absolute error. The TL-PINN solver demonstrated promising results for

solving the inverse problem.

We aim to investigate whether incorporating physical constraints into the loss function enhances the results in this inverse problem for the TL-PINN solver. Therefore, we solve the same problem using a standard feed-forward neural network (FNN) and compare its results to the TL-PINN solver. This comparison is limited to the Richards equation, and the obtained results for both scenarios are illustrated in Figure 5.20. The PINN solver demonstrates enhanced accuracy and offers reliable predictions for both the pressure head and the water flux. In contrast, the FNN yields inaccurate predictions, particularly noticeable for the water flux near the soil surface, where it's already unknown. Based on these results, we conclude that the PINN approach outperforms the standard FNN solver in accuracy and reliability.

To evaluate the TL-PINN solver's robustness in handling noisy measurement data, we train it using modified outputs from HYDRUS-1D as follows:

$$u_{\text{train}} = u + \sigma \mathcal{N}(0, 1), \quad (5.21)$$

where u_{train} represents the training data, including pressure head and concentrations of ammonium and nitrate, u denotes the corresponding HYDRUS-1D data, and $\mathcal{N}(0, 1)$ denotes Gaussian noise with magnitude σ . Four magnitudes are utilized herein: $\sigma \in \{0, 0.005, 0.01, 0.1\}$.

The obtained results are presented in Table 5.1, showcasing the relative L^2 -error after five different initializations of the solver. Remarkably, the TL-PINN solver demonstrates good accuracy across varying magnitudes of noise (σ). Even in scenario 2, where fewer measured points are utilized, the TL-PINN solver maintains good results. It is evident that increasing the magnitude of noise leads to higher errors, which is expected since the TL-PINN solver lacks integration of techniques to handle highly noisy data, such as dropout or a Bayesian approach. Nevertheless, despite this limitation, the TL-PINN solver consistently delivers good accuracy for both scenarios and can realistically be used for noisy data.

5.5.8 Simulations using the transfer learning mixed-PINN

Here, we aim to numerically investigate the mixed-PINN approach developed in Section 5.4.4 for heterogeneous soils within the transfer learning context. To validate our approach, we used numerical tests that incorporate a discontinuity between the physical parameters of two soil layers to include an aspect of heterogeneity. In the first test scenario, the soil domain consists of two layers, where the first layer is loamy soil, and the second layer is characterized by sandy loam with the following parameters (Bandai and Ghezzehei, 2022):

$$\text{Loamy layer: } K_s = 0.2496 \text{ m/day}, \alpha_v = 3.6 \text{ m}^{-1}, \theta_s = 0.43, \theta_r = 0.078, n_v = 1.56, \quad (5.22)$$

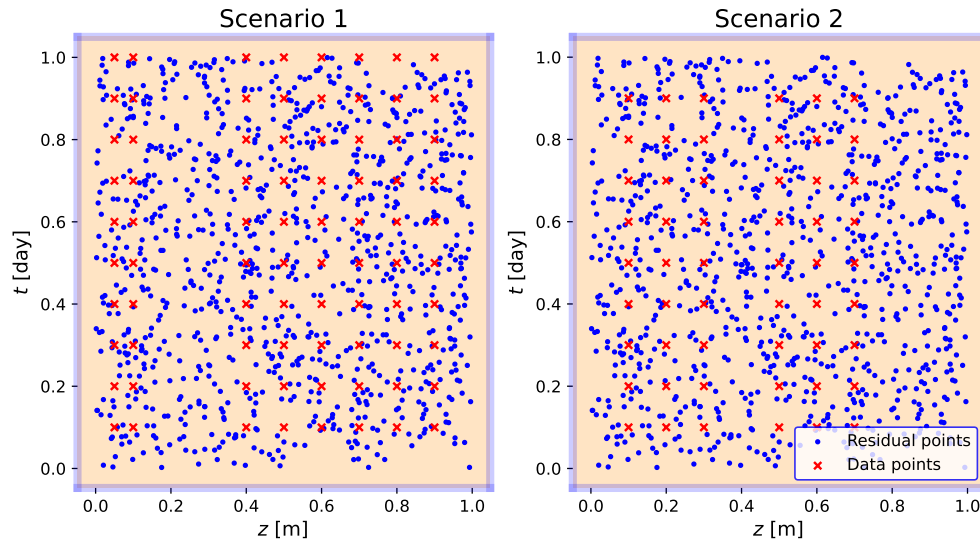


Figure 5.18 The two studied scenarios in the inverse problem.

Profiles of the obtained solutions

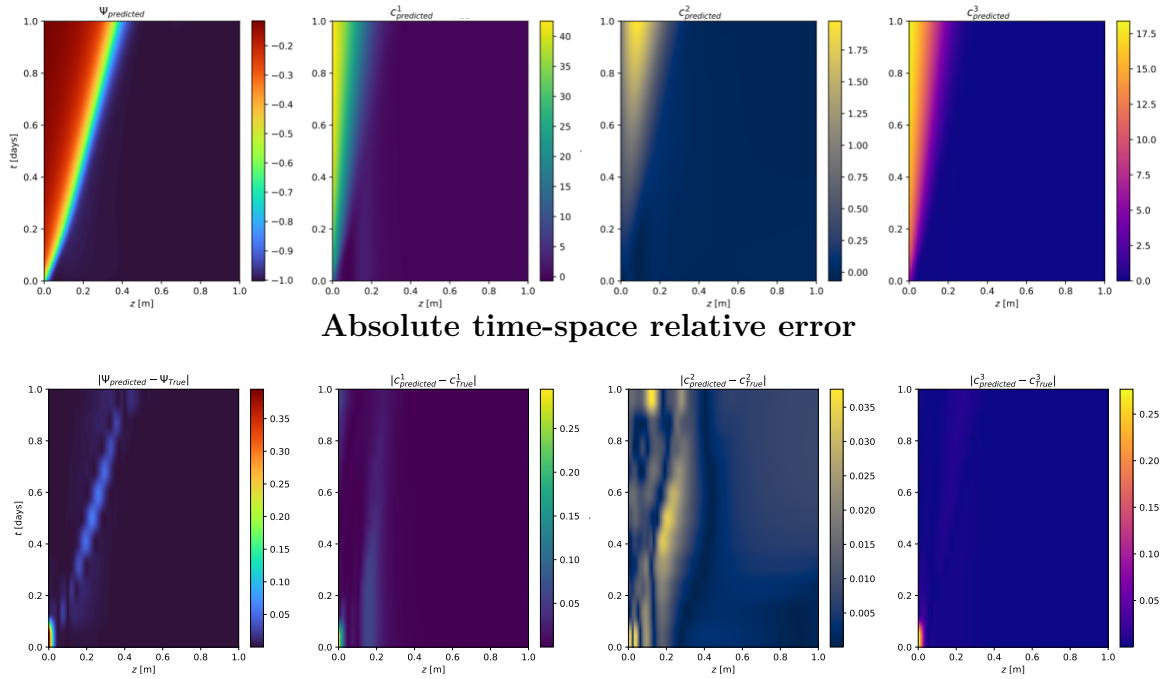
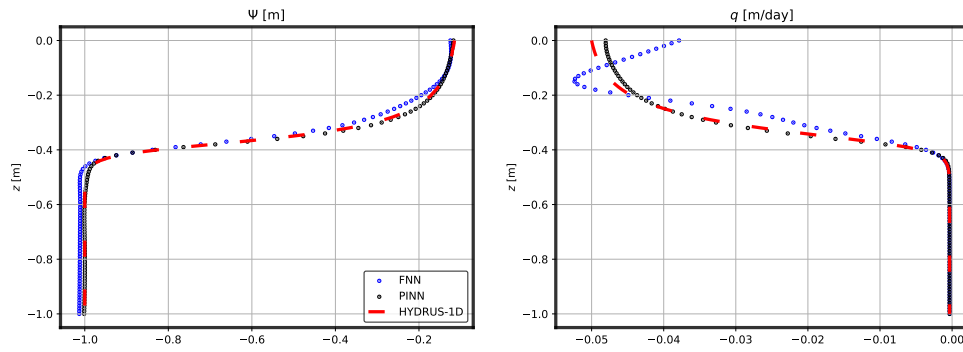


Figure 5.19 Inverse problem-scenario 1: Profiles (top), errors (down) of pressure head and nitrogen species concentrations obtained using TL-PINN in the time-space domain $[0, 1] \times [0, 1]$.

$$\text{Sandy loam layer: } K_s = 1.061 \text{ m/day}, \alpha_v = 7.5 \text{ m}^{-1}, \theta_s = 0.41, \theta_r = 0.065, n_v = 1.89. \quad (5.23)$$

Scenario 1



Scenario 2

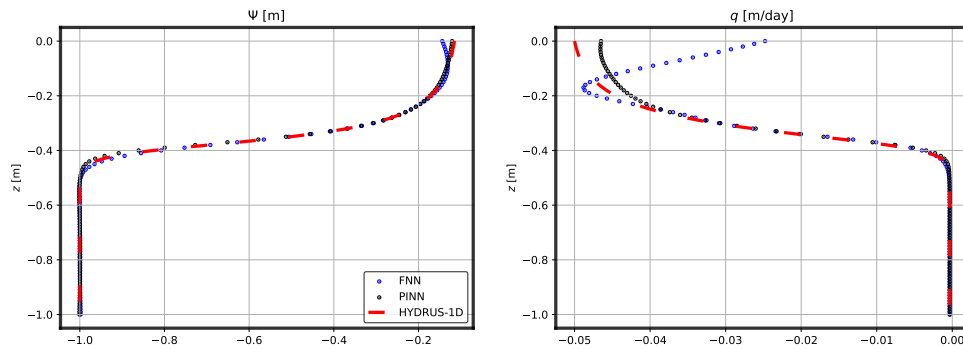


Figure 5.20 Inverse problem test: Profiles of pressure head and water flux obtained using a PINN and a standard FNN at $T = 1$ Day after 2000 Adams epochs.

The loamy and sandy loam layers each have a thickness of 0.5 m. For the mixed-PINN architecture, we utilized two neural networks, as described in Section 5.4.4, to solve each PDE. The neural networks have an architecture consisting of five hidden layers, each containing 50 neurons. This architecture is utilized for all PINN solute solvers. The training parameters include 50,000 epochs using the Adam optimizer for the PINN water solver and 20,000 epochs for the PINN solute solvers. The simulation is performed using 20,000 collocation points in the time-space domain. The collocation points are generated using the Halton sequence as the sampling strategy to improve the accuracy and the training process (Song *et al.*, 2023).

The obtained results are depicted in Figure 5.21-(a)-(b). We present the pressure head, water content, water flux, and concentration profiles of the ammonium species over the time-space domain $[0, 2 \text{ days}] \times [-1, 0 \text{ m}]$ as well as the reference solution using HYDRUS. The proposed transfer learning mixed-PINN effectively captures the solution for layered soils with large discontinuity in terms of the physical properties of the medium. The water flux and solute concentrations are accurately predicted, with errors less than 0.5% for water flux and 0.43% for solute concentrations. However, we

Table 5.1 Relative L_2 -error for the inverse problem with different magnitude of noisy data used for the training.

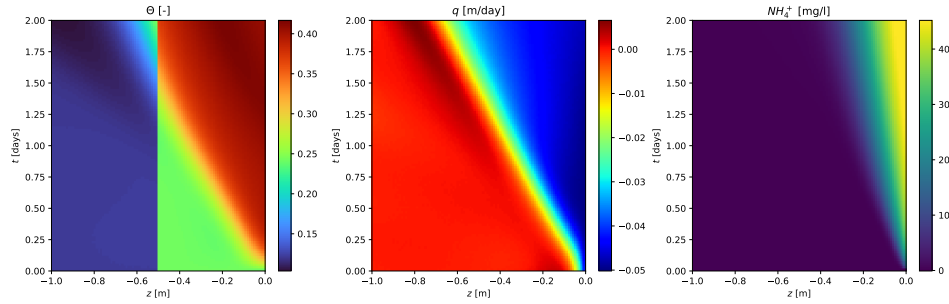
Scenario 1				
σ	0	0.005	0.01	0.1
Ψ	$0.0051 \pm 8.37 \times 10^{-9}$	$0.0055 \pm 1.29 \times 10^{-8}$	$0.051 \pm 4.83 \times 10^{-8}$	$0.18 \pm 1.49 \times 10^{-6}$
q	$0.0093 \pm 3.13 \times 10^{-7}$	$0.00987 \pm 9.92 \times 10^{-8}$	$0.0703 \pm 4.9 \times 10^{-9}$	$0.11 \pm 5.16 \times 10^{-6}$
c_1	$0.003 \pm 9.45 \times 10^{-6}$	$0.0098 \pm 2.55 \times 10^{-9}$	$0.098 \pm 4.22 \times 10^{-11}$	$0.099 \pm 3.06 \times 10^{-10}$
c_2	$0.007 \pm 9.0 \times 10^{-10}$	$0.0099 \pm 2.024 \times 10^{-9}$	$0.0776 \pm 1.44 \times 10^{-8}$	$0.101 \pm 2.26 \times 10^{-8}$
c_3	$0.008 \pm 2.94 \times 10^{-6}$	$0.0097 \pm 1.96 \times 10^{-10}$	$0.045 \pm 2.25 \times 10^{-10}$	$0.0983 \pm 1.6 \times 10^{-9}$
Scenario 2				
σ	0	0.005	0.01	0.1
Ψ	$0.0055 \pm 1.32 \times 10^{-10}$	$0.005515 \pm 1.9 \times 10^{-10}$	$0.05125 \pm 7.56 \times 10^{-8}$	$0.23 \pm 2.30 \times 10^{-7}$
q	$0.00976 \pm 1.44 \times 10^{-8}$	$0.01 \pm 2.862 \times 10^{-9}$	$0.092 \pm 2.024 \times 10^{-7}$	$0.104 \pm 2.66 \times 10^{-6}$
c_1	$0.0098 \pm 2.25 \times 10^{-10}$	$0.009865 \pm 2.1025 \times 10^{-10}$	$0.098 \pm 1.225 \times 10^{-9}$	$0.186 \pm 4.224 \times 10^{-9}$
c_2	$0.0097 \pm 5.953 \times 10^{-8}$	$0.0998 \pm 1.40624 \times 10^{-9}$	$0.03 \pm 8.0999 \times 10^{-9}$	$0.1 \pm 4.224 \times 10^{-9}$
c_3	$0.0096 \pm 2.499 \times 10^{-11}$	$0.00974 \pm 2.7225 \times 10^{-10}$	$0.09 \pm 4.00 \times 10^{-10}$	$0.0972 \pm 3.024 \times 10^{-9}$

observe a slight difference at the final time $T = 2$ days at the wetting front for the pressure head where the mixed-PINN deviates slightly from the HYDRUS reference solution. The relative L^2 -errors for the pressure head and water content, which are related by the van Genuchten formula (8.4a), are 2.5% and 1.7% respectively. This slight deviation is mainly due to the large discontinuity between the saturated hydraulic conductivities K_s of the two soil layers where a factor is around 4.4. In the second numerical test, we perform simulations using an average amplitude discontinuity of factor 2 where we consider the same soil properties of the two layers except for the saturated hydraulic conductivity of the second layer, we set: $K_s = 2 \times 0.2496$ m/day. The results are depicted in Figure 5.21-(c). The mixed-PINN approach performs very well, achieving good accuracy for the water content, pressure head, and all solute concentrations. While increasing the number of collocation points can improve the simulation results of the mixed-PINN method, additional efficient treatment of high discontinuities of the physical properties of the medium can be adopted to improve the method. Some ideas from shock-capturing techniques usually used in the finite volume framework (Beljadid and LeFloch, 2017; Beljadid *et al.*, 2019; Sonntag and Munz, 2014), can be incorporated in the proposed approach in future studies.

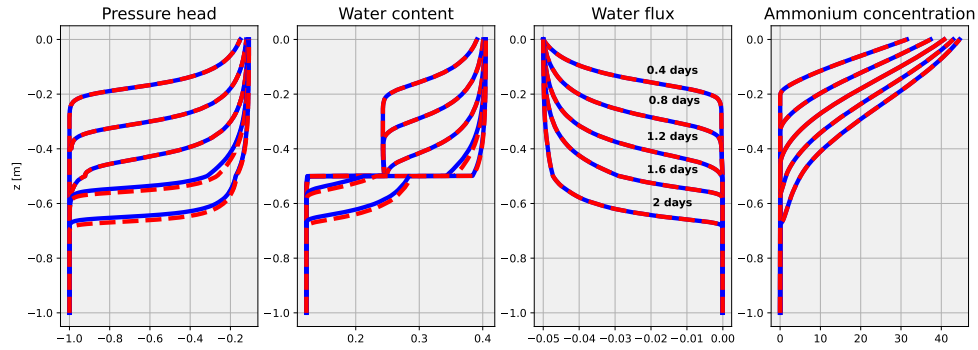
Transfer learning played a crucial role in accelerating the training of the PINN solute solver, achieving convergence in less than 5000 iterations while maintaining residuals below 10^{-4} . On the other hand, the standard PINN approach (Raissi *et al.*, 2019), even when initialized with transfer learning, failed to provide accurate results and exhibited unstable training behavior under the same conditions (results not shown here). These findings underscore the efficiency of the proposed mixed-PINN combined with the transfer learning approach for modeling water flow and multiple solute transport in

layered soils.

(a) Profiles of the obtained solutions for Scenario 1



(b) Cross-sections of the obtained solutions for Scenario 1



(c) Cross-sections of the obtained solutions for Scenario 2

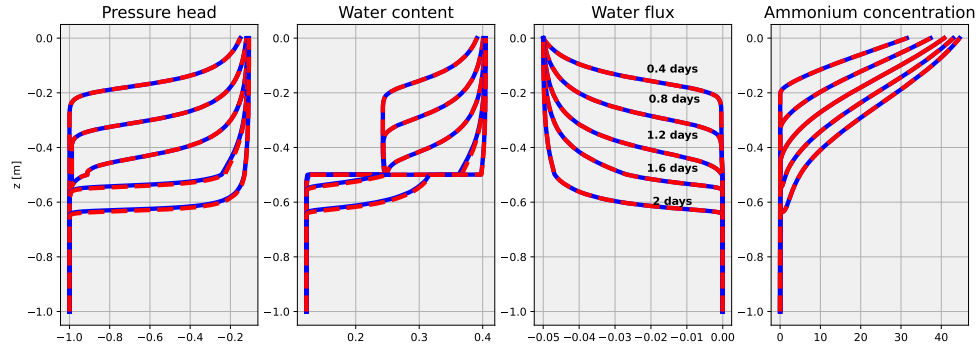


Figure 5.21 Layered soil test: Profiles (top) and cross-sections (bottom) of water content, water flux, and ammonium concentration obtained using the transfer learning mixed-PINNs in the time-space domain $[0, 2 \text{ days}] \times [-1, 0 \text{ m}]$. The results are compared with HYDRUS, represented by red dashed lines. Panel (b) depicts scenario 1 using the soil parameters detailed in (5.23), while panel (c) presents the second scenario in which the discontinuity between the two layers is reduced by adjusting the saturated hydraulic conductivity as follows: $K_s^{\text{layer } 2} = 2 \times K_s^{\text{layer } 1}$.

5.6 Limitations and future objectives

Here, we comprehensively explore various aspects, encompassing advantages and limitations, as well as proposed enhancements that could be integrated into the transfer learning technique proposed in this study.

The TL-PINN approach excels by exploiting the sequential nature of the governing equations. This enables the efficient solution of multiple solute transport equations, avoiding the need for complex coupling terms in the training process compared to solving the entire system simultaneously. Given that the PINN constitutes the foundation of the TL-PINN solver, enhancing efficiency can be realized through the integration of various improvements into the PINN architecture or training methodology. For example, one approach to enhance efficiency involves dynamically adapting the global weights associated with each term in the loss function, as proposed in (Chen *et al.*, 2023b). Additionally, local adaptation of weights associated with individual collocation points can be achieved through methods such as the self-adaptive PINN (McClenny and Braga-Neto, 2023) or the residual-based attention PINN (Anagnostopoulos *et al.*, 2024). These techniques will be valuable for obtaining more accurate solutions of the Richards equation, especially in regions where the solution exhibits sharp fronts. By directing the solver to focus more attention on these areas, higher spatial resolution can be achieved, thereby improving overall accuracy.

A major drawback of PINNs for real-time applications is their long training time. This becomes even more problematic when changing initial or boundary conditions, as it requires restarting the entire training process. Consequently, this limits the practical use of PINNs in situations where quick solution updates are essential. This disadvantage is also common to standard numerical methods. Two efficient solutions can be adopted: (1) utilize meta-learning techniques, where a PINN solver is trained on multiple tasks in the offline stage to enable solving new tasks in the online stage, even if they do not belong to the task distribution used in meta-training (Chen *et al.*, 2021a); (2) Consider the deep operator learning approach, which leverages the neural network's capability to approximate any operator between two infinite spaces (Jin *et al.*, 2022; Lu *et al.*, 2021a). In the context of the coupled water flow and solute transport model, this method allows learning the operator that connects the initial and boundary conditions to the solutions.

It is important to note that our study focuses on a one-way coupling system, assuming that solute transport does not influence soil water movement (Nielsen *et al.*, 1986). However, two-way coupling can be considered in some cases (Illiano *et al.*, 2021), where a modified model is used. The proposed transfer learning technique may be extended for these cases where more challenging aspects should be addressed such as the convergence and computational cost of the algorithm due to the fully coupled highly nonlinear nature of the system of PDEs.

5.7 Conclusion

This study introduces a transfer learning physics-informed neural network (TL-PINN) approach for solving water infiltration and nitrogen species transport in unsaturated soils. The approach involves decoupling the fully coupled system by initially solving the Richards equation. The optimal parameters obtained by minimizing the associated loss function are then transferred to initialize the next PINN solver. This process is repeated until the last PINN solver for the corresponding transport equation is reached. To enhance the training of the PINN solver, various strategies were employed, encompassing the adaptive activation technique, mini-batch processing, and a learning rate schedule. To optimize the computational cost of the TL-PINN solver, a freezing technique was proposed. It is worth noting that alternative physics-informed machine learning solvers can be considered to replace the PINN approach.

A 1D numerical experiment was conducted to assess the proposed methodology, and was subsequently extended to a 2D case. The results indicate that a TL-PINN effectively captures all the desired solutions, including pressure head, water flux, and nitrogen species concentrations. In contrast, the standard PINN solver struggled and failed to converge to the exact solution due to the high nonlinearity of the loss functions associated with different PDEs and their respective initial and boundary conditions. The success of the decoupling approach in the TL-PINN solver is apparent in its effective training. Another test was performed, to evaluate the impact of transferring parameters from the highly nonlinear Richards equation to the linear advection-dispersion equations. The results highlight the necessity of this technique, showcasing its superior accuracy compared to randomly initialized PINN solvers. The freezing technique was also evaluated, and the results indicate that freezing half of the architecture of the PINN is the most suitable choice. The effectiveness of the TL-PINN solver was further validated through its application in a 2D scenario, an inverse problem with unknown initial and boundary conditions, as well as a layered soils. For efficient training of the TL-PINN solver under discontinuities of the physical properties of the medium, we proposed the mixed-PINN approach, which provides accurate results under these conditions.

The proposed approach prioritizes accurate water flow modeling by dedicating the majority of iterations to optimizing the total loss of the Richards equation. This strategy enables us to capture crucial solute concentration profiles with fewer overall iterations, making it particularly well-suited for complex scenarios involving multiple solutes. Our future endeavors will include evaluating the TL-PINN solver in real-world scenarios, encompassing field experiences to assess various irrigation systems and fertigation strategies. Training a PINN solver may prove challenging in cases with time-dependent boundary conditions. The objective is to advance the development of more efficient PINN solvers to achieve efficiency comparable to standard numerical solvers. Furthermore, considering additional factors such as soil deformation ([Schrefler](#)

et al., 1994) is a promising avenue for future research, to enhance the transfer learning PINN approach.

The next chapter is devoted to generalize the S-PINN and TL-PINN approaches by developing a unified framework for one-way coupled systems of PDEs, with a particular focus on water infiltration and multi-species solute transport problems.

Code availability

The source code used in this study is publicly available from the GitHub repository <https://github.com/Hamza-Kamil/Transfer-Learning-PINN-Nitrogen>.

Appendix. The mixed-PINN training procedure

Here, we provide a detailed description of the training procedure for the mixed-PINN approach discussed in Section 5.4.4. The formulation is based on the Richards equation and can be extended similarly to solute transport equations. Let the solution and flux networks be denoted as $\mathcal{N}_\Psi$, $\mathcal{N}_{q_x}$, and $\mathcal{N}_{q_z}$. Here, $\mathcal{N}_\Psi$ represents the surrogate for the pressure head, while $\mathcal{N}_{q_x}$ and $\mathcal{N}_{q_z}$ represent the water flux in the x and z directions, respectively.

After randomly sampling collocation points within the time-space domain, the following loss functions are derived:

$$\begin{cases} J_r(\Theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \left(\hat{f}_R(t_i^r, \mathbf{x}_i^r) \right)^2, \\ J_c(\Theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \left(\hat{f}_x(t_i^r, \mathbf{x}_i^r) \right)^2 + \left(\hat{f}_z(t_i^r, \mathbf{x}_i^r) \right)^2, \end{cases} \quad (5.24)$$

where $J_r(\Theta)$ is the PDE residual loss and $J_c(\Theta)$ represents the compatibility loss. These terms are defined as:

$$\begin{cases} \hat{f}_R = \frac{\partial \theta(\mathcal{N}_\Psi)}{\partial t} + \nabla \cdot \begin{bmatrix} \mathcal{N}_{q_x} \\ \mathcal{N}_{q_z} \end{bmatrix}, \\ \hat{f}_x = \mathcal{N}_{q_x} + K(\mathcal{N}_\Psi) \frac{\partial \mathcal{N}_\Psi}{\partial x}, \\ \hat{f}_z = \mathcal{N}_{q_z} + K(\mathcal{N}_\Psi) \left(\frac{\partial \mathcal{N}_\Psi}{\partial z} + 1 \right). \end{cases} \quad (5.25)$$

These formulations are for a two-dimensional spatial problem. For a one-dimensional spatial setup, only one network for the flux ($\mathcal{N}_{q_z}$) is needed, while in a three-dimensional setup, three networks would be required.

The total loss for the mixed-PINN is defined similarly to the total loss equation (5.16), where the residual term is replaced by $J_r(\Theta)$ defined in equation (5.24), and the compatibility loss is added as well.

CHAPTER 6 A comprehensive analysis of physics-informed neural networks for solving one-way coupled problems

Abstract

Physics-informed neural networks (PINNs) offer efficient solvers for differential equations. However, their performance often degrades when applied to complex problems, particularly in multiphysics modeling. In this study, we focus on one-way coupled models, where each equation depends only on the outputs of preceding ones. Such structures frequently arise in thermo-mechanics coupling, fluid–structure interaction, electro-thermal processes, and multispecies transport.

We examine why standard PINNs struggle on one-way coupled systems. When all equations are trained simultaneously, the network tends to ignore the inherent one-way causal structure due to spectral bias, leading to biased learning and poor convergence. We analyze these issues using the neural tangent kernel, the loss landscape, the distribution of back-propagated gradients, gradient conflict, and the resulting ill-conditioning. To address these challenges, we introduce PINN strategies that explicitly preserve the one-way causal order, including sequential training, transfer learning, and other causality-aware variants.

Numerical experiments on benchmark problems demonstrate that the proposed approaches preserve one-way causality and improve both accuracy and training efficiency compared to standard PINNs.

Scientific machine learning (SciML) is an interdisciplinary field that integrates machine learning with scientific computing to solve complex problems governed by ordinary and partial differential equations (ODEs and PDEs), which represent the physical laws underlying natural phenomena. Since closed-form solutions typically exist only in idealized scenarios, researchers have long relied on numerical methods to obtain approximate solutions. However, solving such problems remains challenging due to factors such as ill-posedness, the curse of dimensionality, chaotic dynamics, and the coupling of multiphysics and multiscale processes. Numerical methods are generally classified into two main categories: mesh-based and mesh-free approaches.

Mesh-based solvers, such as the finite element method (FEM), finite volume method (FVM), and finite difference method (FDM), rely on discretizing the computational domain into small elements, where the solution is approximated at specific nodes. These methods are effective in many scenarios but can become computationally expensive when applied to high-dimensional problems or parametric settings involving uncertainty quantification and model calibration. In contrast, mesh-free solvers avoid the need for mesh generation by selecting discrete points within the computational domain and computing the solution at those points. Recently, physics-informed neural networks (PINNs) (Raissi *et al.*, 2019) have emerged as efficient mesh-free solvers that approximate the solution of a given equation by formulating a multiobjective optimiza-

tion problem. By leveraging the expressive power of deep neural networks, PINNs can directly solve both linear and nonlinear PDEs without relying on iterative linearization or traditional discretization schemes.

PINNs have demonstrated significant success in solving PDEs for single-physics problems (see, e.g., (Wang *et al.*, 2023a; Mullins *et al.*, 2025; Wang *et al.*, 2024b)). However, their use in multiphysics settings, where several coupled physical processes interact to form a complete system, remains limited. Multiphysics problems are widespread in real-world applications, especially in subsurface soil systems, where different mechanisms occur simultaneously. For example, water infiltration in unsaturated soils can be modeled using the classical Richards equation (Richards, 1931) or extended formulations such as the BCJ-R equation (Equation 18 in (Beljadid *et al.*, 2020)) developed by Beljadid, Cueto-Felgueroso, and Juanes (Beljadid *et al.*, 2020). In practice, water infiltration interacts with additional processes, such as heat transfer or multispecies solute transport, each governed by its own PDE. Capturing the complete behavior of such systems therefore requires solving these equations in a coupled framework.

Exact solutions for multiphysics problems are generally limited to simplified scenarios, making efficient numerical solvers essential for addressing more complex cases. Even with established numerical methods, solving multiphysics problems is challenging due to the nonlinear and potentially degenerate nature of the governing PDEs. Traditional numerical solvers typically adopt one of two main strategies (Soulaïmani *et al.*, 2005): monolithic schemes that solve the coupled system simultaneously, and sequential (also referred to as segregated or splitting) methods that decouple the system and solve it iteratively, effectively treating it as a one-way coupled problem (Kim *et al.*, 2011; Soulaïmani *et al.*, 2005). This sequential perspective motivates the design of special PINN solvers: by adapting the one-way coupling strategy from classical numerical methods, we can train neural networks in a manner that respects the inherent directional dependencies of the system, improving convergence and stability compared to simultaneous training of all equations.

Building on this perspective, we propose a family of PINN models for one-way coupled multiphysics systems. Building on existing sequential approaches, including the sequential PINN (S-PINN) (Kamil *et al.*, 2025; Wang *et al.*, 2023b) and the transfer learning PINN (TL-PINN) (Kamil *et al.*, 2024d), we develop new training strategies: the smooth sequential PINN (2S-PINN), the backward compatible PINN (BC-PINN), its smooth variant, and the one-way causal PINN (OC-PINN). These models explicitly enforce one-way causality within the loss function during training. Numerical results show that preserving the one-way causal structure improves stability and computational efficiency for such systems.

The rest of the paper is structured as follows. In Section 6.1, we discuss coupled systems, with a focus on one-way coupling. Section 6.2 presents the motivation and key challenges of this study. Section 6.3 reviews related work in the context of physics-informed learning. Section 6.4 describes the proposed PINN models tailored to one-way

coupled systems. Section 6.5 presents numerical experiments to evaluate the proposed approach. Finally, Section 8.6 concludes the paper and suggests directions for future research.

6.1 Coupled systems

Differential equations form the foundation for expressing physical laws within a mathematical framework. They are essential for modeling a wide range of physical phenomena in a continuous setting and can be solved using numerical techniques. In many engineering applications, it is necessary to model the interactions of multiple physical phenomena, which requires solving several interdependent equations. Typical examples include fluid–structure interaction (Soulaïmani *et al.*, 2005), heat and mass transfer (Delgado and Sun, 2001), electromagnetic–thermal coupling (Ma *et al.*, 2022), and multisolute transport in unsaturated soils (Šimůnek *et al.*, 2006), all of which involve interdependent components governed by different physical processes.

Solving a system of coupled nonlinear PDEs is considerably more challenging than solving a single equation. The complexity increases further when the system involves complex geometries, high-dimensional domains, strong nonlinearities, steep front propagation, or multiscale interactions. Therefore, it is essential to develop efficient numerical methods, whether through traditional approaches such as FEM or modern deep learning techniques such as PINNs, to address the challenges of simulating multiphysics problems.

A system of equations can be either strongly or one-way coupled. In strongly coupled systems, equations are interdependent and need to be solved simultaneously. A typical example is fluid–structure interaction, where the fluid and structure mutually influence each other, as in the case of nonlinear aeroelasticity. In contrast, one-way coupled systems exhibit directional dependencies, allowing sequential solution once the required inputs are known. Examples include solute transport in unsaturated soils (Šimůnek *et al.*, 2006), and degradation chains where one chemical species transforms into another, such as pesticide breakdown or nitrogen transformation in groundwater (McNab Jr and Narasimhan, 1993). Table 6.1 summarizes typical one-way coupled systems from the literature.

6.2 Motivation

One-way systems inherently exhibit directional causality, requiring the equations to be solved sequentially from the first phenomenon governed by its corresponding equation to the last. This structure is analogous to temporal causality, where earlier time steps must be resolved before advancing. In one-way coupled systems, the emphasis is on solving the equations in a fixed order, ensuring that each phenomenon is learned only after its dependencies have been fully computed.

Solving one-way coupled systems simultaneously with PINNs can be challenging and

Mathematical model	Description	References
Water flow, heat transfer, and single or multispecies solute transport in the vadose zone.	A model for water flow in unsaturated soils, coupled with solute transport and heat transfer. Water infiltration is typically described by the Richards equation, while solute transport is governed by advection-dispersion-reaction equations.	(Šimůnek <i>et al.</i> , 2006; Kamil <i>et al.</i> , 2024d,e,b; Wang <i>et al.</i> , 2023b)
Surface flow and pollutant transport.	A system describing surface water flow coupled with pollutant transport. Water flow is modeled by the Saint-Venant equations, with solute transport captured by advection-diffusion equations.	(Karjoun <i>et al.</i> , 2022; Niu <i>et al.</i> , 2023)
Electro-thermal coupling in integrated circuits.	Modelling of the interaction between electric and thermal fields in integrated circuits. The electric potential is governed by the Laplace equation, coupled with a heat equation that includes a source term based on electric field and current density.	(Ma <i>et al.</i> , 2022)
Multiple reactive species transport in groundwater systems	A system of coupled transport equations describing the fate of multiple reactive species in groundwater. Each species has distinct retardation factors and is connected through first-order kinetic reactions. The transport equations include advection, dispersion, and reaction terms, with the reaction network following a sequential chain.	(Quezada <i>et al.</i> , 2004)

Table 6.1 Some examples of one-way coupled systems from the literature.

can lead to failure, as we will show later. To address this challenge, we analyze a family of PINN solvers specifically designed to respect the inherent causality of one-way coupled systems. Some of these solvers have appeared in the PINN literature, while others, to the best of our knowledge, are newly developed. Our goal is to construct a general framework for this solver family and demonstrate that a simple modification to the loss function can yield specific one-way PINN solvers. We empirically show that enforcing the correct sequence of interactions during training improves both stability and accuracy while preserving the directional structure of the one-way coupling.

In the following, we explain why training a standard PINN solver on a one-way coupled system may be ineffective. Here, “standard” refers to PINN models in which all equations are learned simultaneously, without explicitly accounting for the directionality of the coupling. We consider the coupled model of water infiltration and solute transport in soils given by equation (6.26). This system exhibits *unidirectional* coupling: the first PDE influences the second, but not vice versa. Specifically, the solute transport equation depends on inputs such as the water content θ and water flux q , which are outputs of the first PDE, while the infiltration process is independent of solute concentration. We use a standard PINN model to solve the infiltration-transport system. Details on applying PINNs to multiple equations are provided in Section 6.4. We solve this test case using two fully connected neural networks, each with 5 layers and 50 neurons, to represent the solution of each PDE. The PINN solver is trained with the Adam optimizer. The training data includes 10,000 residual points, 1,000 boundary points, and 100 initial condition points.

6.2.1 One-way causality violation

Figure 6.1(a) presents the predicted solution profiles obtained using the standard PINN solver at various training iterations, compared against a finite element reference solution for validation. Analysis of Figure 6.1(a) shows that the solution of the first PDE exhibits higher complexity, characterized by a steep gradient near the water front, whereas the solution of the second PDE is smoother and therefore easier to learn. The standard PINN solver tends to preferentially learn the simpler second PDE solution, thereby violating the system’s inherent one-way causality. This behavior is consistent with the spectral bias phenomenon in deep learning, where neural networks prioritize learning simpler patterns over more complex ones (Rahaman *et al.*, 2019). One might argue that this issue could be mitigated using Fourier feature embeddings. While such embeddings enhance the network’s ability to capture high-frequency components, they do not resolve the system’s inherent one-way causality.

To address this issue, we implement a sequential PINN strategy (details provided in later) that explicitly respects the one-way causal structure of the system. As illustrated in Figure 6.1(b), this approach preserves causality by constraining the training process to first learn the solution of the initial PDE. Only after achieving sufficient accuracy for the first equation does the model proceed to train the second PDE. This sequential enforcement ensures that the directional dependency of the system is properly maintained during optimization.

It is important to note that this example involves a relatively small system consisting of only two PDEs. In (Kamil *et al.*, 2024d), we showed that for larger one-way coupled systems, particularly those involving strong nonlinearities in some equations, the standard PINN solver fails to converge to an accurate solution of the system. These findings suggest the need to develop PINN solvers that inherently respect the one-way causality in their training approach, allowing for more accurate and reliable solutions for one-way coupled systems.

6.2.2 Neural tangent kernel analysis

In the following, we investigate the reasons behind the PINN’s violation of one-way causality in the infiltration-transport system. Specifically, we examine the gradient distribution and the neural tangent kernel (NTK) (Jacot *et al.*, 2018; Wang *et al.*, 2022d) associated with the loss terms of both PDEs. The gradient distribution of a loss term offers insights into potential imbalances during backpropagation, whereas the NTK characterizes the behavior of network parameters throughout the training process (Jacot *et al.*, 2018). The role of NTK in the context of PINNs has been analyzed in (Wang *et al.*, 2022d). For a detailed mathematical formulation of NTK, the reader is referred to Appendix 6.6.

In our example, we consider the NTK for the loss of each PDE, denoted as $\mathcal{K}_i$, where $i = 1, 2$. The eigenvalues of the NTK for each PDE are plotted in the left panel of

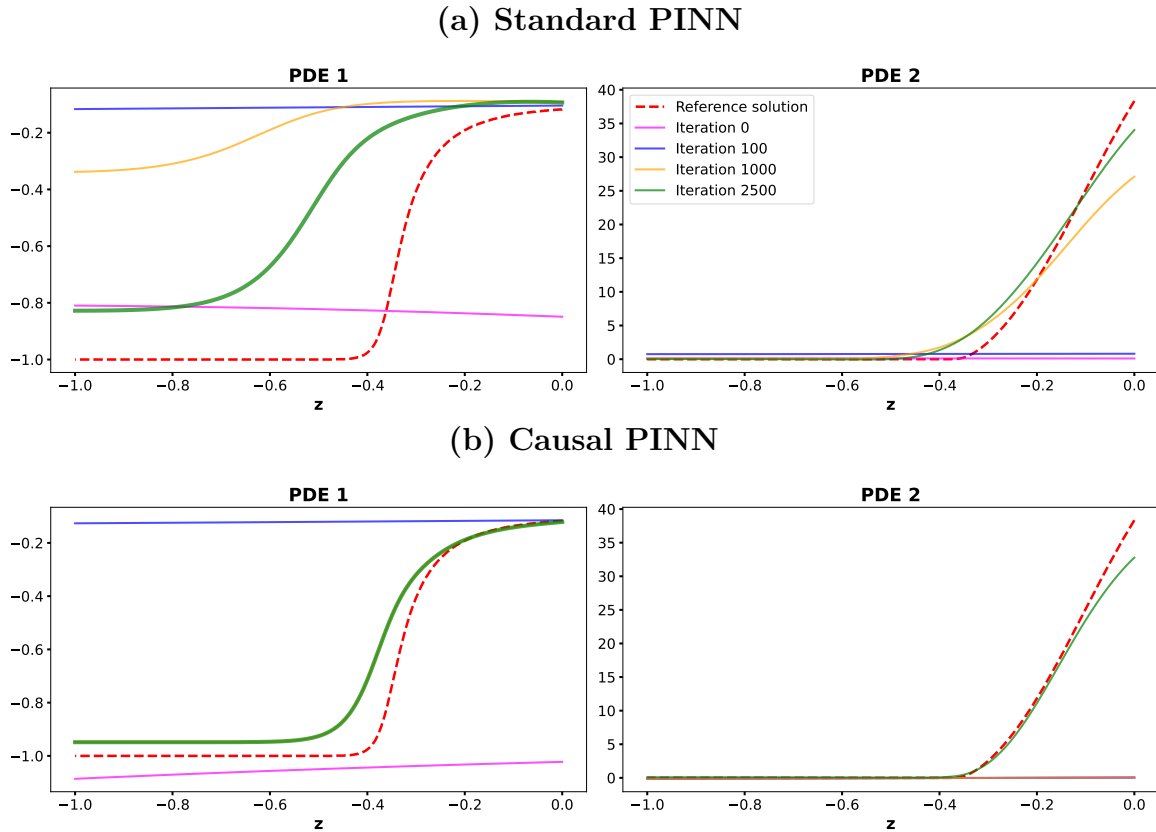


Figure 6.1 Illustration of one-way causality violation. Comparison between a standard PINN (a) and a one-way causal PINN (b) in predicting coupled water flow and solute transport, using finite element solutions as reference. Each subplot shows the solution profiles of PDE 1 (left) and PDE 2 (right) across Adam iterations. The standard PINN (a) fails to respect the causal structure of the system, leading to incorrect learning due to spectral bias. In contrast, the causal PINN (b) focuses training on PDE 1 first, preserving the one-way dependency.

Figure 6.2. A rapid decay is observed for both PDEs, which can be attributed to the spectral bias inherent in the standard PINN solver. Additionally, we compute the distribution of the back-propagated gradients for each PDE total loss term, with the results shown in the middle panel of Figure 6.2. An imbalance in the backpropagated gradients is evident between the two loss terms: the gradients of the loss for the second PDE dominate those of the first PDE. This imbalance explains why the PINN is biased toward the second PDE, leading to faster convergence for its parameters and prioritizing its solution over the first PDE. This phenomenon is referred to as *gradient masking*. Moreover, the gradient distribution shows that the parameter gradients for the first PDE are nearly zero during training, indicating little to no learning. In contrast, the second network exhibits significantly larger gradients, enabling more effective training. Moreover, we define the order of convergence of each PDE based on the eigenvalues

of its NTK (see Appendix 6.6). The results, shown in the right panel of Figure 6.2, indicate a higher order of convergence for PDE 2 compared to PDE 1. This dominance accelerates the convergence of PDE 2 and ultimately leads to a violation of the one-way causality inherent in the infiltration–transport system.

To identify when simultaneous training becomes inefficient, we introduce a practical instability indicator based on gradient magnitudes. For two PDEs i and j with total losses $\mathcal{L}_i$ and $\mathcal{L}_j$, we define the gradient masking ratio

$$G_m^{ij} := \frac{|\nabla \mathcal{L}_i|}{|\nabla \mathcal{L}_j|}. \quad (6.1)$$

If $G_m^{ij} \gg 1$, PDE i dominates the optimization. Empirically, when G_m^{ij} exceeds two orders of magnitude at initialization and persists early in training, simultaneous learning biases toward the dominant PDE. In a one-way system ($i < j$), $G_m^{ij} > 1$ directly violates the inherent causal order. In such case, switching to a one-way causal PINN mitigates gradient masking and restores balanced learning across the coupled system.

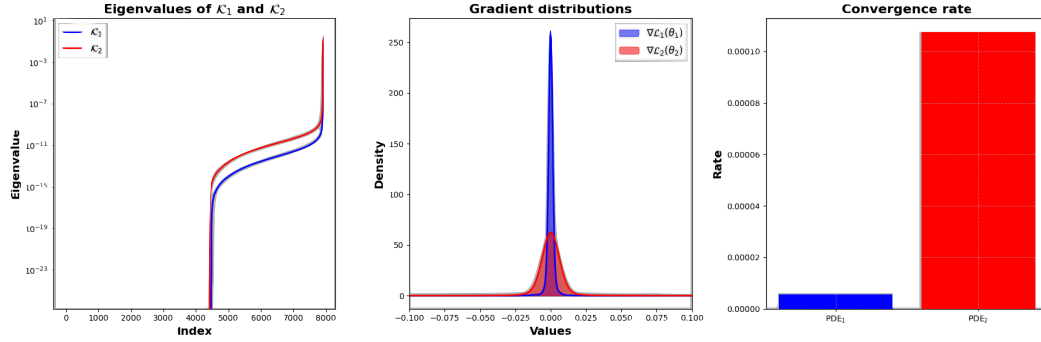


Figure 6.2 Neural tangent kernel analysis. This figure illustrates the issues of spectral bias, unbalanced back-propagated gradients, and differing convergence rates across PDEs. (Left) Eigenvalue spectra of the NTK for each PDE. (Middle) Distribution of back-propagated gradients associated with each PDE loss term. (Right) Convergence rates for each PDE. For further explanation of these concepts, see Appendix 6.6.

6.2.3 Loss landscape analysis

To better understand the challenges of training PINNs on one-way coupled systems, we examine the local geometric properties of the loss landscape. Specifically, we compare two scenarios: one where the PINN is trained on the coupled water–solute transport model, and another where it is trained solely on the water infiltration PDE.

Given the high dimensionality of neural network loss landscapes, direct visualization is not feasible. Therefore, we adopt a two-dimensional projection technique (Im *et al.*, 2016), with the detailed methodology provided in Appendix 6.6. The corresponding results are illustrated in Figure 6.3.

The Figure shows that the local 2D projection of the loss surface for the PINN trained on the coupled system converges to a sharp local minimum, with steep variations in some directions that make optimization unstable and lead to poorer generalization. In contrast, the single-PDE case exhibits a much smoother landscape, allowing for more stable convergence. As expected, the coupled model yields higher loss values due to the additional contribution from the second PDE. Although 2D projections provide a limited view of the full loss surface, they illustrate the increased optimization difficulty of the coupled system and support decomposing the optimization into separate losses.

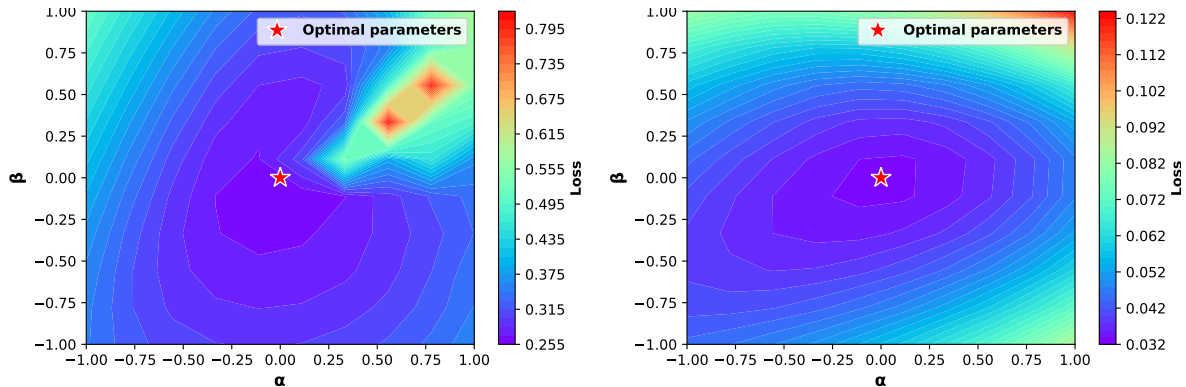


Figure 6.3 Loss landscape analysis. This figure visualizes the local geometry of the loss landscape when training a PINN on a coupled system of two PDEs versus a single PDE, using a two-dimensional projection method around a local minimum (Im *et al.*, 2016). **Left:** Projected loss landscape for a PINN trained on the coupled water–solute transport problem in unsaturated soils. **Right:** Projected loss landscape for a PINN trained only on the water infiltration PDE.

Training a PINN model is already challenging for a single equation. As shown earlier, these difficulties become even more pronounced in coupled systems. Below, we highlight two additional key challenges.

6.2.4 Ill-conditioning problem

Recent studies have shown that the loss function of a PINN is often inherently ill-conditioned, prompting more research attention on optimization strategies rather than architectural innovations (Rathore *et al.*, 2024; Cao and Zhang, 2025). Ill-conditioning of PINNs is typically assessed through the condition number of the Hessian of the loss function. When this Hessian matrix is ill-conditioned, the PINN is said to be ill-conditioned. The Hessian provides a local estimate of the curvature of the loss landscape around a given set of network parameters. A large condition number indicates that the curvature varies significantly across different directions, being flat in some directions and steep in others, which makes it challenging for standard optimizers to converge. The analysis in (Cao and Zhang, 2025; Rathore *et al.*, 2024) demonstrates

that ill-conditioning in single-physics PINNs primarily arises from the properties of the underlying differential operators. Our analysis reveals that this severe spectral ill-conditioning is intrinsic to the combined operators and is not significantly affected by the training strategy itself. Specifically, for the four-PDE system, the condition number κ remains consistently in the same order for both simultaneous and sequential configurations. We do not observe a substantial increase in κ simply as additional PDEs are added.

The performance disparity between simultaneous and sequential training lies not in the magnitude of the global condition number, but in the distribution of gradient contributions. In simultaneous training, the total loss and the resulting update step are:

$$\mathcal{L}_{\text{total}} = \sum_{i=1}^n \mathcal{L}_i, \quad \Delta\theta = -\eta \sum_{i=1}^n \nabla_{\theta} \mathcal{L}_i \quad (6.2)$$

In coupled systems, simultaneous training involves multiple operators with disparate sensitivities and scales. This creates a highly anisotropic loss landscape where gradient imbalance leads to gradient masking: stiff operators align with high-curvature spectral directions and overshadow foundational operators that align with low-curvature, flat directions. Numerical results at initialization confirm a profound imbalance:

$$|\nabla_{\theta} \mathcal{L}_{\text{PDE } 1}| \approx 2.43 \times 10^{-5}, \quad |\nabla_{\theta} \mathcal{L}_{\text{PDE } 4}| \approx 4.96 \times 10^{-3} \quad (6.3)$$

This $200\times$ disparity causes the optimizer to follow the dominant signal, neglecting the weaker but physically foundational components. Sequential training mitigates this effective ill-conditioning by isolating the physical manifolds. By solving and freezing each PDE in sequence, we remove competing sensitivities and prevent gradient masking, thereby avoiding the deterioration of the optimization landscape.

6.2.5 Gradient conflict problem

PINNs can be seen as a special case of multi-task learning, where several competing objectives such as the residual loss, boundary loss, and initial condition loss are optimized at the same time. Recent studies (see e.g., (Wang *et al.*, 2025)) have shown that the gradient of the total loss is not necessarily a descent direction for each individual loss component. In other words, for a given loss term $\mathcal{L}_i$, the inner product between its gradient $\nabla \mathcal{L}_i$ and the gradient of the total loss $\nabla \mathcal{L}$ may become negative, i.e., $\nabla \mathcal{L}_i \cdot \nabla \mathcal{L} < 0$. Empirical evidence indicates that this descent alignment condition is frequently violated during training, resulting in conflicts between competing objectives. This issue is known as the gradient conflict problem and creates a dilemma where improving one objective may deteriorate another. This challenge has mainly been studied in the context of single-physics problems.

In coupled systems, the number of loss terms increases with the number of equations, which amplifies the risk of gradient conflict and makes optimization more challenging.

In the case of one-way coupled systems, a sequential training strategy that respects the causal order can reduce such conflicts by isolating the training of each equation. This leads to more aligned gradient directions and smoother convergence. However, gradient conflict may still occur during the training of each individual equation, particularly if a conflict-aware training strategy is not used or if a second-order optimizer is not employed to ensure a descent direction for all loss terms, as discussed in (Wang *et al.*, 2025).

6.3 Related works

Traditional numerical solvers for coupled multiphysics problems generally follow one of two main strategies (Soulaïmani *et al.*, 2005). The first is the *monolithic* approach, in which all coupled equations are solved simultaneously as a single, large system. This fully accounts for interactions between different physical phenomena at each iteration, offering high accuracy and stability. However, monolithic schemes often result in large, potentially stiff linear or nonlinear systems, requiring robust solvers and significant computational resources, especially for high-dimensional or strongly nonlinear problems.

The second strategy is the *sequential* or *splitting* approach (also called *segregated* methods) (Kim *et al.*, 2011; Soulaïmani *et al.*, 2005), where the coupled system is decomposed into individual subsystems solved iteratively in a predefined order. Each subsystem uses the recent updates of the other subsystems as input, effectively treating the coupling in a one-way fashion during each iteration. This reduces the size of the systems solved at a time, lowers computational cost, and allows leveraging specialized solvers tailored to each subsystem. Methods such as block Gauss–Seidel or successive substitution are commonly used to implement this iterative decoupling (Soulaïmani *et al.*, 2005).

In this work, we extend the concept of one-way sequential coupling to the PINN framework. Rather than training a single network to learn all coupled equations simultaneously, we enforce sequential training in which each neural network corresponding to a specific PDE is trained only after its upstream dependencies have been optimized. This preserves the inherent causal structure of one-way coupled systems, analogous to how sequential solvers handle strongly directional interactions in classical numerical methods. This perspective naturally connects to advances in SciML, where training deep learning models under physical constraints has shown strong potential. Many studies have extended the methodology proposed by Raissi *et al.* (Raissi *et al.*, 2019) to increasingly complex problems. Our approach focuses on causal PINN solvers, which have demonstrated advantages in challenging multiphysics settings (Wang *et al.*, 2024b). Causality in PINNs can take various forms. Temporal causality requires earlier time steps to be learned before later ones. In (Wang *et al.*, 2024b), soft temporal causality was introduced, where a modified loss focuses the solver on previous times before ad-

vancing. Time-marching schemes have been successful for complex phase field PDEs (Mattey and Ghosh, 2022), either by training on sequential temporal windows or adaptively adding future points when prior errors meet a threshold.

Multiphysics causality is another form, as in one-way coupled systems. To address this, in (Kamil *et al.*, 2024d, 2025), we proposed a transfer learning PINN for water flow and multispecies transport in unsaturated soils, enforcing sequential solution of the system for improved accuracy in forward and inverse problems. Other sequential strategies are reviewed in subsequent sections.

Causality can also be interpreted as an attention mechanism, guiding the solver to focus on specific regions of the computational domain. Residual adaptation strategies, such as dynamically adding collocation points in high-residual regions (Wu *et al.*, 2023) or assigning adaptive weights to points based on residual loss (Anagnostopoulos *et al.*, 2024), have improved PINN training performance. In this study, one-way causality is enforced via carefully designed weight terms in the loss function that align with the system’s directional structure. This enables stable and accurate solutions for large one-way coupled systems, where standard PINNs often fail or produce biased results. Finally, the framework can be seen as a form of continual learning adapted to SciML problems. Continual learning updates a pre-trained model with new data while avoiding catastrophic forgetting. This aligns with sequential PDE solving, where each equation is learned in order and prior solutions inform subsequent steps. Similar approaches have been explored in SciML, including incremental PINNs (Dekhovich *et al.*, 2025), multifidelity continual learning PINNs (Howard *et al.*, 2024), and multistage PINNs (Khadijeh *et al.*, 2025).

6.4 Physics-informed neural networks

PINNs have gained widespread attention in the field of SciML. However, the application of standard PINNs for solving multiphysics problems characterized by high nonlinearity and multiscaling has often yielded unsatisfactory results (Kamil *et al.*, 2024d). Most research on PINNs has focused on solving single-physics, with limited focus on modeling complex multiphysics problems. In this study, we analyse a family of PINN solvers specifically designed for one-way coupled systems and evaluate their performance on various benchmark problems.

6.4.1 PINNs for single-physics problems

We present the standard PINN approach for solving a single PDE with initial and boundary conditions. Consider the following boundary value problem (BVP):

$$\mathcal{N}(u) = 0, \quad \text{in } \Omega, \quad (6.4)$$

subject to

$$\mathcal{B}(u) = 0, \quad \text{on } \partial\Omega, \quad (6.5)$$

where $\Omega \subset \mathbb{R}^d$ is an open bounded domain with a smooth boundary $\partial\Omega \subset \mathbb{R}^{d-1}$. The operator $\mathcal{N}$ defines the governing equation, including the relation between the solution u , its derivatives, and possible source or sink terms within Ω . For time-dependent problems, Ω also includes the temporal dimension. The operator $\mathcal{B}$ represents the boundary conditions on $\partial\Omega$, and for time-dependent problems, it also incorporates initial conditions.

The PINN framework approximates the solution of the BVP (6.4)–(7.14) using a neural network $u(\mathbf{x}; \Theta)$ parameterized by $\Theta \in \mathbb{R}^p$. This surrogate model maps spatial coordinates $\mathbf{x} \in \mathbb{R}^d$ to the state variable u , where Θ represents the trainable coefficients optimized during training. PINNs incorporate governing physics from operators $\mathcal{N}$ and $\mathcal{B}$ through automatic differentiation to compute the residuals $\mathcal{N}(u(\mathbf{x}; \Theta))$ and $\mathcal{B}(u(\mathbf{x}; \Theta))$ exactly. The optimization objective is given by

$$\mathcal{L}(\Theta) = \alpha_r \mathcal{L}_r(\Theta) + \alpha_{bc} \mathcal{L}_{bc}(\Theta), \quad (6.6)$$

where $\mathcal{L}_r$ and $\mathcal{L}_{bc}$ denote the L^2 -norms of the PDE and boundary residuals over Ω and $\partial\Omega$. These integrals are approximated using discrete collocation points via Monte Carlo sampling:

$$\mathcal{L}_r(\Theta) \simeq \frac{1}{N_r} \sum_{i=1}^{N_r} |\mathcal{N}(u(\mathbf{x}_i^r; \Theta))|^2, \quad (6.7a)$$

$$\mathcal{L}_{bc}(\Theta) \simeq \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} |\mathcal{B}(u(\mathbf{x}_i^{bc}; \Theta))|^2. \quad (6.7b)$$

The PINN framework is a numerical solver trained in a self-supervised manner to address forward problems. As a mesh-free method, it effectively manages high-dimensional settings and provides a computationally efficient alternative to traditional mesh-based solvers. The approach is inherently suitable for inverse problems and uncertainty quantification. In inverse settings, a data-driven term is integrated directly into the global loss function (7.15), eliminating the need for exterior coupling or the additional optimization loops characteristic of classical numerical methods.

6.4.2 PINNs for coupled systems

Solving a coupled system of equations using standard PINNs is straightforward. The total loss function for a PINN solving N equations is defined as:

$$\mathcal{L}(\Theta) = \sum_{i=1}^N \lambda_i \mathcal{L}_i(\theta_i), \quad (6.8)$$

where $\mathcal{L}_i$ denotes the loss associated with the i -th equation, and λ_i is its corresponding weighting coefficient, which we refer to as the PDE weight. The parameter vector $\Theta = (\theta_1, \dots, \theta_N)$ includes the neural network parameters used to approximate the solution of each equation in the system.

We adopt N independent neural networks to solve the system of N equations defined in (6.8), where the solution of the i -th equation is parameterized by θ_i . We utilize decoupled architectures rather than multi-output networks to facilitate specialized approximation of distinct physical features and improve training convergence (Laubscher, 2021). This modularity allows us to employ heterogeneous architectures, where hyperparameters and network depths are tailored to the specific regularity and complexity of each system component.

The choice of the PDE weights λ_i in equation (6.8) plays a crucial role in shaping different PINN solvers. Building on this general framework, we introduce a unified approach for solving systems of equations, with a particular emphasis on one-way coupled systems.

Notations. In the following, we introduce notations to ensure a concise presentation of the mathematical concepts. Consider a one-way coupled system consisting of N equations, which we solve sequentially. Let N_i denote the number of training iterations allocated to the PINN solver for equation i . Consequently, the training of the PINN for the second equation begins at iteration $N_1 + 1$ and ends at $N_1 + N_2$. More generally, the training for equation i starts with iteration $\sum_{j=1}^{i-1} N_j + 1$ and ends with $\sum_{j=1}^i N_j$. The total number of iterations required to train the PINN solvers for the full system is therefore given by $\sum_{i=1}^N N_i$. For brevity, we define the cumulative iteration count as $\mathbb{N}_i := \sum_{j=1}^i N_j$, with $\mathbb{N}_0 = 0$. We refer to the interval $[\mathbb{N}_{i-1} + 1, \mathbb{N}_i]$ as the training interval for equation i .

Remark 5 *It is important to note that this notation is not limited to the sequential PINN approach. The standard PINN can also be interpreted as a solver requiring $\mathbb{N}_N$ iterations, even though it solves all the equations simultaneously rather than one after another.*

Remark 6 *The choice of training iterations N_i is problem-dependent, and a larger number of iterations should be allocated to the more challenging equations in the system.*

Remark 7 *In practice, alternative stopping criteria may cause training to terminate before reaching the prescribed N_i iterations. In such cases, the actual number of iterations used to train equation i is denoted by N'_i , where $N'_i \leq N_i$. This adjusted value should be used in place of N_i when computing the cumulative count $\mathbb{N}_i$.*

6.4.3 PINNs for one-way coupled systems

To provide a unified framework of PINN solvers for one-way coupled systems, we modify equation (6.8) to include the iteration step t of the gradient descent optimizer:

$$\mathcal{L}(\Theta, t) = \sum_{i=1}^N \lambda_i(\Theta, t) \mathcal{L}_i(\theta_i). \quad (6.9)$$

6.4.3.1 Standard PINN (PINN)

The standard PINN is a straightforward approach for solving systems of equations, where the solution of each equation is represented by a neural network and all equations are solved simultaneously. Consequently, the PDE weights λ_i are set to unity, and the loss function reads:

$$\mathcal{L}(\Theta, t) = \sum_{i=1}^N \mathcal{L}_i(\theta_i), \quad \text{for } t \in [1, \mathbb{N}_N]. \quad (6.10)$$

The standard PINN has been applied to solve systems of equations across various fields. For instance, it has been used to solve the Navier–Stokes equations (Raissi *et al.*, 2019) and to simulate multispecies flow and heat transfer (Laubscher, 2021), among other applications.

The standard PINN has been shown to be inefficient for one-way coupled systems (Kamil *et al.*, 2024d; Laubscher, 2021; Kamil *et al.*, 2025). This inefficiency arises from the training process ignoring the sequential nature of such systems, as discussed in Section 6.2. As the number of equations increases, the loss landscape becomes more complex, making optimization increasingly challenging and unstable, thereby reducing the effectiveness of standard PINNs. Therefore, the choice of the PDE weights should reflect the inherent causality of one-way coupled systems, ensuring that each equation is solved in the correct order. Refer to Figure 6.4 for a visualization of the proposed PINN solvers for one-way coupled systems. A detailed explanation of each solver is provided below.

6.4.3.2 Sequential PINN (S-PINN)

An alternative to the standard PINN for one-way coupled systems is the sequential training approach (S-PINN). In this approach, equations are solved independently, one after another, following the order defined by the one-way coupling. Once the PINN for an equation is trained, its parameters are frozen, and the model is used to provide its solution to subsequent equations that depend on it. This process continues until all equations in the system are solved. The PDE weights in S-PINN are defined as follows:

$$\lambda_i(\Theta, t) = \begin{cases} 1, & \text{if } t \in [\mathbb{N}_{i-1} + 1, \mathbb{N}_i], \\ 0, & \text{otherwise,} \end{cases} \quad (6.11)$$

which yields the following loss function:

$$\mathcal{L}(\Theta, t) = \mathcal{L}_i(\theta_i), \quad \text{if } t \in [\mathbb{N}_{i-1} + 1, \mathbb{N}_i]. \quad (6.12)$$

In S-PINN, $\lambda_i(\Theta, t)$ controls whether the loss of equation i contributes to the total loss at iteration t . These weights act as a mask, enabling only the active equation to affect training. The corresponding parameters θ_i are updated solely within the interval $[\mathbb{N}_{i-1} + 1, \mathbb{N}_i]$ and remain fixed otherwise. This enforces a one-at-a-time training scheme, preserving the sequential nature of one-way coupled systems.

Remark 8 *The S-PINN framework can be generalized by introducing a periodicity parameter n . In this n -cycle iteration, the system equations are solved sequentially, and upon reaching the final equation, the cycle restarts from the first equation. This process repeats for a fixed number of cycles or until updates of the network parameters fall below a convergence threshold. The standard S-PINN corresponds to $n = 1$, which suffices for one-way coupled systems. Extensions with $n \geq 2$ address strongly coupled systems by mitigating gradient conflicts between constituent loss terms (Chen et al., 2025; Haghighat et al., 2022).*

The S-PINN technique has been explored in a limited number of studies. It has been applied to model unsaturated flow phenomena, such as water infiltration and heat transfer (Wang et al., 2023b), simulate multispecies solute transport (Kamil et al., 2024d, 2025), and predict mixed electroosmotic pressure-driven flow (Merdasi et al., 2023). In (Kamil et al., 2024d, 2025), we demonstrated that S-PINN outperforms the standard PINN, especially in systems with many equations exhibiting strong nonlinearities and multiscale behavior, which can otherwise lead to convergence issues or training failure. We also showed that combining S-PINN with transfer learning, referred to as Transfer Learning PINN (TL-PINN), further improves accuracy and accelerates training. The TL-PINN approach will be discussed later.

6.4.3.3 Smooth sequential PINN (2S-PINN)

The S-PINN solver uses PDE weights that change discretely across training iterations, causing $N - 1$ abrupt transitions in the total loss function. These discontinuities can hinder optimizer stability and slow convergence. To address this, a smooth variant of the weights is introduced, leading to the smooth sequential PINN (2S-PINN). Figure 6.5 illustrates the difference between S-PINN and its smooth counterpart, 2S-PINN. The smoothing function used is defined as follows:

$$f_i^s(t) = \frac{1}{\left(\exp(\lambda_s(t - \lambda_c - \lambda_w)) + 1 \right) \left(\exp(\lambda_s(-t + \lambda_c - \lambda_w)) + 1 \right)}, \quad (6.13)$$

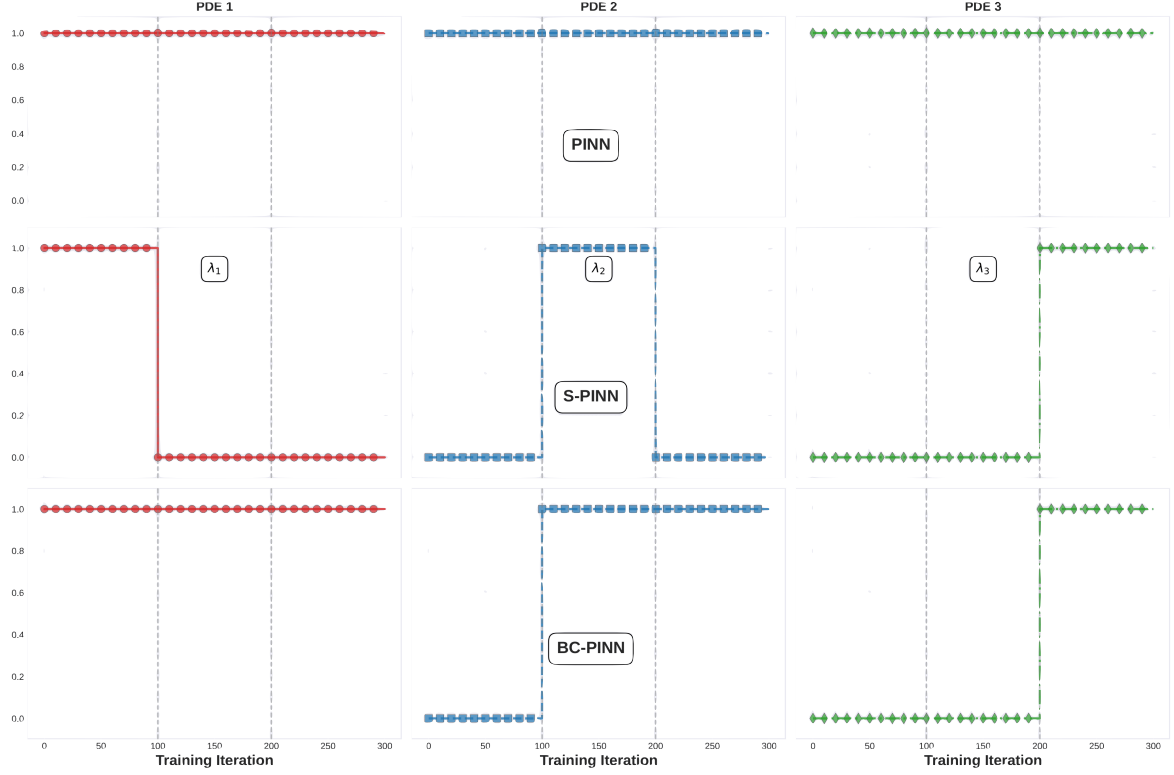


Figure 6.4 PDE weights as a function of the training step for three PINN variants applied to a one-way coupled system of three equations. Each equation is trained for 100 iterations. The standard PINN uses constant (unity) weights, while S-PINN and BC-PINN employ iteration-dependent weights. See the corresponding sections for details on each solver.

where λ_c defines the center of the plateau, λ_w determines the width of the plateau, creating a window of size $2\lambda_w$ centered at λ_c , and λ_s controls the sharpness of the transition from 0 to 1.

The dependence of $f_i^s(t)$ on equation i is captured by the parameter λ_c , which corresponds to the center of the training interval for equation i . This function is used only for equations 2 to $N-1$. For the first and last equations, a different smooth function is applied:

$$f_i(t) = \frac{1}{1 + \exp(\lambda_s(t - \mathbb{N}_i - 1))}. \quad (6.14)$$

Finally, the PDE weights in the 2S-PINN are defined as follows for $t \in [1, \mathbb{N}_N]$:

$$\lambda_i(t, \Theta) = \begin{cases} f_1(t), & \text{if } i = 1, \\ f_i^s(t), & \text{if } i = 2, \dots, N-1, \\ 1 - f_N(t), & \text{if } i = N. \end{cases} \quad (6.15)$$

The 2S-PINN loss function is expressed as follows:

$$\mathcal{L}(\Theta, t) = f_1(t)\mathcal{L}_1(\theta_1) + \sum_{i=2}^{N-1} f_i^s(t)\mathcal{L}_i(\theta_i) + (1 - f_N(t))\mathcal{L}_N(\theta_N), \quad \text{for } t \in [1, \mathbb{N}_N]. \quad (6.16)$$

6.4.3.4 Backward compatible PINN (BC-PINN) and smooth variant

Inspired by BC-PINN ([Mattey and Ghosh, 2022](#)), we propose an analogous approach for one-way coupled systems. In this method, learning of equation i begins at iteration $\mathbb{N}_{i-1} + 1$ and continues until the final iteration $\mathbb{N}_N$. The first equation remains active throughout the entire training phase. The loss for equation i is activated at iteration $\mathbb{N}_{i-1} + 1$ and continues to be updated until the end. Thus, the training interval for equation i is $[\mathbb{N}_{i-1} + 1, \mathbb{N}_N]$, and the PDE weights read:

$$\lambda_i(\Theta, t) = \begin{cases} 0, & \text{if } t \leq \mathbb{N}_{i-1}, \\ 1, & \text{otherwise.} \end{cases} \quad (6.17)$$

The BC-PINN loss function is expressed as follows:

$$\mathcal{L}(\Theta, t) = \sum_{j=1}^{N_i} \mathcal{L}_j(\theta_j), \quad \text{if } t \leq \mathbb{N}_i. \quad (6.18)$$

A smooth variant of BC-PINN, referred to as SBC-PINN, can also be employed. This results in the following loss function:

$$\mathcal{L}(\Theta, t) = \mathcal{L}_1(\theta_1) + \sum_{i=2}^N (1 - f_i(t))\mathcal{L}_i(\theta_i), \quad \text{for } t \in [1, \mathbb{N}_N]. \quad (6.19)$$

6.4.3.5 One-way causal PINN (OC-PINN)

Inspired by the time-causality PINN ([Wang *et al.*, 2024b](#)), we propose a similar strategy for one-way coupled systems, referred to as the one-way causal PINN (OC-PINN). In this approach, the PDE weight λ_i remains close to zero if the losses of the previous $i - 1$ equations have not been sufficiently minimized. This enforces a soft causal structure, activating λ_i only when preceding losses are reduced, ensuring a sequential and prioritized learning process. Formally, λ_i is defined as:

$$\lambda_i(\Theta, t) = f_{\text{oc}}^i(\theta_1, \dots, \theta_{i-1}). \quad (6.20)$$

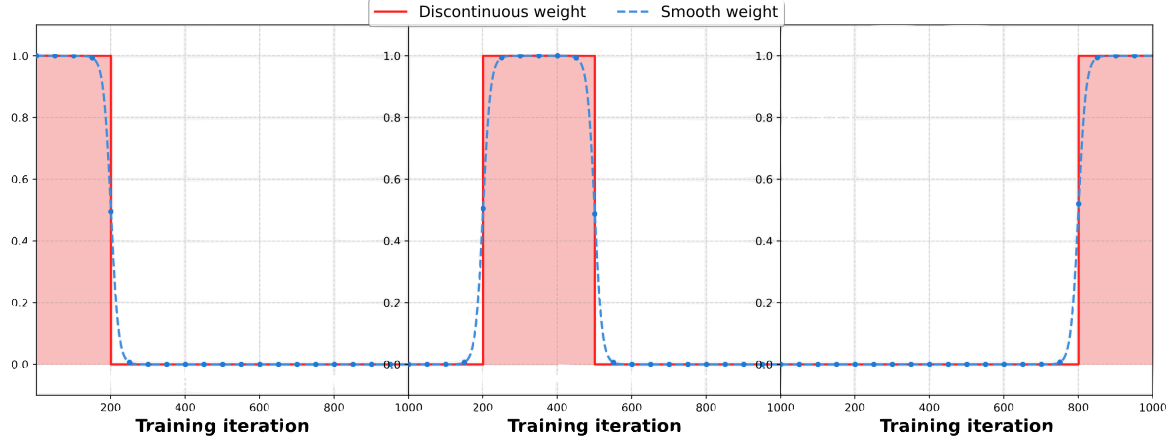


Figure 6.5 Smooth vs. Discontinuous weights: Comparison of sequential PINN PDE weights. The red dashed lines show the discontinuous weights in S-PINN, while the smooth curves represent 2S-PINN. From left to right: weights for equation 1, a middle equation (equation 2 to $N - 1$), and equation N .

where f_{oc} encodes the one-way causality in the system. Following (Wang *et al.*, 2024b), one possible choice is:

$$f_{\text{oc}}^i(\theta_1, \dots, \theta_{i-1}) = \exp \left(-\varepsilon_0 \sum_{j=1}^{i-1} \mathcal{L}_j(\theta_j) \right), \quad (6.21)$$

with ε_0 as a tunable hyperparameter controlling sensitivity to earlier losses.

Alternative functions for f_{oc}^i include $1 - \tanh(\varepsilon_0 x_i)$ or $1/(1 + \varepsilon_0 x_i)$, where $x_i = \sum_{j=1}^{i-1} \mathcal{L}_j(\theta_j)$.

In general, any smooth function can be used that satisfies: $f_{\text{oc}}(x) \rightarrow 1$ as $x \rightarrow 0$, $f_{\text{oc}}(x) \rightarrow 0$ as $x \rightarrow \infty$, and $f_{\text{oc}}(x) \in [0, 1]$.

The loss function for OC-PINN is then obtained by substituting equation (6.21) into the general PINN loss formulation:

$$\mathcal{L}(\Theta, t) = \sum_{i=1}^N \exp \left(-\varepsilon_0 \sum_{j=1}^{i-1} \mathcal{L}_j(\theta_j) \right) \mathcal{L}_i(\theta_i), \quad \text{for } t \in [1, \mathbb{N}_N]. \quad (6.22)$$

To illustrate OC-PINN, consider a one-way coupled system with two PDEs. If the loss of PDE 1 remains large, the weight λ_2 stays near zero, delaying training on PDE 2. This prioritizes solving earlier equations first, respecting the one-way causal structure of the system.

6.4.3.6 Transfer Learning PINN (TL-PINN)

Empirical studies have shown that S-PINN can be improved by initializing network parameters from a pre-trained PINN, leading to faster convergence and higher accuracy (Kamil *et al.*, 2024d). This fine-tuning strategy can be further enhanced by freezing certain layers during training, a technique known as transfer learning, resulting in the TL-PINN approach. TL-PINN is a variant of S-PINN that aligns well with the sequential structure of one-way coupled systems: after solving equation i , the learned parameters θ_i^* are used to initialize the parameters for equation θ_{i+1} . This transfer continues until the full system is solved. However, the benefit may be limited when the equations differ significantly in structure.

6.4.3.7 Summary

A comparison of the discussed PINN solvers and their key characteristics is provided in Table 6.2. Each method is characterized by whether it enforces one-way causality, the type of causality (hard or soft), the smoothness of PDE weights across optimizer iterations, the training interval for equation i , and the associated loss function. Notably, OC-PINN is the only solver that applies soft causality through PDE weights explicitly dependent on the network parameters Θ , whereas all other approaches enforce causality in a hard-coded manner.

Figure 6.6 illustrates a one-way coupled system of three equations, with each neural network parameterized by θ_i to approximate the solution of equation i . Different PINN solvers are compared by highlighting whether the network parameters remain trainable or are frozen during each training stage.

Scheme	One-way causality	Type	Weight	Training interval	Loss
PINN	No	-	Smooth	$[1, \mathbb{N}_N]$	(6.10)
S-PINN	Yes	Hard	Discontinuous	$[\mathbb{N}_{i-1} + 1, \mathbb{N}_i]$	(6.12)
TL-PINN	Yes	Hard	Discontinuous	$[\mathbb{N}_{i-1} + 1, \mathbb{N}_i]$	(6.12)
2S-PINN	Yes	Hard	Smooth	$[\mathbb{N}_{i-1} + 1, \mathbb{N}_i]$	(6.16)
BC-PINN	Yes	Hard	Discontinuous	$[\mathbb{N}_{i-1} + 1, \mathbb{N}_N]$	(6.18)
SBC-PINN	Yes	Hard	Smooth	$[\mathbb{N}_{i-1} + 1, \mathbb{N}_N]$	(6.19)
OC-PINN	Yes	Soft	Smooth	$[1, \mathbb{N}_N]$	(6.22)

Table 6.2 Summary of PINN solver, showing causality, causality type, weight type, training interval of equation i , and loss function.

6.5 Numerical examples

In this section, we evaluate the standard and the one-way PINN solvers on various one-way coupled systems. For all simulations, we report the mean relative L^2 -error along with the standard deviation, based on at least five runs of the PINN solvers to account for stochasticity. The relative L^2 -error between a prediction $\hat{u}$ and a reference

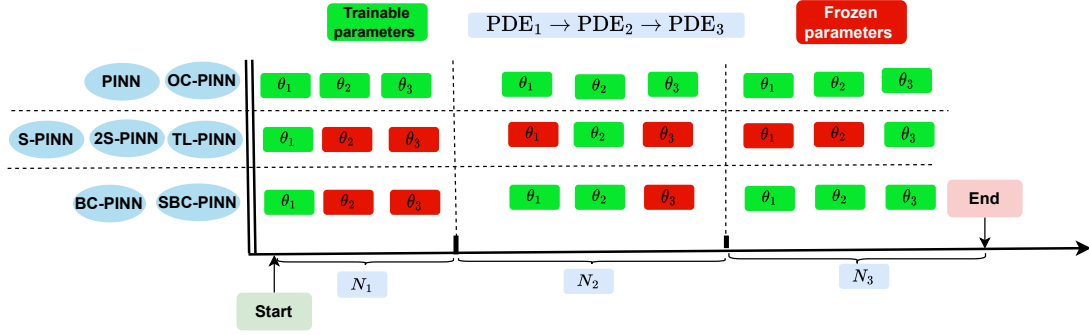


Figure 6.6 Illustration of a system of three one-way coupled PDEs, where each equation is approximated by a neural network with parameters θ_i . The figure shows, for each solver, whether the parameters are trainable or frozen during training.

solution u over a given domain Ω is defined as:

$$\text{Relative } L^2\text{-error} = \frac{\int_{\Omega} (\hat{u}(x) - u(x))^2 dx}{\int_{\Omega} u(x)^2 dx}. \quad (6.23)$$

To ensure a fair comparison across all PINN solvers, we fix the total number of iterations to N_N and employ the Adam optimizer without early stopping. This configuration ensures that every model is trained on the full system for an identical number of iterations.

6.5.1 Thermo-mechanics coupling problem

We consider a 1D steady-state thermo-mechanical system where temperature and displacement in a solid body are one-way coupled. This system is defined as follows:

$$\begin{cases} \frac{d}{dx} \left(k(x) \frac{dT}{dx} \right) = 0, \\ \frac{d}{dx} \left(E(x) \left(\frac{du}{dx} - \alpha(T - T_0) \right) \right) = 0. \end{cases} \quad (6.24)$$

The first ODE models heat transfer, a second-order diffusion equation, where T is the temperature field and k is the thermal conductivity. The second ODE models linear elastic deformation, where u is the displacement field, E is the elastic modulus, α is the thermal expansion coefficient, and T_0 is the reference temperature. Further problem specifics, including constitutive relations, boundary conditions, and exact solutions, are provided in Appendix 6.6.

In the thermo-mechanics coupling problem (6.24), the displacement u depends on the temperature T , showing the one-way coupling in the problem. The authors in (Raj

et al., 2021) used a standard PINN approach for this system. Here, we extend this approach by using the one-way PINNs to explore possible improvements.

We use two feedforward neural networks with 3 hidden layers and 10 neurons per layer: $T(x; \theta_1)$ for ODE 1 and $u(x; \theta_2)$ for ODE 2. Each network is trained for $N_1 = N_2 = 1000$ Adam iterations. The relative L^2 -error is monitored throughout training. Figure 6.7 presents the results over ten independent runs.

All PINN solvers perform well on this problem, with TL-PINN showing a slight advantage, achieving errors on the order of 10^{-5} . One-way PINNs attain similar accuracy to the standard PINN, despite requiring fewer iterations per equation. In the standard PINN, all network parameters are updated jointly over a total of $N_1 + N_2 = 2000$ iterations. In contrast, one-way solvers like S-PINN and TL-PINN update each network independently using only N_i iterations for the corresponding equation. This demonstrates the efficiency gained by exploiting the system’s one-way causal structure.

Next, we consider the inverse version of the coupled thermo-mechanics problem. The objective is to infer the thermal expansion coefficient α from limited observations of the displacement field u . The parameter α is treated as a trainable variable and is updated during training using the Adam optimizer. We compare three PINN solvers: the standard PINN, the S-PINN, and the OC-PINN. To assess robustness, we perform five independent experiments, each with different initializations of α and the neural network parameters. A mean squared error data-misfit term is added to the loss function of the second ODE using 10 displacement measurement points. The results are reported in Table 6.3.

Table 6.3 Inverse parameter estimation of α using PINN solvers in a 1D thermo-mechanical system. The statistics (mean $\pm$ std) are derived from five independent trials. Ground truth is $\alpha = 1$.

	PINN	S-PINN	OC-PINN
α (mean $\pm$ std)	0.9883 ± 0.0108	0.9764 ± 0.0162	0.9941 ± 0.0140

All PINN solvers identify the thermal expansion coefficient with high accuracy, converging to the exact value ($\alpha = 1$) with a mean absolute error on the order of 10^{-2} , where OC-PINN provides the most precise results. Numerical experiments reveal that convergence rates vary significantly: on average, the standard PINN requires 2,327 iterations, while S-PINN requires 4,577. In contrast, OC-PINN demonstrates superior efficiency, reaching the target within 1,800 iterations. The slower convergence of S-PINN is due to the fact that during the first training stage, the solver minimizes only the heat transfer residual, where no signal from α appears. Because α is exclusive to the mechanical residual, its identification is effectively delayed until the second stage of training. Consequently, only a fraction of the total iterations contributes to the inverse estimation. In contrast, OC-PINN achieves faster and more consistent convergence to

the ground truth, regardless of the initialization of α .

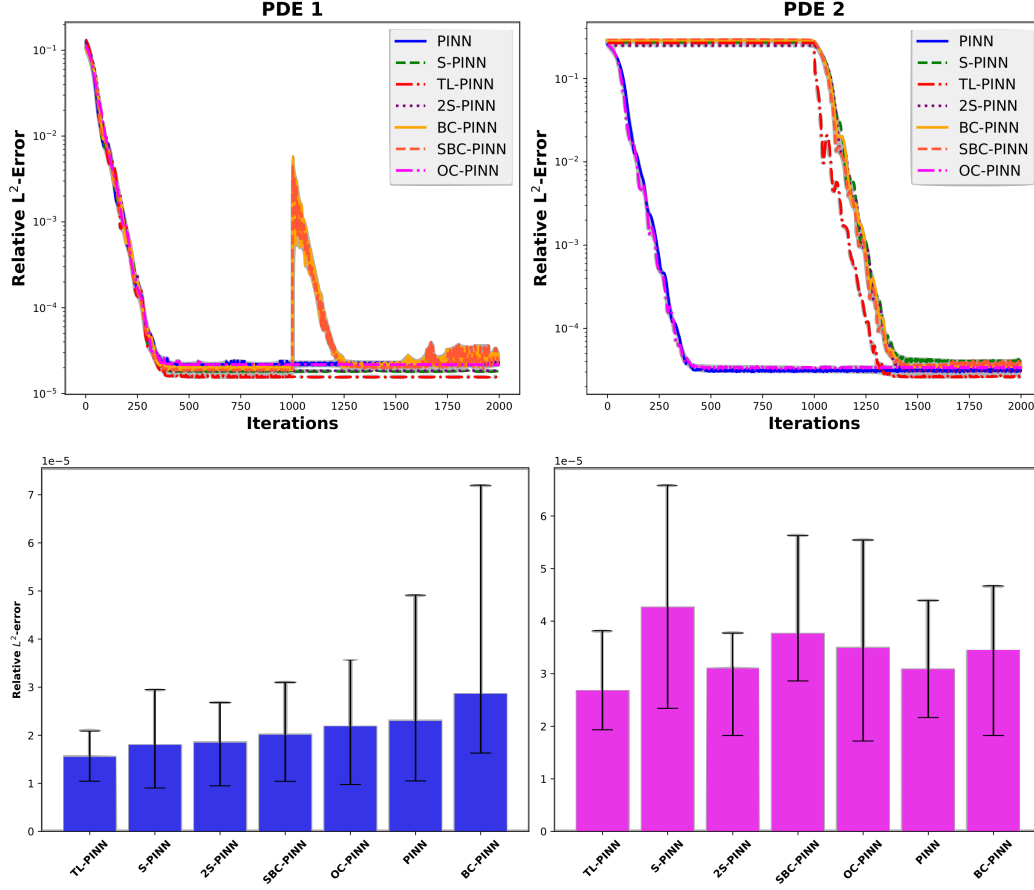


Figure 6.7 Thermo-mechanics coupling problem. Evolution of relative L^2 -errors for each PINN solver, bottom: Final relative L^2 -error values. The left panels display results for ODE 1, while the right panels show results for ODE 2. For each PINN solver, we report the minimum, maximum, and mean values across ten independent runs.

6.5.2 Electro-thermal coupling problem

We consider a one-way coupled system governing a 2D steady-state electro-thermal coupling problem, where the electrokinetic field is coupled to the thermal field. The full system is described by the following coupled 2D elliptic PDEs:

$$\begin{cases} \nabla \cdot (\sigma \nabla u) = 0, \\ -\nabla \cdot (k \nabla T) = Q(u), \end{cases} \quad (6.25)$$

where u is the electric potential, σ denotes the electric conductivity, T represents the temperature, k is the thermal conductivity, and Q is the heat source. Additional problem settings, including the boundary conditions, are detailed in Appendix 6.6.

The system (6.25) is one-way coupled: the electric potential u is needed to derive the heat source Q as specified in Appendix 6.6. In (Ma *et al.*, 2022), the authors propose an S-PINN solver for the system (6.25). Here, we further analyze the other one-way PINNs alongside the standard PINN to explore potential improvements.

We use two neural networks with 5 hidden layers and 50 neurons per layer: $u(x, y; \theta_1)$ for PDE 1 and $T(x, y; \theta_2)$ for PDE 2. Training involves $N_1 = 5000$ and $N_2 = 10000$ Adam iterations, respectively. Relative L^2 -errors are computed against reference solutions, and Figure 6.8 presents results from five independent runs.

All PINN variants demonstrate good performance on this coupled system, with S-PINN exhibiting a slight advantage on the first equation and TL-PINN achieving the highest accuracy on the second. Relative L^2 -error curves confirm that these one-way solvers consistently outperform the standard PINN. Notably, S-PINN and TL-PINN achieve superior results by training each equation for only N_i iterations, whereas the standard PINN requires $N_1 + N_2$ iterations to reach a comparable state. As in the previous numerical example, these models accelerate convergence by effectively exploiting the system's inherent one-way causality.

6.5.3 Infiltration-transport in unsaturated soils

We now consider the coupled problem of water infiltration and contaminant transport in a soil column. Water infiltration is governed by the Richards equation, while contaminant transport is modeled using a linear advection-dispersion equation. The coupled system is formulated as:

$$\begin{cases} \frac{\partial \theta}{\partial t} - \nabla \cdot [K \nabla (\Psi + z)] = 0, \\ \frac{\partial(\theta c)}{\partial t} - \nabla \cdot (\theta \mathbf{D} \nabla c - c \mathbf{q}) = 0, \end{cases} \quad (6.26)$$

where Ψ is the pressure head, z represents the spatial coordinate, $\mathbf{q} = -K \nabla (\Psi + z)$ is the water flux, K denotes the unsaturated hydraulic conductivity, and θ is the volumetric water content. The variable c represents the solute concentration in the liquid phase, and $\mathbf{D}$ is the dispersion tensor. Further details about the model setup, including the initial and boundary conditions as well as the constitutive relationships governing the coupled system, are provided in Appendix 6.6.

The system (6.26) is one-way coupled, meaning that the water infiltration process influences solute transport, but the solute concentration does not affect the water flow. A detailed description of this model, including the application of PINNs and finite element methods, can be found in (Kamil *et al.*, 2024e,d, 2025).

We employ two neural networks with 5 hidden layers and 50 neurons per layer: $\Psi(t, z; \theta_1)$ for PDE 1 and $c(t, z; \theta_2)$ for PDE 2. Figure 6.9 presents the evolution of the loss for each solver, together with the weighting parameters λ_1 and λ_2 . All PINNs solvers converge to a total loss on the order of 10^{-5} , with some one-way PINNs achieving a

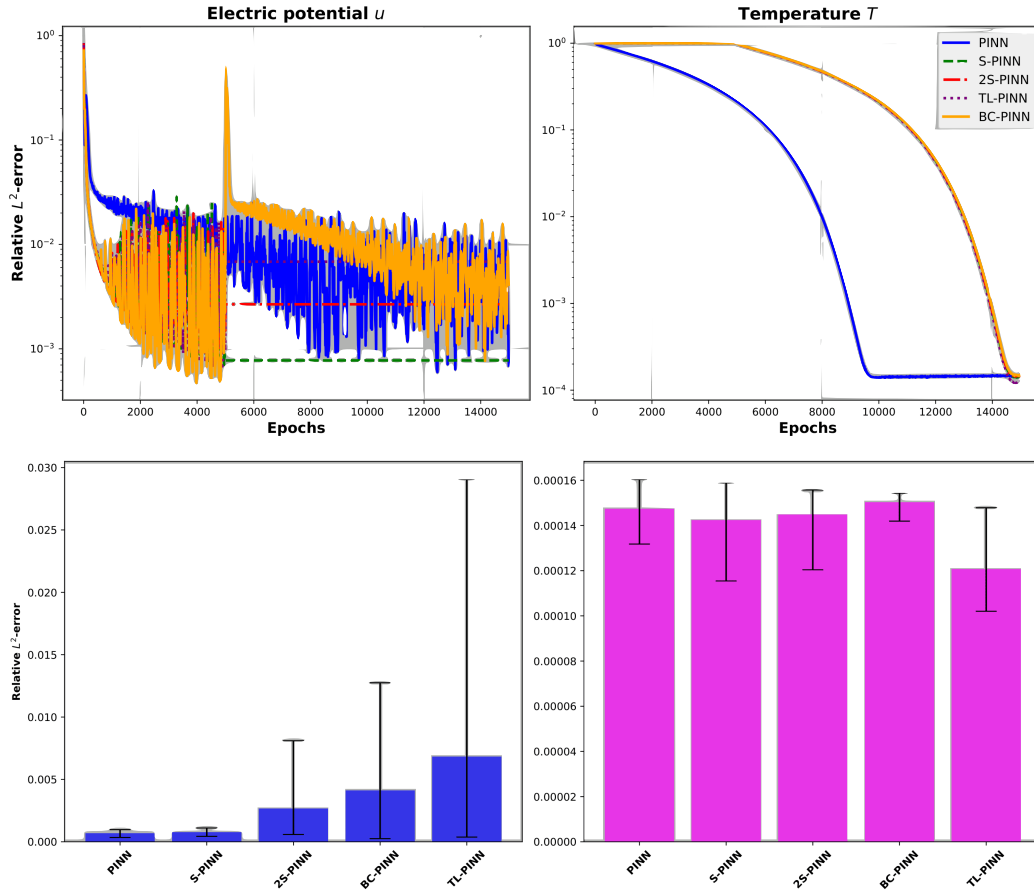


Figure 6.8 Electro-thermal coupling problem. Top: Evolution of relative L^2 -errors for each PINN solver, bottom: Final relative L^2 -error values. The left panels display results for PDE 1, while the right panels show results for PDE 2. For each PINN solver, we report the minimum, maximum, and mean values across five independent runs.

loss as low as 10^{-6} for PDE 1. In contrast, neither the standard PINN nor OC-PINN reach this level of accuracy.

For S-PINN, the total loss curve (in black) shows a discontinuity in iteration N_1 , marking the shift to the PDE 2 training. In contrast, 2S-PINN yields a smoother transition with reduced oscillations in the PDE 2 loss. A similar improvement is seen with SBC-PINN over BC-PINN, with SBC-PINN showing even smaller oscillations. These results highlight the advantage of smooth over discontinuous PDE weight adjustments in enhancing training stability and solver performance.

A notable observation in SBC-PINN is the increase in the loss of PDE 1 once the PDE 2 objective is introduced. This supports the hypothesis that simultaneous training of coupled PDEs poses a significant challenge for PINNs, as the optimization of one equation can impede the progress of the other and potentially lead to catastrophic forgetting. Specifically, terms governing PDE 2 may interfere with the learned dynamics

of PDE 1, highlighting the difficulty of balancing multiple objectives within a single framework. This interaction underscores the need for careful weighting strategies to avoid degrading performance on one equation while optimizing another.

We further analyze one-way PINNs using the relative L^2 -error as a more informative metric. Figure 6.10 presents the relative errors and final values for each solver across both PDEs. Solvers primarily focused on the first PDE, such as TL-PINN, S-PINN, and 2S-PINN, achieve higher accuracy than those training both equations simultaneously. BC-PINN performs well on the first PDE before iteration N_1 , but its accuracy drops once the second PDE is introduced, losing earlier gains. Overall, all one-way PINNs outperform the standard PINN, with smooth 2S-PINN consistently more accurate than the discontinuous S-PINN. TL-PINN matches S-PINN on the first PDE but significantly outperforms it on the second, demonstrating the benefit of transfer learning from PDE 1 to PDE 2, as shown in (Kamil *et al.*, 2024d).

We now shift our focus to a more complex numerical test, involving a system of four coupled PDEs, to investigate whether the same conclusions drawn from the previous two-PDE system hold in this more challenging scenario.

6.5.4 Infiltration and multi-species solute transport problem

We consider a system of four coupled one-way PDEs describing water infiltration governed by the Richards equation and coupled to the following multispecies solute transport (??):

$$\begin{cases} \frac{\partial \theta c_1}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c_1 - c_1 \mathbf{q}) - \mu_1 \theta c_1, \\ \frac{\partial \theta c_2}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c_2 - c_2 \mathbf{q}) + \mu_1 \theta c_1 - \mu_2 \theta c_2, \\ \frac{\partial \theta c_3}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c_3 - c_3 \mathbf{q}) + \mu_2 \theta c_2 - \mu_3 \theta c_3, \end{cases} \quad (6.27)$$

where μ_i denote the first-order decay coefficients for each species. Further details on system (6.27) are provided in Appendix 6.6.

System (6.27) is solved using the proposed PINN solvers with four neural networks, each having five hidden layers and 50 neurons per layer: $\Psi(t, z; \theta_1)$ for the first PDE, and $c_{i-1}(t, z; \theta_i)$ for the i -th PDE, where $i = 2, \dots, 4$.

For this four-equation infiltration and multispecies transport system, we observed numerically that training the fully coupled standard PINN is unstable and requires several stabilization mechanisms, in contrast to the one-way formulations. In particular, the simultaneous optimization of all PDE residuals leads to imbalance among loss components and erratic convergence behavior. To obtain stable training, we incorporated a modified multilayer perceptron (MLP) architecture (Wang *et al.*, 2021a) together with adaptive residual sampling; this combined strategy is referred to in Figure 6.11 as *Modification*. As shown in Figure 6.11, this modification reduces the relative L^2 -error of the standard PINN by approximately 31% for PDE 1 and 33% for PDE 3, while

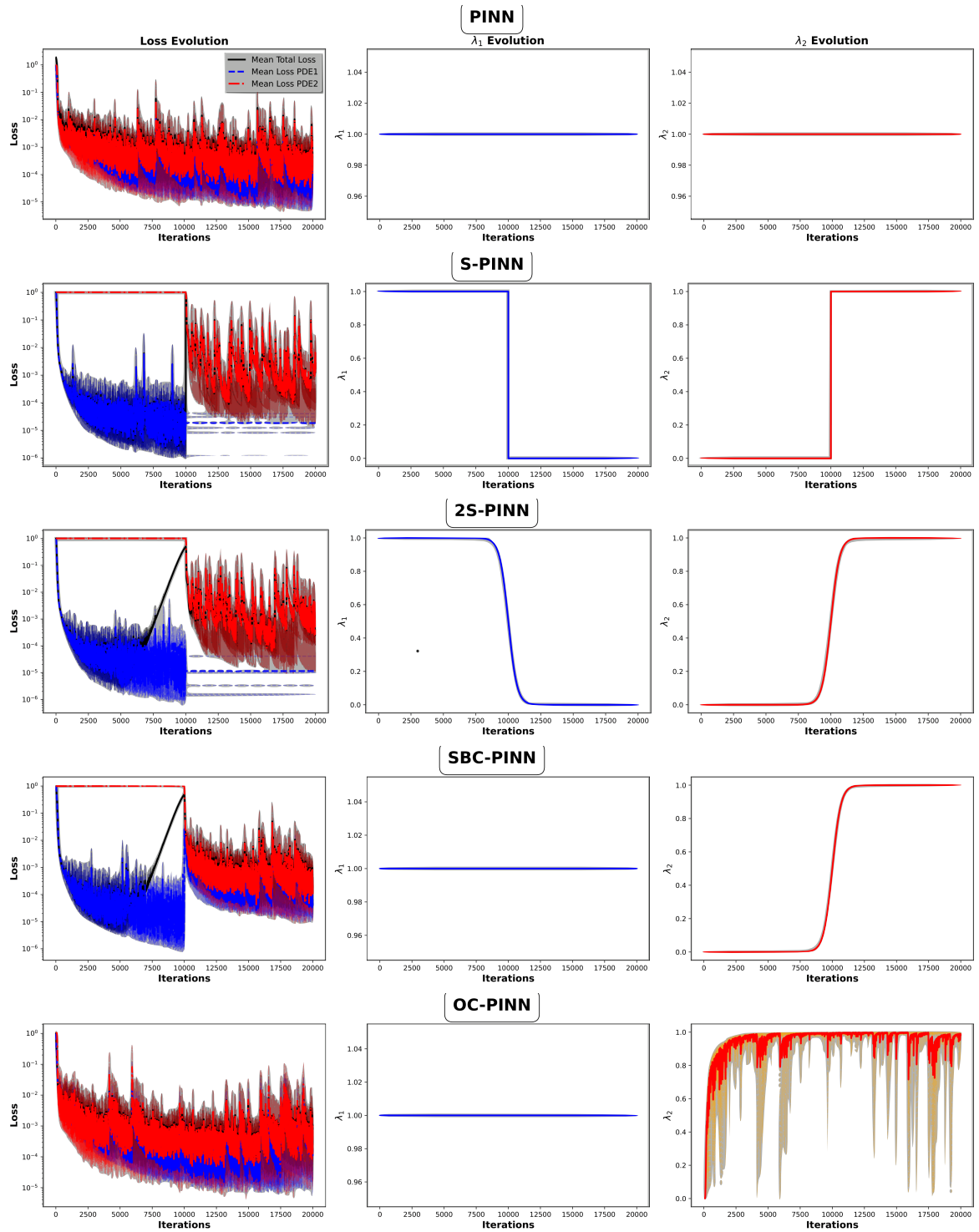


Figure 6.9 Infiltration-transport in unsaturated soils. Loss and PDE weight evolution for different PINN solvers. The results are presented by showing all five runs and the mean values for the losses of PDE 1, PDE 2, and the total loss.

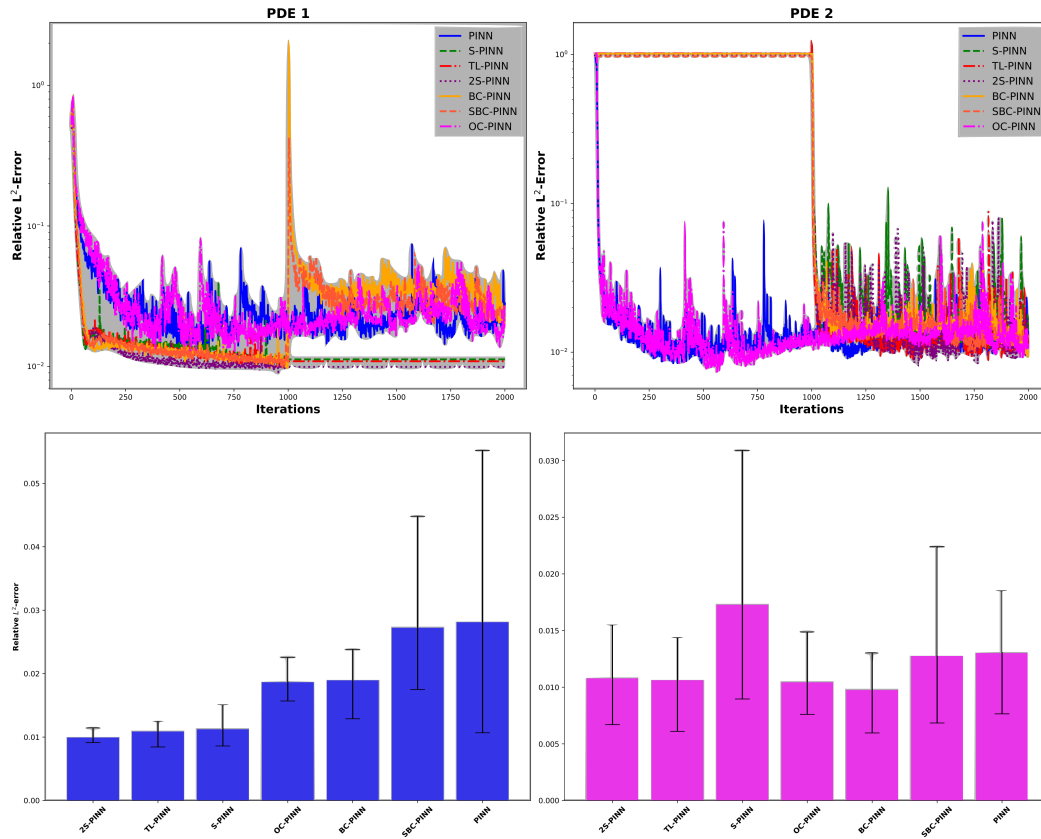


Figure 6.10 Infiltration-transport in unsaturated soils. Top: Evolution of relative L^2 -errors for each PINN solver, bottom: Final relative L^2 -error values. The left panels display results for PDE 1, while the right panels show results for PDE 2. For each PINN solver, we report the minimum, maximum, and mean values across five independent runs.

only marginal changes are observed for PDEs 2 and 4. Despite these reductions, the fully coupled formulation remains sensitive to hyperparameter selection and stabilization parameters. In contrast, the S-PINN achieves substantially lower errors without requiring architectural or sampling adjustments. Applying the same modification results in comparable error levels, indicating that the sequential formulation itself is the primary source of robustness rather than architectural refinement.

We present in Figures 6.12–6.13 the evolution of the total loss and the relative L^2 -error for each PDE using the one-way solvers. All one-way PINN variants achieve stable convergence for every loss component, with residuals decreasing to the order of $\mathcal{O}(10^{-6})$. The corresponding relative L^2 -errors remain consistent across the four PDEs and across independent runs. These results confirm that enforcing the one-way causal structure within the loss formulation leads to improved training stability for multiphysics systems governed by one-way coupled PDEs.

Finally, we report the computational cost of each PINN solver in Figure 6.14. The

standard PINN requires 394.361 s. In contrast, S-PINN, TL-PINN, and 2S-PINN significantly reduce the computational time to 110.259 s, 114.157 s, and 103.103 s, respectively. OC-PINN incurs the highest computational expense (488.225 s) due to its soft-causality formulation and simultaneous optimization. BC-PINN and SBC-PINN require 256.220 s and 262.408 s, respectively. In general, strictly sequential one-way strategies (e.g., S-PINN) provide the best trade-off between accuracy and computational efficiency, whereas soft or partially coupled formulations introduce additional overhead.

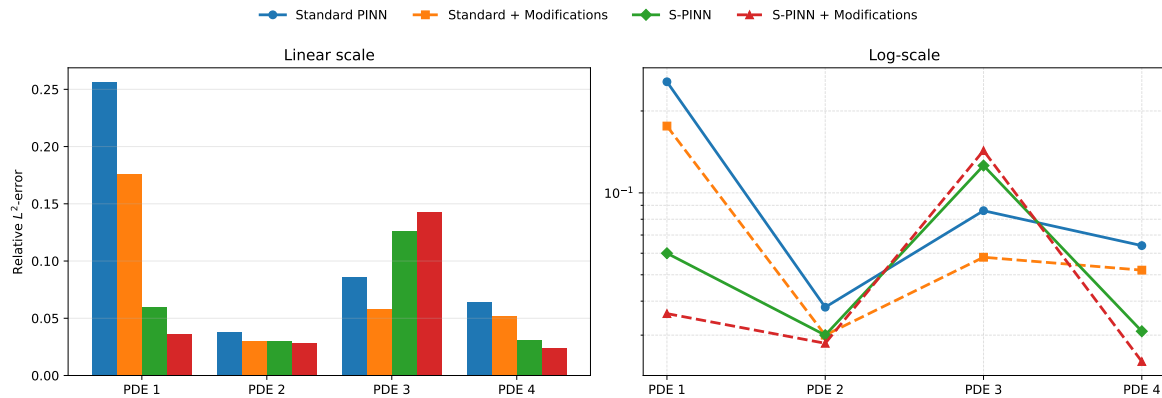


Figure 6.11 Infiltration and multispecies solute transport problem. Left: Linear-scale comparison of relative L^2 -errors for the standard PINN and S-PINN solvers, with and without architectural and sampling enhancements. Right: Log-scale representation of the same data highlighting relative performance differences across PDEs 1–4.

6.6 Conclusion

In this study, we analyzed a family of PINN solvers designed for one-way coupled systems. Unlike standard PINNs that optimize all governing equations simultaneously within a single loss function, the proposed solvers explicitly incorporate the directional coupling structure by enforcing one-way causality during training. Numerical experiments were carried out on a variety of one-way coupled problems, including thermo-mechanical systems, electro-thermal interactions, and infiltration–transport processes in unsaturated soils.

Results show that for simple stationary problems with two equations, standard and one-way PINNs perform similarly, with a slight improvement in efficiency for one-way PINNs. For more complex systems with four PDEs and strong nonlinearities, standard PINNs become inefficient, while one-way PINNs achieve better performance with comparable accuracy.

For a new one-way coupled problem, S-PINN is recommended as a robust default solver due to its strict enforcement of causality and consistent mitigation of gradient

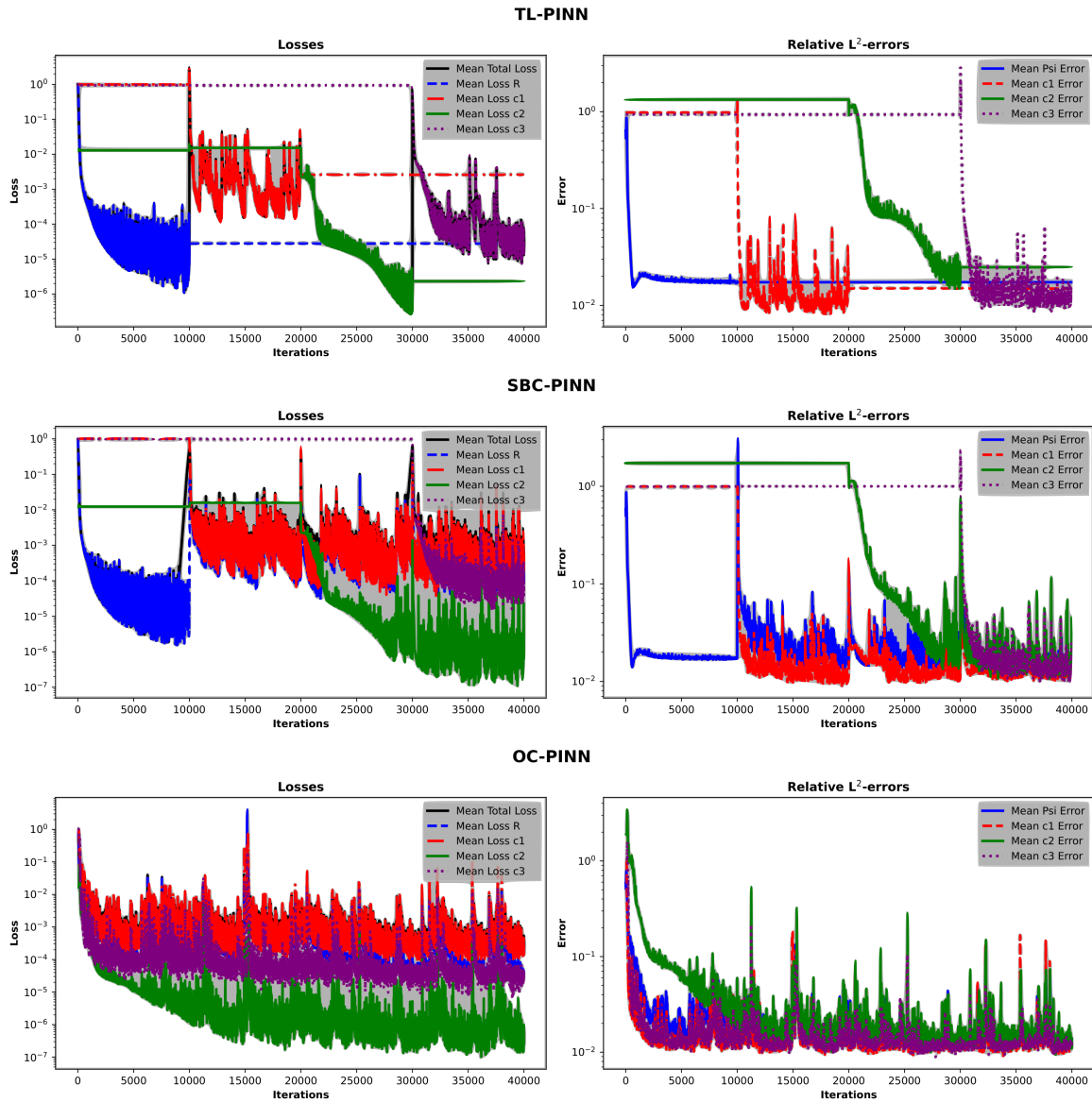


Figure 6.12 Infiltration and multispecies solute transport problem. Evolution of the total loss and relative L^2 -error over five independent runs. The plot shows the mean values for PDEs 1 to 4.

masking. If the coupled equations share similar mathematical structures (e.g., multiple advection-type PDEs with comparable sensitivities), TL-PINN can improve computational efficiency. In inverse settings, where multiple solution fields or parameters must be inferred simultaneously, OC-PINN is preferred because its soft-causality formulation enables stable joint optimization. The practical selection guideline is summarized in Table 6.4.

We note that the general hyperparameters for the proposed one-way PINN solvers, including the stage-wise iteration counts N_i and the OC-PINN parameter ε_0 , are cur-

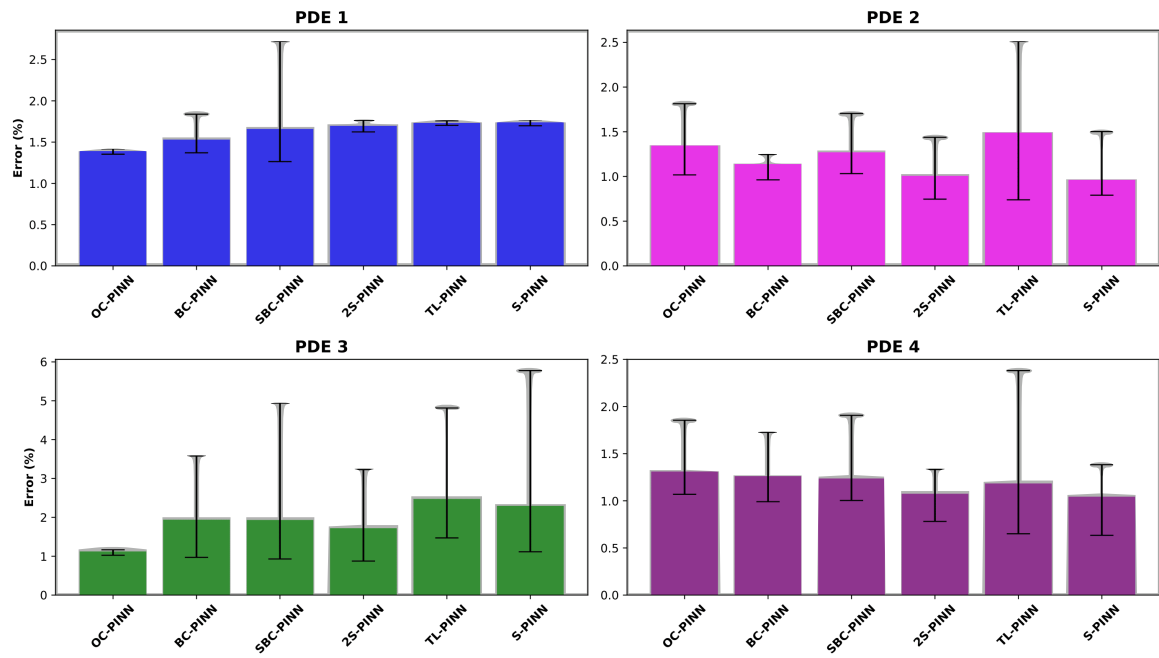


Figure 6.13 Infiltration and multi-species solute transport problem. Comparison of PINN solvers in terms of L^2 -error.

rently pre-determined based on the expected physical complexity of each sub-problem. Although these values are fixed at the start of training based on domain knowledge, adaptively tuning them during the optimization process represents a promising direction for future research. Utilizing real-time metrics such as residual decay or gradient norm tracking to dynamically adjust the training schedule could further enhance the robustness of these solvers.

Problem characteristics	Recommended solver
Forward problem with one-way causality	S-PINN (default choice)
Forward problem with similar PDE structures (e.g., advection-diffusion equations)	TL-PINN (improved efficiency)
Inverse problem (joint field/parameter inference)	OC-PINN (soft-causality formulation)
Mild coupling without significant gradient imbalance	Simultaneous PINN may be sufficient

Table 6.4 Practical guideline for selecting a PINN solver based on problem characteristics.

Although this paper focuses on one-way coupled PDEs, many multiphysics systems involve strong two-way interactions. In such settings, the coupling is no longer hierarchical, and mutual feedback can intensify gradient interference and training instability. These challenges suggest that the mechanisms identified in this study, specifically gradient conflict and loss imbalance, are equally significant in strongly coupled problems. Although the proposed one-way causality-aware strategies rely on a predefined ordering of equations, these approaches can be extended to fully bidirectional systems. One

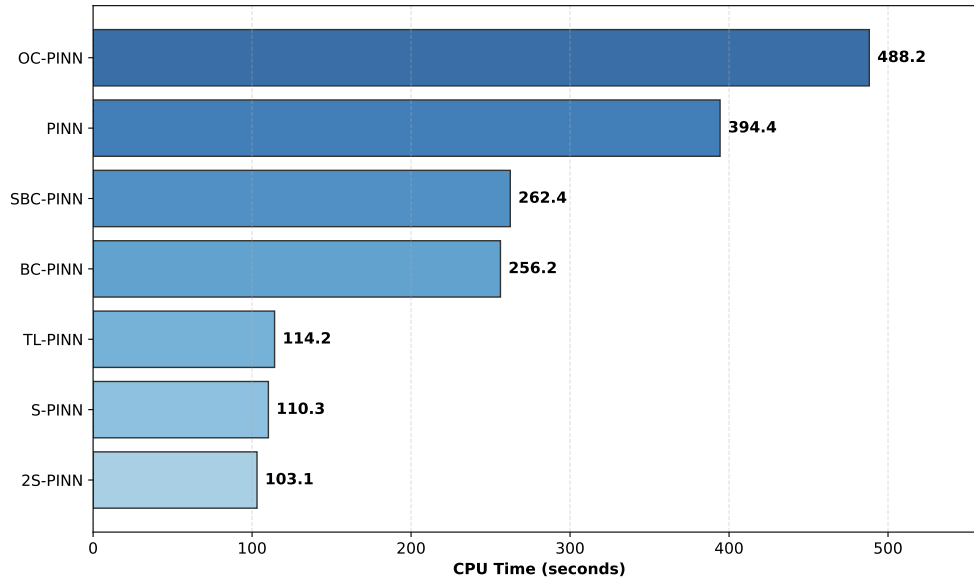


Figure 6.14 Computational efficiency comparison for the infiltration and multispecies solute transport problem. The bar chart illustrates the CPU time (in seconds) required for various PINN solvers.

possible extension to strongly coupled systems, which represents a key direction for future research, is the periodic sequential PINN strategy discussed in Remark 8. This approach adapts the Gauss-Seidel iterative algorithm to the training of PINN models for strongly coupled systems. In this framework, the constituent equations are solved sequentially in repeated cycles until global convergence is achieved.

This work highlights the challenges of training PINNs on large PDE systems and emphasizes the need for deeper theoretical understanding. Future research should aim to establish stronger theoretical foundations and develop more robust training strategies for complex multiphysics problems.

Appendix A. Configurations for the studied test cases

Thermo-mechanics coupling problem. In the thermo-mechanical coupling problem (6.24), we consider a 1D bar extending from $x = 0$ to $x = 1$. The problem is subjected to the following boundary conditions:

$$\begin{cases} u(x=0) = T(x=0) = 0, \\ u(x=1) = T(x=1) = 1. \end{cases} \quad (6.28)$$

The thermal conductivity k and elastic modulus E vary with position as follows: $E(x) = \frac{1}{1+x}$ and $k(x) = 10E(x)$. The exact solution for the coupled thermo-mechanical problem (6.24) with the boundary conditions (6.28) is:

$$T(x) = \frac{2x + x^2}{3}, \quad u(x) = \frac{x^3}{9} + \frac{14x^2}{27} + \frac{10x}{27}. \quad (6.29)$$

For the PINN algorithms, we generate 10^3 residual collocation points, sampled using the Sobol algorithm. The boundary conditions are exactly enforced by modifying the network output as follows:

$$\mathcal{N}(x; \theta) = x + x(1 - x)\tilde{\mathcal{N}}(x; \theta), \quad (6.30)$$

where $\mathcal{N}(x; \theta)$ represents the modified output of the network $\tilde{\mathcal{N}}(x; \theta)$.

Electro-thermal coupling problem. In the electro-thermal coupling problem (6.25); we consider a square domain $[0, 1]^2$, with Γ_N representing the lateral boundaries and Γ_D denoting the top and bottom interfaces. The coupled problem is subjected to the following boundary conditions:

$$\begin{cases} \nabla u \cdot \mathbf{n} = 0, & \text{on } \Gamma_N, \\ u = u_D, & \text{on } \Gamma_D, \\ T = T_D, & \text{on } \Gamma_D. \end{cases} \quad (6.31)$$

The heat source Q in the heat equation is defined as:

$$Q = J^T E, \quad (6.32)$$

where $J = \sigma E$, with J is the electric current density and $E = -\nabla u$ is the electric field. We use the following parameters (Ma *et al.*, 2022): $k = 1$ W/(m · K), $\sigma = 1$ S/m, $u_D = 1$ V on the top boundary and $u_D = 0$ V on the bottom boundary, and $T_D = 273$ K.

For the PINN algorithms, We generate 10^4 residual collocation points and 5×10^2 points for each lateral boundary. The points are sampled using the Latin Hypercube algorithm.

Infiltration-transport problem. The coupled water flow and solute transport problem (6.26) is subjected to the following initial and boundary conditions:

$$\begin{cases} \Psi(0, x) = -1, & c(0, x) = 0, \\ \mathbf{q} \cdot \mathbf{n} = q_0, & -(\theta \mathbf{D} \nabla c - c \mathbf{q}) \cdot \mathbf{n} = \bar{c}_0 q_0, & \text{on } z = 0, \\ \frac{\partial \Psi}{\partial z} = \frac{\partial c}{\partial z} = 0, & \text{on } z = -1. \end{cases} \quad (6.33)$$

The following constitutive relationships describe the capillary pressure using the Mualem-van Genuchten model:

$$\theta = \begin{cases} \theta_r + (\theta_s - \theta_r) (1 + (\alpha_v |\Psi|)^{n_v})^{-m_v} & \text{if } \Psi \leq 0, \\ \theta_s & \text{if } \Psi > 0, \end{cases} \quad (6.34a)$$

$$K = \begin{cases} K_s \frac{[1 - (\alpha_v |\Psi|)^{n_v-1} (1 + (\alpha_v |\Psi|)^{n_v})^{-m_v}]^2}{(1 + (\alpha_v |\Psi|)^{n_v})^{m_v/2}} & \text{if } \Psi \leq 0, \\ K_s & \text{if } \Psi > 0, \end{cases} \quad (6.34b)$$

$$m_v = 1 - \frac{1}{n_v}, \quad n_v > 1. \quad (6.34c)$$

$$\theta \mathbf{D} = D_L |\mathbf{q}| + \theta \tau D_w. \quad (6.34d)$$

The following parameters were used: $K_s = 0.2496 \text{ m/day}$, $\alpha_v = 3.6 \text{ m}^{-1}$, $\theta_s = 0.43$, $\theta_r = 0.078$, $n_v = 1.56$, $q_0 = -0.05 \text{ m/day}$, $D_w = 0 \text{ m}^2/\text{day}$, $D_L = 0.1 \text{ m}$, $\bar{c}_0 = 5$. The final simulation time is set to one day.

We generate 10^4 residual collocation points, 10^2 points for the initial condition and surface bottom boundary, and 10^3 points for the soil surface boundary. .

Infiltration and multi-species solute transport problem. In system (6.27), we consider three solute species with respective concentrations c_i that are transported with water flow in unsaturated soil. Water flow is governed by the Richards equation, similar to the infiltration-transport test. In this system, each solute species i undergoes transformation into solute $i + 1$ through a first-order decay reaction characterized by the coefficient μ_i . This forms a sequential decay chain involving three solute species. The initial and boundary conditions are the same as those in the infiltration-transport test.

Appendix B. Neural tangent kernel analysis

Here, we provide the details of the NTK used in the analysis of Section 6.2. We adopt the framework outlined in (Wang *et al.*, 2022d) to define the NTK in the context of PINNs for systems of multiple equations.

We consider minimizing the loss function (6.8) associated with a one-way coupled system using gradient descent with an infinitesimally small learning rate. For a given equation, we define the following components at any training iteration t :

$$\mathcal{R}(t) = \left(\mathcal{N}(u(\mathbf{x}_j; \Theta(t))) \right)_{j=1}^{N_r}, \quad (6.35a)$$

$$\mathcal{B}(t) = \left(\mathcal{B}(u(\mathbf{x}_j; \Theta(t))) \right)_{j=1}^{N_b}, \quad (6.35b)$$

where $\mathcal{R}(t)$ and $\mathcal{B}(t)$ denote the residual, and boundary loss terms. Their mapping spaces are $\mathcal{R} : \mathbb{R}_+ \rightarrow \mathbb{R}^{N_r}$ and $\mathcal{B} : \mathbb{R}_+ \rightarrow \mathbb{R}^{N_b}$.

We define the NTK for the residual and boundary loss terms, denoted by $\mathcal{K}_{\mathcal{R}}$, and $\mathcal{K}_{\mathcal{B}}$,

respectively, as follows:

$$\mathcal{K}_{\mathcal{R}}(t) = \mathbf{J}_{\mathcal{R}}(t)\mathbf{J}_{\mathcal{R}}(t)^T, \quad (6.36a)$$

$$\mathcal{K}_{\mathcal{B}}(t) = \mathbf{J}_{\mathcal{B}}(t)\mathbf{J}_{\mathcal{B}}(t)^T, \quad (6.36b)$$

where $\mathbf{J}_{\mathcal{R}}$, and $\mathbf{J}_{\mathcal{B}}$, are the Jacobians of $\mathcal{R}$ and $\mathcal{B}$, respectively, with respect to the network parameters Θ .

We define also the following NTK matrix:

$$\mathcal{K}_{\mathcal{RB}}(t) = \mathbf{J}_{\mathcal{R}}(t)\mathbf{J}_{\mathcal{B}}(t)^T, \quad (6.37)$$

The global NTK matrix $\mathcal{K}(t)$ is then expressed as:

$$\mathcal{K}(t) = \begin{bmatrix} \mathcal{K}_{\mathcal{R}}(t) & \mathcal{K}_{\mathcal{RB}}(t) \\ \mathcal{K}_{\mathcal{RB}}(t)^T & \mathcal{K}_{\mathcal{B}}(t) \end{bmatrix}. \quad (6.38)$$

The NTK matrix (6.38) is positive semi-definite, and we are particularly interested in its eigenvalues to define the following notion.

Definition 1. The convergence rate of equation i at iteration t , denoted by $C_i(t)$, is defined as (Wang *et al.*, 2022d):

$$C_i(t) = \frac{\sum_{j=1}^d \lambda_j}{d} = \frac{\text{Tr}(\mathcal{K}_i(t))}{d}, \quad (6.39)$$

where $\mathcal{K}_i(t)$ is the NTK matrix associated with equation i , as defined in (6.38), d is the dimension of the matrix, λ_j are its eigenvalues, and $\text{Tr}(\cdot)$ denotes the trace operator.

Definition 2. Equation i is said to dominate equation j at iteration t if the convergence rate of equation i is significantly higher than that of equation j , i.e., $C_i(t) \gg C_j(t)$.

In the analysis presented in Section 6.2.2, we adopt the above definitions to study a one-way coupled system consisting of two equations. The convergence rate for each equation is evaluated during the early stages of training to assess which equation dominates the other.

Appendix C. Loss landscape analysis

Here, we present the methodology used to visualize the two-dimensional projection of the PINN loss landscape discussed in Section 6.2. A projection technique is applied to explore the landscape in the vicinity of a local minimum. Further details can be found in (Im *et al.*, 2016).

Let $\mathcal{L}(\Theta)$ denote the loss function of a PINN solver, where $\Theta \in \mathbb{R}^n$ represents the trainable parameters. To obtain an informative 2D projection of this high-dimensional loss landscape, we follow a three-step process:

1. First, we obtain a set of parameters Θ^* by solving:

$$\Theta^* = \arg \min_{\Theta} \mathcal{L}(\Theta)$$

using the Adam optimizer. It is important to note that due to the non-convex nature of neural network optimization problems, Θ^* represents a convergence point that may be a local minimum or a saddle point, with no guarantee of global optimality.

2. Next, we generate two orthogonal random directions $d_1, d_2 \in \mathbb{R}^n$ from a standard Gaussian distribution and normalize them:

$$d_i = \frac{\tilde{d}_i}{\|\tilde{d}_i\|}, \quad \tilde{d}_i \sim \mathcal{N}(0, I_n), \quad i = 1, 2$$

3. Finally, we construct a 2D projection by evaluating:

$$f(\alpha, \beta) = \mathcal{L}(\Theta^* + \alpha d_1 + \beta d_2)$$

where $\alpha, \beta \in [-1, 1]$ control the perturbation magnitude along each random direction.

The resulting function $f(\alpha, \beta)$ is visualized over a uniform grid of (α, β) values using both surface and contour plots, providing insights into the local geometry of the loss landscape near the parameters Θ^* .

Appendix D. Gradient propagation in one-way PINNs

This section examines how neural network parameters are updated during training in one-way PINNs. Each equation is approximated by a separate neural network trained via gradient descent (GD):

$$\theta_i^{t+1} = \theta_i^t - \eta_t \nabla_{\theta_i^t} \mathcal{L}(\Theta), \quad (6.40)$$

where η is the learning rate, and Θ denotes all trainable parameters with θ_i associated with equation i .

For the S-PINN solver, using the loss (6.12), gradients are computed as:

$$\nabla \mathcal{L}(\Theta, t) = \nabla \mathcal{L}_i(\theta_i), \quad \text{if } t \in [\mathbb{N}_{i-1} + 1, \mathbb{N}_i], \quad (6.41)$$

meaning θ_i is updated only within its assigned interval, and remains fixed otherwise:

$$\theta_i^t = \begin{cases} \theta_i^0, & t \leq \mathbb{N}_{i-1}, \\ \text{updated by (6.40)}, & t \in [\mathbb{N}_{i-1} + 1, \mathbb{N}_i], \\ \theta_i^{\mathbb{N}_i}, & t > \mathbb{N}_i. \end{cases} \quad (6.42)$$

A similar scheme applies to BC-PINN:

$$\theta_i^t = \begin{cases} \theta_i^0, & t \leq \mathbb{N}_{i-1}, \\ \text{updated by (6.40)}, & \text{otherwise.} \end{cases} \quad (6.43)$$

For smooth variants, all parameters are updated throughout training, though some may experience negligible updates at times.

Due to the complexity of the PINN loss landscape, GD is rarely used in practice, as it often leads to training failure. More effective alternatives include the Adam optimizer and L-BFGS. However, with Adam, the properties in (6.41)–(6.43) no longer hold: momentum can cause parameter updates even outside active intervals, potentially degrading previously learned solutions. To mitigate this, one can freeze θ_i when its associated weight $\lambda_i = 0$, preserving learned information and avoiding unintended loss increases.

CHAPTER 7 Physics-informed neural networks for solving the BCJ-R equation: Modeling saturation overshoot in porous media

Abstract

We propose a physics-informed neural network (PINN) for solving the BCJ-R equation to simulate saturation overshoot in porous media. The BCJ-R equation which is endowed with an entropy function was developed by Beljadid, Cueto-Felgueroso and Juanes to model unstable infiltration in porous media as an extension of the Richards equation. To avoid the complexity associated with computing higher-order derivatives in the BCJ-R equation, we reformulate the problem as a coupled system of two second-order equations. The PINN architecture is based on a modified multilayer perceptron and incorporates several training enhancements, including random Fourier feature embedding to reduce spectral bias, weight factorization to improve convergence, and temporal causality enforcement via windowed training with time masking. To address multi-objective optimization challenges, we mitigate gradient imbalance across loss terms using adaptive weighting based on gradient normalization. Gradient conflicts between loss terms are further alleviated by employing the second-order SOAP optimizer. A data-parallel, multi-GPU implementation accelerates training, with strong and weak scaling confirming good performance. We validate the predictions of our approach against finite element simulations and experimental data. The numerical model based on the BCJ-R equation and the proposed PINN's approach accurately captures key features of the saturation overshoot in porous media. Finally, our numerical simulations validate that the proposed numerical approach preserves the essential thermodynamic properties of the BCJ-R equation where we observe an entropy-increasing process for an isolated system.

7.1 Introduction

Modeling water infiltration in unsaturated porous media plays a critical role in numerous environmental and engineering applications, including irrigation management (Saccon, 2018), contaminant transport (Essaid *et al.*, 2015), groundwater recharge estimation (Singh *et al.*, 2019), and many others. Mathematically, water infiltration is typically described by a continuum model in the form of an initial-boundary value problem governed by a partial differential equation (PDE). The most commonly used model is the Richards equation (Richards, 1931), a highly nonlinear PDE that generalizes the classical Darcy–Buckingham law to account for unsaturated flow conditions. However, a key limitation of the Richards equation is its inability to reproduce saturation overshoot and gravity-driven fingers, as documented in experimental studies (see, e.g., DiCarlo (2004); Glass *et al.* (1989)).

Saturation overshoot is fundamentally a one-dimensional phenomenon that refers to

the situation where the water saturation at the tip of the wetting front exceeds the saturation level behind it. In higher dimensions, this phenomenon leads to the formation of gravity-driven preferential flow paths known as fingers. In this study, we focus on saturation overshoot, as it has been established as a necessary and sufficient condition for the development of gravity-driven fingering (DiCarlo, 2004; Eliassi and Glass, 2001).

Several extensions of the Richards equation have been proposed to capture saturation overshoot and gravity-driven fingering during infiltration in dry porous media (see, e.g., DiCarlo (2013); Xiong (2014); Cueto-Felgueroso and Juanes (2009); Eliassi and Glass (2002)). Among these, Beljadid, Cueto-Felgueroso, and Juanes (Beljadid *et al.*, 2020) developed a new continuum model for unstable infiltration, in which the authors proposed a new PDE (Equation 18 in (Beljadid *et al.*, 2020)) endowed with an entropy function. This equation, referred to as the BCJ-R equation, extends the classical Richards formulation by incorporating an entropy function and is derived within the framework of second-gradient theory, where the free-energy functional is constructed as a symmetric expansion of the capillary energy density in the Richards equation.

The BCJ-R equation is a highly nonlinear, high-order in space, making it mathematically and computationally challenging. First, the presence of the high-order term necessitates an additional boundary condition. Second, the strong coupling between advection, diffusion, and dynamic capillarity results in sharp moving fronts and multiple characteristic time scales, rendering explicit time integration schemes impractical and fully implicit methods computationally expensive due to ill-conditioned Jacobian matrices. Third, saturation overshoot is highly sensitive to the initial soil state and the applied water flux.

Due to the complexity of the aforementioned difficulties, it is useful to develop different classes of numerical methods for solving the BCJ-R equation. This is particularly important for the scientific community, as it helps make this highly nonlinear PDE more tractable for practical applications. The goal of this study is to develop an efficient PINN framework for solving the BCJ-R equation and modeling saturation overshoot phenomena.

PINNs have been successfully applied to a wide range of complex PDEs, including the Richards equation (Bandai and Ghezzehei, 2022; Kamil *et al.*, 2024d, 2025), and phase-field models such as Allen–Cahn and Cahn–Hilliard equations (Mattey and Ghosh, 2022), the Kuramoto–Sivashinsky chaotic dynamics (Wang *et al.*, 2023a), and thin-film flow problems (Nakamura *et al.*, 2022). A common feature among these models and the BCJ-R equation is the need to accurately satisfy both initial and boundary conditions to find the correct solution profile.

In this paper, we propose a new numerical framework for solving the BCJ-R equation by developing a PINN model (Raissi *et al.*, 2019). The PINN acts as a mesh-free solver, and no linearization or discretization of the partial differential operators is required, thanks to the use of automatic differentiation. PINNs act as self-supervised solvers that

learn the solution of forward PDE problems by enforcing the equations at collocation points, making them well-suited for high-dimensional and complex geometries. Their flexibility also enables effective treatment of inverse problems and uncertainty quantification. In inverse settings, a data loss term can be directly added to the global loss function, eliminating the additional optimization steps typically required by traditional numerical methods.

The rest of the paper is structured as follows. Section 7.2 outlines the setup for the BCJ-R equation. The methodology for the proposed PINN solver is explained in Section 8.3. The numerical results are presented and discussed in Section 7.4. Section 8.6 offers a summary of the study, key conclusions, and outline some future perspectives.

7.2 Mathematical model

7.2.1 Governing equations

In this study, we adopt the BCJ-R equation, developed by Beljadid, Cueto-Felgueroso, and Juanes [Beljadid et al. \(2020\)](#), to simulate unstable infiltration in porous media. The BCJ-R equation is endowed with an entropy function that ensures thermodynamic consistency and captures key instability-driven phenomena, including saturation overshoot in one dimension and fingered flow in higher dimensions. The governing form of the BCJ-R equation is given by [Beljadid et al. \(2020\)](#):

$$\phi \frac{\partial S}{\partial t} + \nabla \cdot \left[K(S) \nabla \left(z + \psi(S) + \sqrt{\kappa(S)} \nabla \cdot \left(\sqrt{\kappa(S)} \nabla S \right) \right) \right] = 0, \quad \text{in } (0, T] \times \Omega \quad (7.1)$$

where S is the water saturation, ϕ is the medium porosity, K represents the unsaturated hydraulic conductivity, ψ is the water suction head, and κ represents the capillary pinning function, z represents the vertical dimension (gravity axis) directed downward and t is time. The domain $\Omega \subset \mathbb{R}^d$, with $d = 1, 2, 3$, represents the porous media, and the time interval is given by $I = (0, T]$, where $T > 0$ is the final simulation time.

7.2.2 Constitutive functions

In the following, we define each constitutive function used in the BCJ-R equation. The unsaturated hydraulic conductivity K is given by:

$$K(S) = K_s k_r(S), \quad (7.2)$$

where K_s represents the saturated hydraulic conductivity, and $k_r(S)$ denotes the water relative permeability, which is defined using a power-law relation ([Brooks and Corey, 1966](#)):

$$k_r(S) = S^a \quad a > 1, \quad (7.3)$$

where a is a parameter that depends on the properties of the porous medium.

The relationship between the water suction head ψ and saturation S is described by Leverett's equation (Leverett, 1941):

$$\psi(S) = h_{\text{cap}} J(S), \quad (7.4)$$

where h_{cap} represents the capillary rise, and $J(S)$ denotes the Leverett J -function, given by:

$$J(S) = S^{-1/\alpha} \left[1 - \exp \left(\beta(S - v_e) \right) \left(1 + \beta \frac{\alpha}{\alpha - 1} S \right) \right], \quad (7.5)$$

where $\alpha > 1$, $\beta > 0$, and $v_e \geq 1$ are free parameters.

The capillary pinning function $\kappa(S)$ is given by:

$$\kappa(S) = h_{\text{cap}} \delta^2 \frac{\alpha}{\alpha - 1} \left[1 - \exp \left(\beta(S - v_e) \right) \right] S^{(\alpha-1)/\alpha}, \quad (7.6)$$

where δ is the characteristic length scale of the diffuse interface.

Figure 7.1 illustrates the relative permeability $k_r(S)$, the Leverett J -function $J(S)$, and the capillary pinning function $\kappa(S)$. The BCJ-R equation exhibits strong nonlinearity with respect to saturation, particularly in the J -function near dry and full saturation, where steep gradients arise.

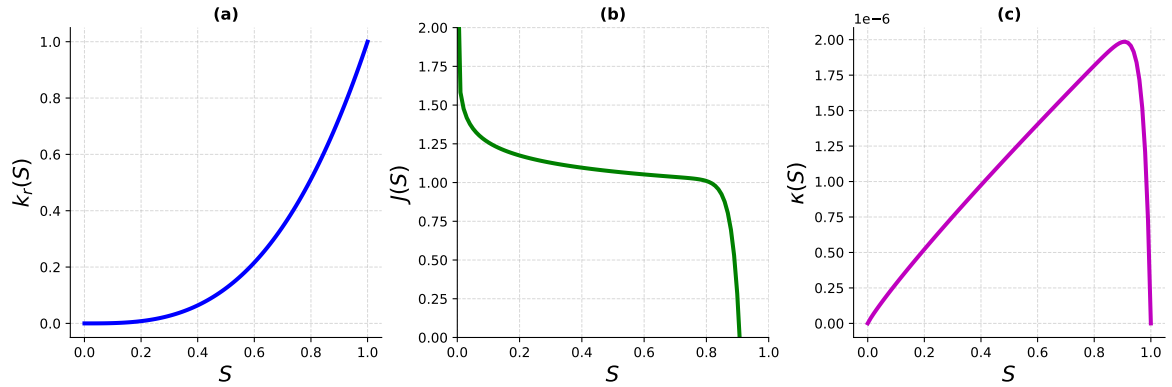


Figure 7.1 Constitutive functions of the BCJ-R equation (7.7). (a) relative permeability $k_r(S)$, (b) Leverett J -function $J(S)$, and (c) capillary pinning function $\kappa(S)$. The following parameter values are used to evaluate these functions: $\phi = 0.3$, $K_s = 0.4$, $h_{\text{cap}} = 0.02$, $a = 3.0$, $v_e = 1.0$, $\alpha = 10.0$, $\beta = 40.0$, and $\delta = 0.01$.

7.2.3 Initial and boundary conditions

The BCJ-R equation (7.7) is reformulated as a system of two coupled second-order equations:

$$\begin{cases} \phi \frac{\partial S}{\partial t} + \nabla \cdot [K(S) \nabla (z + w)] = 0, \\ w = \psi(S) + \sqrt{\kappa(S)} \nabla \cdot (\sqrt{\kappa(S)} \nabla S). \end{cases} \quad (7.7)$$

where w is an auxiliary variable introduced to rewrite the original high-order equation as a system of two second-order partial differential equations.

To ensure that the system is well-posed, it must be supplemented with appropriate initial and boundary conditions. The following boundary conditions are considered:

$$\begin{cases} S = S_D & \text{on } \Gamma_D, \\ \nabla w \cdot \mathbf{n} = 0 & \text{on } \partial\Omega, \\ \nabla S \cdot \mathbf{n} = 0 & \text{on } \Gamma_N, \end{cases} \quad (7.8)$$

along with the initial condition given in terms of saturation as:

$$S(t = 0, \mathbf{x}) = S_0(\mathbf{x}). \quad (7.9)$$

The boundary $\partial\Omega$ represents the soil domain boundary, while Γ_D is the Dirichlet boundary where water is applied, and Γ_N is the Neumann boundary corresponding to the remaining part of the domain. At Γ_D , the two imposed conditions, $S = S_D$ and $\nabla w \cdot \mathbf{n} = 0$, imply that water flux is given by $R_s K_s$, where the flux ratio is defined as $R_s = k_r(S_D)$.

7.2.4 Entropy function

The BCJ-R equation (Beljadid *et al.*, 2020) is endowed with an entropy function that serves as a mathematical mechanism to ensure the well-posedness and thermodynamic consistency of the model, particularly in cases where weak solutions arise and uniqueness is not guaranteed. Beljadid, Cueto-Felgueroso, and Juanes (Beljadid *et al.*, 2020) derived the entropy function of the BCJ-R equation and demonstrated that it satisfies an entropy-increasing property for isolated systems. In Section 7.4.2, we aim to validate this physical constraint using our PINN model.

We consider the entropy function of the BCJ-R equation over a one-dimensional domain $[0, L]$ at a given time $t \geq 0$, defined as (Beljadid *et al.*, 2020):

$$\mathcal{E}(t) = - \int_0^L U(S(t, z)), dz, \quad (7.10)$$

where $U(S)$ is a convex entropy function of the saturation that satisfies the following relation Beljadid *et al.* (2020):

$$U''(S) = \frac{V_c \sqrt{\kappa(S)}}{f(S)}, \quad (7.11)$$

with $V_c > a$ constant, and $f(S) = \frac{K_s, k_r(S)}{\phi}$ is the flux function.

Integrating Eq. (7.11) twice yields the entropy function:

$$U(S) = \int_S^1 s \frac{\sqrt{\kappa(s)}}{f(s)} ds - s \int_S^1 \frac{\sqrt{\kappa(s)}}{f(s)} ds \quad (7.12)$$

up to an affine function.

We note that any affine function remains constant over time in a closed porous medium, and can therefore be ignored when verifying the entropy-increasing property of $U(S)$. In Section 7.4.2, we compute the entropy difference $\mathcal{E}(t) - \mathcal{E}(0)$ in a closed porous medium using our PINN framework, in order to verify that it satisfies the entropy-increasing property.

7.3 Methods

In the following sections, an overview of PINNs is presented, along with the enhanced techniques adopted in this study for modeling unstable infiltration using the BCJ-R equation.

7.3.1 Overview of physics-informed neural networks

PINNs (Raissi *et al.*, 2019) are mesh-free, neural network-based solvers designed to approximate solutions of differential equations. They are trained by minimizing a multi-objective loss function that enforces the governing equations as soft constraints, alongside the initial and boundary conditions. PINNs have garnered considerable attention for their flexibility and ability to operate without requiring labeled data.

The core idea behind PINNs is straightforward. Consider a time-dependent PDE of the form:

$$u_t + \mathcal{N}(u) = 0 \quad \text{in } (0, T] \times \Omega. \quad (7.13)$$

subject to the following initial and boundary conditions:

$$\mathcal{B}(u) = 0, \quad \text{on } (0, T] \times \partial\Omega, \quad (7.14a)$$

$$u = u_0, \quad \text{in } \{t = 0\} \times \Omega. \quad (7.14b)$$

The solution of the PDE (7.13), constrained by (7.14), can be approximated using a physics-informed neural network $u(\cdot; \Theta)$. This neural network takes time-space collocation points $\mathbf{x}_t = (t, \mathbf{x})$ sampled from the domain $(0, T] \times \Omega$ as input and outputs the approximate PDE solution.

Training the network $u(\mathbf{x}; \Theta)$ requires solving a nonconvex optimization problem by minimizing a weighted loss function:

$$\mathcal{L}(\Theta) = \alpha_r \mathcal{L}_r(\Theta) + \alpha_{ic} \mathcal{L}_{ic}(\Theta) + \alpha_{bc} \mathcal{L}_{bc}(\Theta), \quad (7.15)$$

where $\mathcal{L}_r$, $\mathcal{L}_{ic}$, and $\mathcal{L}_{bc}$ represent the residual, initial, and boundary condition losses, respectively, weighted by α_r , α_{ic} , and α_{bc} . Each term is defined as:

$$\mathcal{L}_r(\Theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{N}(u(\mathbf{x}_t^i; \Theta)) \right|^2, \quad (7.16a)$$

$$\mathcal{L}_{ic}(\Theta) = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} \left| u(\mathbf{x}_t^i; \Theta) - u_0(\mathbf{x}_t^i) \right|^2, \quad (7.16b)$$

$$\mathcal{L}_{bc}(\Theta) = \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} \left| \mathcal{B}(u(\mathbf{x}_t^i; \Theta)) \right|^2. \quad (7.16c)$$

where N_r , N_{ic} , and N_{bc} represent the number of collocation points used to evaluate the residual, initial, and boundary conditions, respectively. Note that the operators $\mathcal{N}$ and $\mathcal{B}$ are evaluated using automatic differentiation.

7.3.2 Enhancements in PINNs

The original PINN formulation (Raissi *et al.*, 2019) often fails to converge when applied to complex PDEs, particularly in cases involving stiffness, sharp gradients, or multiple scales (Wang *et al.*, 2022a). To address these challenges, various enhancements have been proposed to improve the performance and robustness of PINNs. A comprehensive overview of these advancements can be found in (Faroughi *et al.*, 2024). In this section, we present the specific techniques adopted in this study to enhance the performance of the PINN model.

7.3.2.1 Forward pass

Architecture. Most PINN studies rely on the standard multi-layer perceptron (MLP) architecture, which is generally effective for simple PDEs but tends to underperform on more complex problems. To overcome these limitations, alternatives such as the modified MLP (mMLP) (Wang *et al.*, 2021a) and PirateNet (Wang *et al.*, 2024a) have been proposed. For our problem, we found that the mMLP offers a favorable trade-off between accuracy and computational efficiency, and is therefore adopted. The forward pass of the mMLP is described in Appendix 7.5.

Fourier embedding. To capture the complex spatial patterns present in the BCJ-R solution, we found that incorporating Fourier features into our PINN model is highly effective. The saturation profile exhibits a wide spectrum of spatial frequencies, including sharp overshoots at the tip of the finger and fine-scale undershoots trailing the wetting front. These localized and high-frequency features are particularly challenging for standard neural networks to approximate accurately. This difficulty stems from the well-known spectral bias of deep learning models, which tend to prioritize learning low-frequency components and struggle to resolve high-frequency variations.

Fourier embeddings mitigate this issue by mapping the original input coordinates into a

higher-dimensional feature space using sinusoidal random basis functions. This transformation enhances the network’s capacity to represent high-frequency components of the solution (Tancik *et al.*, 2020), making training more efficient and significantly improving accuracy in regions with sharp gradients or localized structures. Further details about the Fourier embedding used in our model are provided in Appendix 7.5.

Random weight factorization. In deep neural networks, weights and biases can vary in scale across layers, which may hinder effective training. To improve convergence, we adopt the random weight factorization method proposed in (Wang *et al.*, 2022c). This technique has been shown to reduce spectral bias and help the optimizer avoid poor local minima in complex loss landscapes. In this study, only the weights are factorized, while the original biases are preserved.

The RWF method factorizes each weight matrix $\mathbf{W}_l$ as follows:

$$\mathbf{W}_l = \text{diag}(\exp(\mathbf{s}_l))\mathbf{V}_l, \quad (7.17)$$

where $\mathbf{s}_l \in \mathbb{R}^{d_l}$ is a trainable scaling vector, and $\mathbf{V}_l \in \mathbb{R}^{d_l \times d_{l-1}}$ is a trainable weight matrix.

The scaling vectors for each hidden layer are initialized from a multivariate normal distribution $\mathcal{N}(\mu, \sigma I)$, where μ and σ are hyperparameters, and I is the identity matrix. The optimizer is applied directly to the parameters $\mathbf{s}_l$ and $\mathbf{V}_l$.

Locally adaptive activation function. In the BCJ-R equation, the saturation profile exhibits a steep gradient near the wetting front. It has been shown that introducing a trainable parameter in the activation function can improve network performance by dynamically adjusting the loss landscape, leading to better convergence than fixed activations (Jagtap *et al.*, 2020). This approach has also proven to be effective for infiltration problems governed by the Richards equation (Bandai and Ghezzehei, 2022; Kamil *et al.*, 2024d, 2025).

The locally adaptive activation is defined as:

$$\sigma_{\text{adaptive}}(x) = \tanh(s\alpha x), \quad (7.18)$$

where s is a fixed scaling hyperparameter and α is a trainable parameter. Each hidden layer has its own α , while $s = 20$ is used consistently across all layers.

7.3.2.2 Respecting temporal causality

PINNs are time–space methods that learn the full solution simultaneously, without prioritizing earlier time steps. This lack of temporal causality can hinder convergence, especially for complex PDEs. The BCJ-R equation is a high-order phase-field equation, similar in structure to the Allen–Cahn and Cahn–Hilliard equations, which are known to be challenging for PINNs due to spatiotemporal heterogeneity. Recent studies have shown that enforcing temporal causality, either in a hard form (Mattey and Ghosh,

2022) or a soft form (Wang *et al.*, 2022a), can significantly improve the training and predictive accuracy of PINNs for complex models. In this study, both approaches are adopted for the BCJ-R equation and are presented in Appendix 7.5.

7.3.2.3 Adaptive loss weighting

Training PINNs is a multi-objective optimization problem involving the PDE residual, initial condition, and boundary condition losses. These objectives can compete during training, making it important to balance their contributions. Prior studies have shown that adapting the weights of each objective improves convergence (Wang *et al.*, 2021a). In this work, we use the gradient normalization method (Wang *et al.*, 2023a), which adjusts each weight based on the relative gradient magnitude. This ensures that all objectives contribute equally in terms of gradient scale. Details are provided in Appendix 7.5.

7.3.2.4 Curriculum training

In the BCJ-R equation, the problem becomes increasingly difficult as the initial saturation approaches zero. This leads to larger overshoots and steeper infiltration fronts, making it challenging to model dry porous media accurately. Despite the enhancements introduced earlier, the PINN model still struggles under such conditions. To address this, we adopt a curriculum training strategy (Krishnapriyan *et al.*, 2021), where the model is first trained on simpler cases and then gradually adapted to harder ones by transferring the learned parameters. Specifically, we start training with relatively wet soils (initial saturation around 0.3) and progressively reduce the initial saturation until reaching the target case.

7.3.2.5 Random collocation sampling

To train the PINN for the BCJ-R equation, we employ a random batch collocation strategy based on uniform sampling. The initial and boundary condition points are first sampled on a fixed grid and remain unchanged throughout training, whereas the residual points are randomly re-sampled at each optimizer iteration. This approach is designed to enhance training stability, mitigate overfitting to fixed collocation sets, and facilitate efficient data-parallel training.

7.3.2.6 Optimiser

Training a PINN involves solving a highly nonconvex optimization problem, which is generally ill-conditioned (Rathore *et al.*, 2024). As a result, an efficient optimization scheme is necessary to avoid poor local minima that lead to inaccurate solutions. Another difficulty is that PINNs are inherently multi-task learning models, where multiple loss terms must be minimized simultaneously. This leads to competitive training dynamics and gradient conflicts, making optimization particularly challenging.

Recently, a new optimizer called SOAP (Vyas *et al.*, 2024) was proposed and further studied in the context of PINNs (Wang *et al.*, 2025). SOAP belongs to the family of Shampoo optimizers and is a quasi-Newton method that uses a Hessian-based preconditioner to naturally balance conflicting gradients and accelerate convergence. In our experiments, SOAP consistently outperformed commonly used first-order methods such as Adam, in terms of accuracy and stability. Therefore, we use SOAP in all simulations presented in this study. Further details about SOAP optimizer can be found in (Vyas *et al.*, 2024).

7.3.2.7 Data parallelism

To accelerate the training of the PINN model, we use data parallelism across multiple GPUs. The model is replicated on each device, and the residual collocation points are split among them. During each training step, each GPU computes local gradients independently. These gradients are then aggregated using an all-reduce operation to update the model parameters synchronously. Weak and strong scaling results for this parallel training strategy are presented in Appendix 7.5.

7.3.3 PINNs for the BCJ-R equation

In the following, we present the adopted PINN formulation specific to the BCJ-R equation.

We use the coupled system (7.7) to reduce the computational cost of automatic differentiation associated with the original high-order equation. The mMLP architecture is then applied, with an output dimension of two: one for the predicted saturation $s(\mathbf{x}_t, \Theta)$, and one for the auxiliary variable $w(\mathbf{x}_t, \Theta)$. To enforce the physical constraint that $S \in (0, 1)$, a sigmoid activation is used in the output layer:

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \in (0, 1), \quad (7.19)$$

while a linear output activation is used for w .

The total loss function for the BCJ-R equation is defined as a weighted sum of seven components:

$$\mathcal{L}(\Theta) = \alpha_{\text{ic}} \mathcal{L}_{\text{ic}}(\Theta) + \sum_{i=1}^2 \left(\alpha_{\text{res}_i} \mathcal{L}_{\text{res}_i}(\Theta) + \alpha_{\text{up}_i} \mathcal{L}_{\text{up}_i}(\Theta) + \alpha_{\text{dw}_i} \mathcal{L}_{\text{dw}_i}(\Theta) \right), \quad (7.20)$$

where each loss component is described as follows:

$$\mathcal{L}_{\text{ic}}(\Theta) = \frac{1}{N_{\text{ic}}} \sum_{i=1}^{N_{\text{ic}}} |s(t=0, \mathbf{x}_i; \Theta) - s_0(\mathbf{x}_i)|^2. \quad (7.21)$$

The residual loss $\mathcal{L}_{\text{res}_1}(\Theta)$ corresponds to the mass conservation equation and is com-

puted as

$$\mathcal{L}_{\text{res}_1}(\Theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \phi \frac{\partial s}{\partial t}(\mathbf{x}_t^{(i)}, \Theta) + \nabla \cdot \left[K(s(\mathbf{x}_t^{(i)}, \Theta)) \nabla (z + w(\mathbf{x}_t^{(i)}, \Theta)) \right] \right|^2. \quad (7.22)$$

The second residual loss $\mathcal{L}_{\text{res}_2}(\Theta)$ enforces the nonlinear relationship defining the auxiliary variable w :

$$\mathcal{L}_{\text{res}_2}(\Theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| w(\mathbf{x}_t^{(i)}, \Theta) - \psi(s(\mathbf{x}_t^{(i)}, \Theta)) - \sqrt{\kappa(s(\mathbf{x}_t^{(i)}, \Theta))} \nabla \cdot \left(\sqrt{\kappa(s(\mathbf{x}_t^{(i)}, \Theta))} \nabla s(\mathbf{x}_t^{(i)}, \Theta) \right) \right|^2. \quad (7.23)$$

We note that when incorporating temporal causality, the residual loss is reformulated as in Equation (7.39).

The boundary condition losses enforce the constraints on Γ_D , Γ_N , and $\partial\Omega$. Specifically, the Dirichlet condition on saturation is imposed via $\mathcal{L}_{\text{up}_1}(\Theta)$:

$$\mathcal{L}_{\text{up}_1}(\Theta) = \frac{1}{N_{\text{up}_1}} \sum_{i=1}^{N_{\text{up}_1}} \left| s(\mathbf{x}_t^{(i)}, \Theta) - S_D(\mathbf{x}_t^{(i)}) \right|^2, \quad (7.24)$$

while the Neumann condition on the auxiliary variable w is captured by $\mathcal{L}_{\text{up}_2}(\Theta)$:

$$\mathcal{L}_{\text{up}_2}(\Theta) = \frac{1}{N_{\text{up}_2}} \sum_{i=1}^{N_{\text{up}_2}} \left| \nabla w(\mathbf{x}_t^{(i)}, \Theta) \cdot \mathbf{n} \right|^2. \quad (7.25)$$

The two losses at the bottom boundary enforce homogeneous Neumann conditions. The term $\mathcal{L}_{\text{dw}_1}(\Theta)$ corresponds to the flux condition on s :

$$\mathcal{L}_{\text{dw}_1}(\Theta) = \frac{1}{N_{\text{dw}_1}} \sum_{i=1}^{N_{\text{dw}_1}} \left| \nabla s(\mathbf{x}_t^{(i)}, \Theta) \cdot \mathbf{n} \right|^2, \quad (7.26)$$

and $\mathcal{L}_{\text{dw}_2}(\Theta)$ applies the zero-flux condition on w :

$$\mathcal{L}_{\text{dw}_2}(\Theta) = \frac{1}{N_{\text{dw}_2}} \sum_{i=1}^{N_{\text{dw}_2}} \left| \nabla w(\mathbf{x}_t^{(i)}, \Theta) \cdot \mathbf{n} \right|^2. \quad (7.27)$$

For all numerical experiments, the initial loss weights are set as follows:

$$\alpha_{\text{ic}} = \alpha_{\text{up}_1} = 100, \alpha_{\text{res}_1} = \alpha_{\text{dw}_1} = \alpha_{\text{dw}_2} = \alpha_{\text{up}_2} = 1, \alpha_{\text{res}_2} = 5. \quad (7.28)$$

A schematic design of the proposed PINN for modeling unstable infiltration in porous media using the BCJ-R equation is illustrated in Figure 7.2.

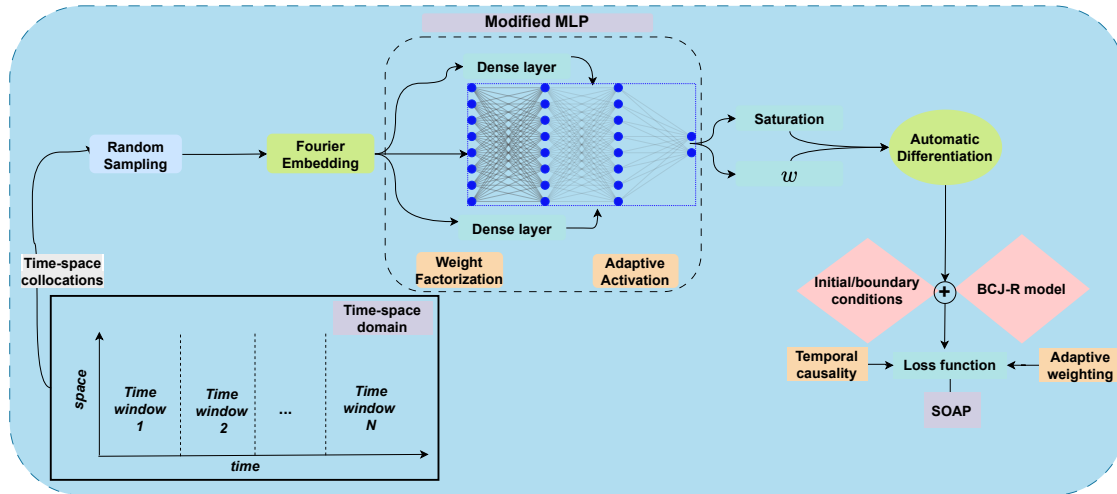


Figure 7.2 Schematic representation of the PINN framework for solving the BCJ-R equation of unstable infiltration in porous media.

7.3.4 Implementation

The PINN algorithm was implemented using the JAX library, chosen for its efficiency and scalability in training deep learning models. JAX provides powerful tools such as just-in-time compilation (JIT), vectorized function mapping (`vmap`), and data-parallel training through the `pmap` function, as well as XLA (Accelerated Linear Algebra). Our implementation builds upon the open-source codebase `jaxpi` (Wang *et al.*, 2023a), which includes many of the features described earlier. For both the Adam and SOAP optimizers, we use the following hyperparameters:

$$\beta_1 = 0.9, \quad \beta_2 = 0.999, \quad \eta = 10^{-3}, \quad \text{decay rate} = 0.9, \quad \text{decay steps} = 5000. \quad (7.29)$$

We note that early stopping is applied, forcing the model to stop training if no reduction in the total loss is observed after 10^3 iterations.

All numerical experiments were conducted on the Narval supercomputing cluster, part of the Digital Research Alliance of Canada, using an NVIDIA A100 GPU.

7.4 Results

We use the PINN model to solve the BCJ-R equations. The results are evaluated against either finite element simulations () or experimental data from the literature, using the relative L^2 error as a measure of accuracy.

7.4.1 Saturation overshoot simulation

In this section, we use the proposed PINN model to solve the BCJ-R equation and reproduce the saturation overshoot phenomenon observed at the wetting front. We aim to verify that our PINN model reproduces the previously reported behavior showing that saturation overshoot diminishes as the initial water saturation increases (Beljadid *et al.*, 2020; Cueto-Felgueroso and Juanes, 2009; DiCarlo, 2004).

We consider a sandy soil column with an initial uniform saturation S_0 , which is varied to assess its influence on the height of the overshoot. The domain consists of a 0.6 m soil column with a water inlet boundary condition of $S_D = 0.457$ at the surface ($z = 0$ m) and free drainage at the bottom.

The BCJ-R equation parameters used in this simulation are set as follows:

$$\phi = 0.3, K_s = 40 \text{ cm/min}, h_{\text{cap}} = 2 \text{ cm}, a = 5, \quad (7.30)$$

$$v_e = 1.0, \alpha = 10, \beta = 40, \delta = 2 \text{ cm}.$$

The selected PINN hyperparameters and training setup are summarized in Table 7.1. We present results for three initial saturation values: $S_0 = 0.2$, $S_0 = 0.1$, and $S_0 = 0.01$, comparing the PINN predictions with the reference FEM solutions.

The results are presented in Figure 7.3. Overall, good agreement is observed between the PINN predictions and the FEM reference solutions across all three test cases. However, larger prediction errors are noted in drier soils, particularly along the infiltration front where steep saturation gradients occur. As the initial soil saturation S_0 decreases, the saturation overshoot becomes more pronounced, underscoring the strong dependence of this phenomenon on initial moisture conditions. Therefore, we conclude that simulations with higher initial water saturation lead to reduced saturation overshoot profiles.

The results also show that the width of the saturation finger tip increases as S_0 decreases from 0.2 to 0.01, indicating more severe overshoot in drier soils. As infiltration progresses, the finger slowly increases in width with depth. This behavior is mainly attributed to lateral capillary diffusion effects (Beljadid *et al.*, 2020) and the absence of considering hysteresis in the BCJ-R equation.

The PINN model successfully captures fine-scale saturation features, including small fluctuations downstream of the wetting front, as well as the structure near the finger tip. We have noticed that the PINN training process and the FEM simulation become more challenging as S_0 decreases. For the driest scenario ($S_0 = 0.01$), a curriculum training strategy was required to achieve satisfactory PINN performance (see Section 7.3.2.4 for more details). Moreover, the number of collocation points, temporal windows, and total training time were increased to ensure convergence. These observations highlight the increasing complexity of solving the BCJ-R equation under dry initial conditions. On the other hand, increasing the flux ratio R_s produces a gravity-dominated regime

with fast advection and strong nonlinear effects. For larger R_s values, the training of PINN became unstable, and convergence could not be achieved even after increasing the number of collocation points, temporal windows, and total training iterations. We also note that, due to memory limitations, it was not possible to use a larger number of collocation points. This limitation requires further investigation and the development of improved training strategies to alleviate it.

7.4.2 Entropy test analysis

In the following numerical test, we validate the ability of the proposed PINN model to respect the entropy dissipation property of the BCJ-R equations. Specifically, we examine a water redistribution scenario in a closed soil domain, where the mathematical entropy function is expected to decrease over time.

We consider a vertical soil column with no-flux boundary conditions at both ends. The initial saturation distribution is shown in Figure 7.4, and the system is allowed to evolve without external inflow or outflow. We monitor the total entropy of the system over time to assess whether the PINN model correctly captures the dissipative nature of the system.

The BCJ-R equation parameters used in this simulation are set as follows:

$$\phi = 0.3, K_s = 40 \text{ cm/min}, h_{\text{cap}} = 2 \text{ cm}, a = 3, \quad (7.31)$$

$$v_e = 1.0, \alpha = 10, \beta = 40, \delta = 1 \text{ cm}.$$

The training configuration and hyperparameters of the PINN model are summarized in Table 7.1. The results are shown in Figure 7.4, where we plot the evolution of the difference entropy $\mathcal{E}(t) - \mathcal{E}(0)$. We observe that the entropy predicted by the PINN model decreases monotonically over time, in agreement with the theoretical results reported in (Beljadid *et al.*, 2020), confirming that the BCJ-R equation exhibits dissipative behavior under closed conditions.

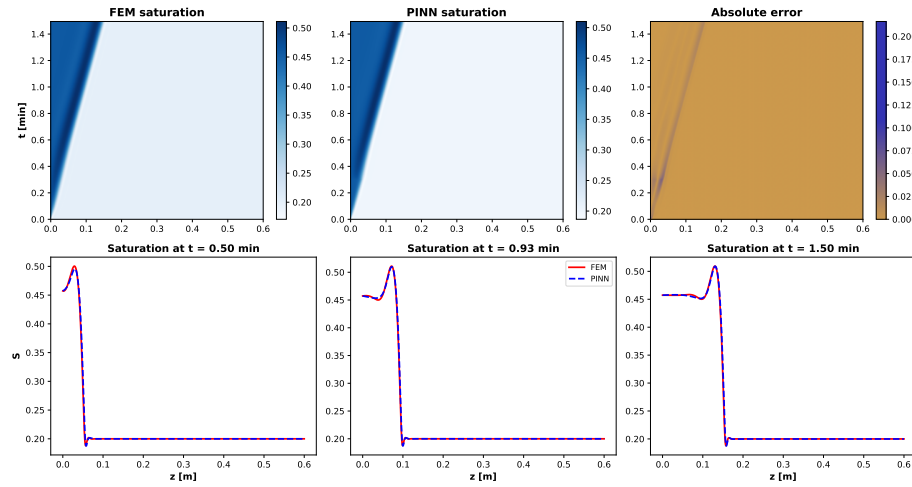
7.4.3 Comparison with experimental data

In this section, we validate the PINN model predictions against laboratory measurements of saturation overshoot from the experiments conducted by DiCarlo (DiCarlo, 2004). The experimental setup consists of a sandy soil column with porosity $\phi = 0.35$ and saturated hydraulic permeability $K_s = 15 \text{ cm/min}$, initially maintained at a dry saturation state $S_0 = 0.01$. Different water inlet fluxes, characterized by various values of dimensionless saturation S_D , are applied to systematically investigate their influence on the shape and temporal development of the saturation overshoot phenomenon. The relative permeability function is modified following the form (Cueto-Felgueroso and Juanes, 2009):

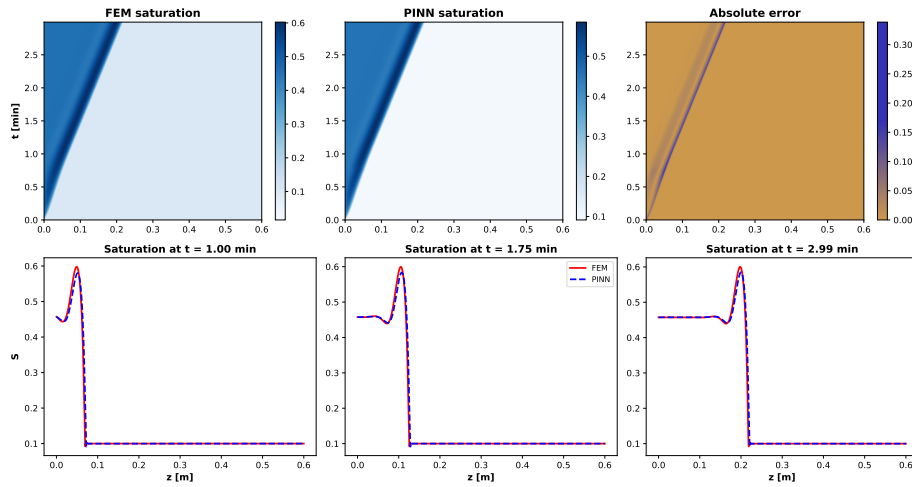
$$k_r(S) = S^a h(S) + S^b (1 - h(S)), \quad (7.32)$$

Figure 7.3 Saturation profiles for different initial conditions. Each panel shows the FEM reference, PINN prediction, and absolute error in the time-space domain, along with cross-sectional comparisons at selected times.

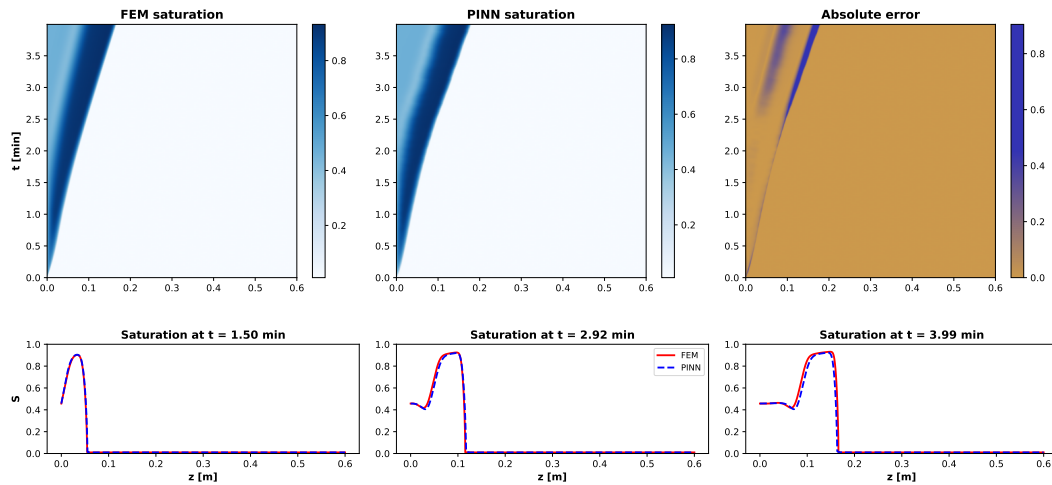
Initial saturation $S_0 = 0.2$



Initial saturation $S_0 = 0.1$



Initial saturation $S_0 = 0.01$



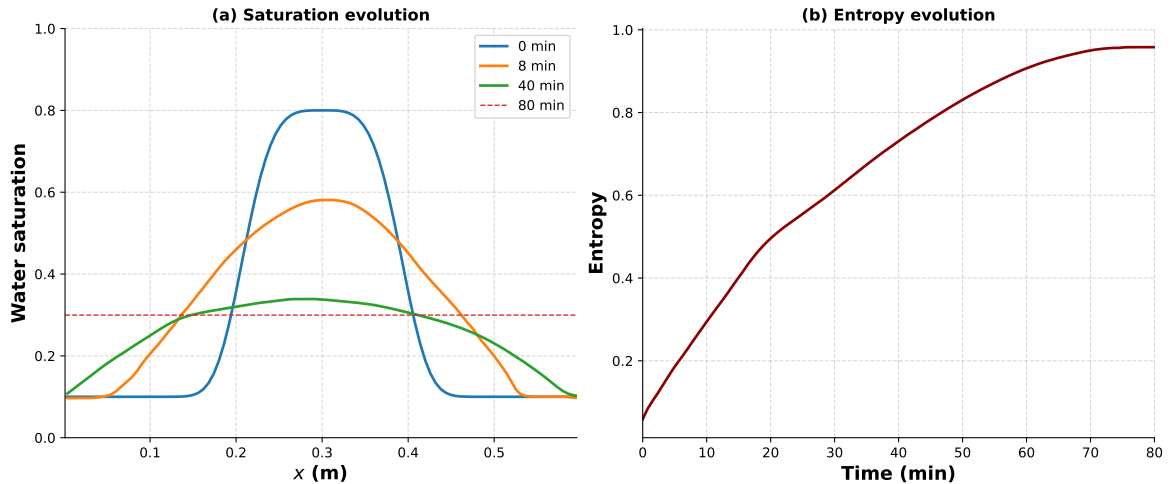


Figure 7.4 Entropy test analysis. (a) Saturation profiles during water redistribution in a closed 1D soil system. (b) Evolution of the difference entropy, $\mathcal{E}(t) - \mathcal{E}(0)$, which increases monotonically over time.

with parameters $a = 11$, $b = 3$, and the switching function defined as:

$$h(S) = 0.5 \tanh(10(S - 0.22)). \quad (7.33)$$

In the DiCarlo experiment (DiCarlo, 2004), the van Genuchten model was used for defining the capillary pressure model. To fit this model to match the generalized Brooks-Corey model (7.4), we conduct a parameter optimization using mean squared error minimization. The target van Genuchten model is characterized by parameters $n_v = 6.23$, $m_v = 1 - 1/n_v$, and $\alpha_{vg} = 0.177 \text{ cm}^{-1}$, which serves as the reference curve for our fitting procedure. The optimization process targets three parameters: α , β , and h_{cap} , which are constrained within the following bounds: $1 \leq \alpha \leq 10$, $40 \leq \beta \leq 100$, and $1 \leq h_{\text{cap}} \leq 10 \text{ cm}$. The optimization domain is restricted to the saturation range $10^{-3} \leq S \leq 0.99$ to avoid potential negative values in the generalized Brooks-Corey model at extreme saturation levels. The fitting procedure employs the Levenberg-Marquardt algorithm, with an initial parameter guess based on the van Genuchten model characteristics. We fix $v_e = 1.025$ and set $\delta = 0.5h_{\text{cap}}$. The optimization converges to the following parameter set:

$$h_{\text{cap}} = 4.8 \text{ cm}, \quad \alpha = 4.4, \quad \beta = 100. \quad (7.34)$$

A comparison between the van Genuchten curve and the generalized Brooks-Corey is given in Figure 7.5a.

The training configuration and hyperparameters of the PINN model are provided in Table 7.1. The comparison between the PINN predictions and DiCarlo's experimental

data is illustrated in Figure 7.5b. The PINN model demonstrates good agreement with the measurements across different water flux scenarios. Although our model demonstrates the capability to match the observed data, parameter calibration was not performed in this study. Our primary focus remains on validating the model in a data-free regime, rather than optimizing parameters to perfectly fit experimental profiles.

Notably, saturation overshoot is observed for moderate water fluxes. In the case of high fluxes, the BCJ-R equation yields results similar to those obtained with the classical Richards equation, where the overshoot diminishes. For intermediate flux values, the overshoot finger becomes more pronounced, exhibiting a wider tip. The PINN model successfully captures both the large- and small-scale features of the experimental data. These results agree well with the analysis given in (DiCarlo, 2004).

7.5 Conclusion

In this study, we proposed a novel numerical framework based on PINNs to model the water saturation overshoot phenomenon by solving the highly nonlinear high-order BCJ-R equation. To address the inherent challenges in training PINNs for complex systems, we integrated several techniques. The core architecture consists of a modified multilayer perceptron enhanced with random Fourier feature embeddings to mitigate spectral bias, and weight factorization to improve convergence. We also enforced temporal causality through both hard and soft mechanisms: the hard constraint was imposed by dividing the time domain into sub-windows and training separate PINNs sequentially, while the soft constraint involved masking later time intervals to emphasize earlier ones during training within each window.

To ensure balanced optimization across multiple loss components, we adopted an adaptive loss weighting strategy based on gradient normalization. Additionally, the SOAP optimizer was employed, demonstrating superior performance compared to Adam. A data-parallel training scheme was also implemented, enabling different collocation points to be distributed across multiple GPUs. Both weak and strong scaling analyses confirmed the good scalability of the proposed PINN algorithm.

Comparative numerical experiments against a finite element solution verified the capability of the proposed PINN model to capture key features of the BCJ-R dynamics, including saturation overshoot and the downstream tail in the saturation profile. We observed that training is particularly difficult for initially dry soils. To overcome this, a curriculum learning strategy was applied by transferring knowledge from models trained on initially wet soils. Furthermore, we validated the PINN’s ability to respect the dissipative nature of closed soil systems, as evidenced by the increasing physical entropy over time. Finally, comparison with laboratory data showed good agreement, further confirming the model’s predictive capability.

A limitation of the PINN model is its inability to reproduce saturation overshoot for

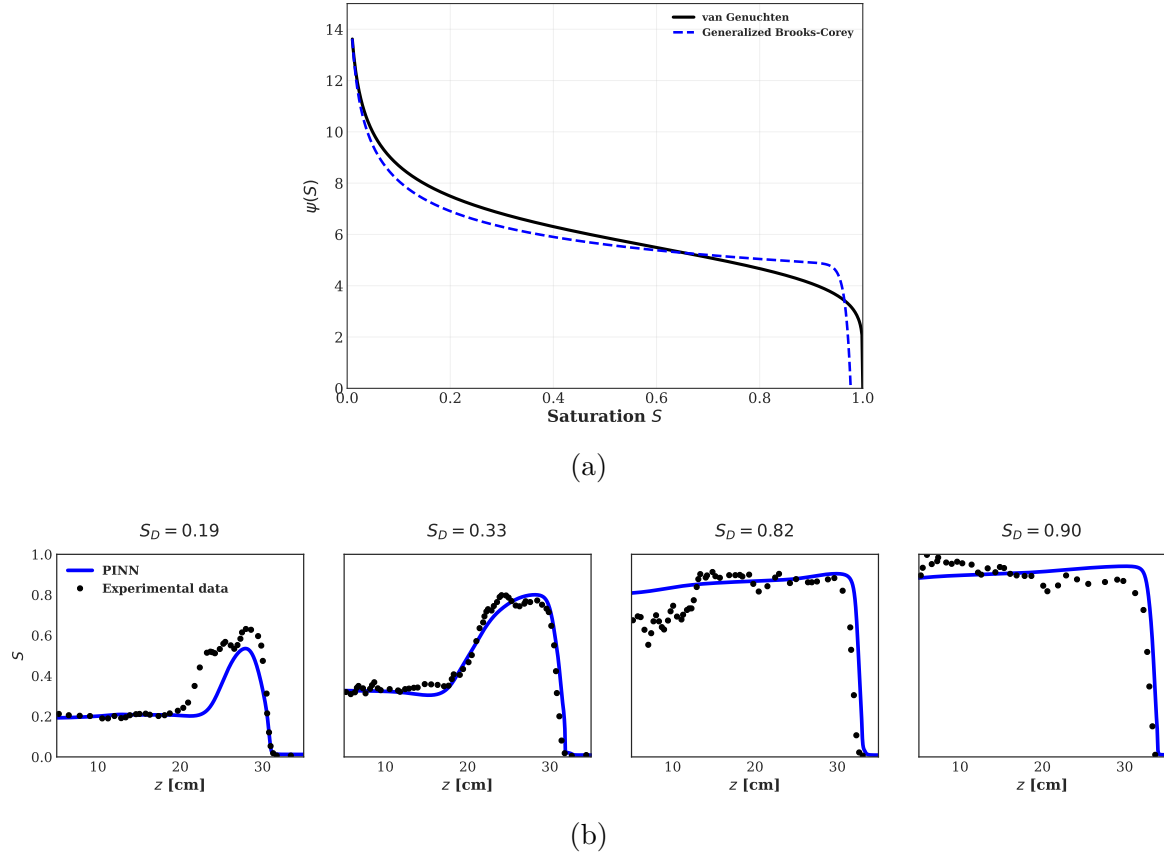


Figure 7.5 Model parameter fitting and validation: (a) Parameter optimization results showing the fit between the van Genuchten model and the generalized Brooks-Corey model. The optimization yields $h_{\text{cap}} = 4.8$ cm, $\alpha = 4.4$, and $\beta = 100$. (b) PINN model validation against DiCarlo experimental data (DiCarlo, 2004). Comparison of predicted saturation profiles with laboratory measurements under four different water flux inlets.

large flux ratios R_s , which generate wider finger tips that are complex to capture even with the FEM that require small time steps and fine spatial meshes. This issue could not be resolved within the current computational constraints, as memory limitations prevented the use of denser collocation points. Further work is needed to develop more robust training strategies and adaptive formulations to alleviate this limitation.

This study focused on one-dimensional cases, and we plan to extend the framework to two dimensions. Despite the proposed enhancements, our PINN models still struggle to capture complex finger formation patterns in 2D flow. To address this, we plan to combine PINNs with finite element methods using a variational PINN framework, which is more effective for resolving sharp wetting fronts and fine-scale gradients. We also aim to tackle inverse problems by calibrating soil parameters from experimental data to enhance the model's practical relevance and reliability.

All previous chapters of this thesis have focused on developing forward solvers or inversion frameworks to estimate unknown PDE parameters or initial and boundary conditions to model water-solute transport systems in unsaturated soils. In the next chapter, we shift focus to a multi-query setting, where the parametric Richards equation must be solved under varying initial and boundary conditions. The proposed approach is based on a data-free, mesh-independent, physics-informed DeepONet, trained as a surrogate model. The developed DeepONet is evaluated for its interpolation, extrapolation, and inverse modeling capabilities.

Appendix A. PINN model details

Modified MLP forward pass.

The model input consists of time–space collocation points $\mathbf{x}_t$, which are mapped to a higher-dimensional feature space of size N_F using random Fourier embedding (Tancik *et al.*, 2020):

$$\Phi(\mathbf{x}_t) = [\cos(\mathbf{B}\mathbf{x}_t), \sin(\mathbf{B}\mathbf{x}_t)]^T, \quad (7.35)$$

where $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^{N_F}$ is the encoding function and $\mathbf{B} \in \mathbb{R}^{N_F \times 2}$ is a random matrix sampled from a Gaussian distribution $\mathcal{N}(0, \zeta^2)$.

The embedded coordinates $\Phi(\mathbf{x}_t)$ are fed into two dense layers to compute gating functions:

$$G_1 = \tanh(\mathbf{W}_1\Phi(\mathbf{x}_t) + \mathbf{b}_1), \quad G_2 = \tanh(\mathbf{W}_2\Phi(\mathbf{x}_t) + \mathbf{b}_2), \quad (7.36)$$

These gates control the flow of information through each hidden layer l of the mMLP, defined as:

$$\begin{aligned} z_l &= \tanh(\mathbf{W}_l h_{l-1} + \mathbf{b}_l), & (\text{Dense layer } l) \\ h_l &= z_l \odot G_1 + (1 - z_l) \odot G_2, & (\text{Gating operation}) \end{aligned} \quad (7.37)$$

where $\odot$ denotes the point-wise product and $h_0 = \mathbf{x}_t$.

The weights $\mathbf{W}_l$ are initialized using the Glorot scheme (Glorot and Bengio, 2010b), and biases $\mathbf{b}_l$ are set to zero.

Hard temporal constraint.

We adopt the time-marching scheme proposed in (Mattey and Ghosh, 2022; Wight and Zhao, 2020; Krishnapriyan *et al.*, 2021). Instead of solving the PDE over the entire time domain $(0, T]$, we split it into N equal time intervals, and train the PINN model one window at a time in sequence. Every time window $\mathcal{T}_i = [i\Delta t, (i+1)\Delta t]$ has an associated PINN with its own trainable parameters.

For the first window $\mathcal{T}_0$, the initial condition S_0 is given. Once trained, the predicted solution at $t = \Delta t$ is passed as the initial condition to the next window $\mathcal{T}_1$, and the process continues until the final window.

This strategy enforces temporal causality in a hard manner by requiring the model to learn earlier time steps before moving forward. However, local violations of causality may still occur within each time slab, as the model does not prioritize earlier points within a given window. To address this, we incorporate a soft form of temporal causality that prioritizes earlier points within each time slab, as detailed below.

Soft temporal constraint.

The collocation-based PINN approach is a time-space method, which does not inherently enforce temporal causality. Even with hard time-marching, each time slab still treats time and space jointly, which can lead to local causality violations. To address this, we adopt the soft temporal enforcement strategy proposed in (Wang *et al.*, 2022a), which reformulates the residual loss to prioritize earlier time steps.

Given a sequence of increasing time coordinates $(t_1 < \dots < t_M)$, the residual loss at time t_j is defined as:

$$\mathcal{L}_r(t_j, \Theta) = \frac{1}{N_x} \sum_{i=1}^{N_x} |\mathcal{N}(u(t_j, \mathbf{x}_i; \Theta))|^2, \quad (7.38)$$

where $(\mathbf{x}_i)_{i=1}^{N_x} \in \Omega$ are collocation points in the spatial domain.

To guide the network to focus on earlier times first, the total residual loss is rewritten as a weighted sum:

$$\mathcal{L}_r(\Theta) = \frac{1}{M} \sum_{j=1}^M w_j \mathcal{L}_r(t_j, \Theta), \quad (7.39)$$

with weights

$$w_j = \exp \left(-\varepsilon \sum_{k=1}^{j-1} \mathcal{L}_r(t_k, \Theta) \right), \quad j = 1, \dots, M, \quad (7.40)$$

where $\varepsilon > 0$ is a causality parameter controlling how strongly earlier time errors influence the weights. By design, $w_1 = 1$ and w_j decreases if earlier losses are large.

This soft enforcement encourages the PINN to learn earlier time steps before moving to later ones, allowing the solution to evolve in a causally consistent way. This approach has proven effective for phase field models such as Allen–Cahn (Wang *et al.*, 2022a). In our implementation, we set $\varepsilon = 1.0$, and $M = 16$.

Adaptive loss weighting.

The weight of each loss term i is computed in a periodic manner as follows:

$$\alpha_i = \frac{\sum_{j=1}^N \|\nabla_{\theta} \mathcal{L}_j(\theta)\|}{\|\nabla_{\theta} \mathcal{L}_i(\theta)\|}, \quad (7.41)$$

where $\|\cdot\|$ represents the L^2 norm.

According to equation (7.41), we can deduce the following equation:

$$\|\alpha_i \nabla_{\theta} \mathcal{L}_i(\theta)\| = \|\alpha_j \nabla_{\theta} \mathcal{L}_j(\theta)\|, \quad \forall i, j. \quad (7.42)$$

Therefore, the gradients of all weighted loss terms have equal magnitudes, preventing the model from prioritizing the minimization of specific terms over others during training.

The weights are updated using a moving average that takes into account the previous value of it, as follows:

$$\alpha_{new} = \eta \alpha_{old} + (1 - \eta) \alpha_{new}, \quad (7.43)$$

where η is a hyper-parameter.

In our implementation, we set the momentum coefficient to $\eta = 0.9$, and the frequency of weight updates is fixed at every 1000 iterations.

Appendix B. PINN settings of each numerical test

For all tests we have used the mMLP with tanh and 4 GPUs.

Table 7.1 PINN configurations for the studied numerical tests (see Sections 7.4.1, 7.4.2, and 7.4.3).

Parameter	Entropy test	$S_0 = 0.2$	$S_0 = 0.1$	$S_0 = 0.01$	DiCarlo test
Collocation points	4096	4096	8192	16384	8192
IBC points	1026	1026	2064	2064	2064
Architecture	3×100	3×128	3×256	4×256	3×256
Max iterations	50k	100k	150k	200k	150k
Time windows	2	5	10	40	10
Fourier embedding	$\zeta = 1, N_F = 100$	$\zeta = 1, N_F = 128$	$\zeta = 1, N_F = 256$	$\zeta = 1, N_F = 256$	$\zeta = 1, N_F = 256$

Appendix C. Ablation study

We present an ablation study to evaluate the impact of different features in the PINN solver for the BCJ-R equation. The test case involves a one-dimensional problem with

an initial saturation of $S_0 = 0.2$ and a Dirichlet boundary condition $S_D = 0.6$. The training is performed on a single GPU using 4,096 collocation (residual) points and 1,000 initial/boundary condition points.

Each tested architecture consists of 4 hidden layers with 128 neurons per layer, using tanh activation. For optimization, we apply a learning rate warm-up starting from 10^{-3} , followed by an exponential decay every 2,000 steps with a decay rate of 0.9.

Impact of each feature. Tables 7.2 and 7.3 summarize the results obtained using the Adam and SOAP optimizers, respectively. The best performance was observed when all features were enabled, with SOAP yielding the lowest relative L^2 -error.

Table 7.2 Ablation study of design choices on model performance and runtime for the MLP architecture over 50k iterations using Adam optimiser. Abbreviations: F.F. = Fourier Features, W.F. = Weight Factorization, G.N. = Gradient Norm, T.C. = Time Causality, T.M. = Time Marching, L_2 = Relative Error, B.M. = Best Model (iteration).

F.F.	W.F.	G.N.	T.C.	T.M.	L_2	Loss	CPU (min)	B.M.
✗	✗	✗	✗	✗	9.35×10^{-2}	1.77×10^{-3}	3.6	20k
✓	✓	✗	✗	✗	5.33×10^{-2}	4.49×10^{-4}	3.78	40k
✓	✗	✗	✗	✗	5.06×10^{-2}	7.88×10^{-4}	4.3	25k
✗	✗	✗	✓	✗	6.11×10^{-2}	1.71×10^{-3}	3.7	25k
✓	✓	✗	✓	✗	8.94×10^{-2}	1.24×10^{-2}	4.9	25k
✗	✗	✓	✗	✗	2.84×10^{-2}	1.42×10^{-3}	4.6	25k
✗	✗	✗	✗	✓	3.25×10^{-2}	1.44×10^{-4}	15.7	45k
✓	✓	✓	✓	✓	1.46×10^{-2}	8.6×10^{-4}	25.7	45k

Impact of the model architecture. Next, we analyze the impact of the network architecture on the performance of the PINN model for the BCJ-R equation. We consider three architectures: the standard MLP, the modified MLP (Wang *et al.*, 2021a), and the PirateNet model (Wang *et al.*, 2024a). All features are enabled for each model, and the SOAP optimizer is applied consistently across experiments. The same experimental setup as in the previous ablation studies is applied. The results are illustrated in Figure 7.6. Among the three architectures, the modified MLP yields the lowest L_2 error, indicating the highest accuracy. However, the standard MLP offers a better balance between accuracy and computational cost. In this study, achieving higher accuracy is prioritized in order to capture fine-scale details in the saturation profile of the BCJ-R equation. The higher computational cost associated with the modified MLP can be mitigated through appropriate parallel training strategies.

Parallel training. We analyze the data-parallel strategy adopted for training the PINN model. In this parallel approach, as new devices (GPUs) are added, a copy of

Table 7.3 Ablation study of design choices on model performance and runtime for the MLP architecture over 50k iterations using SOAP optimiser.

F.F.	W.F.	G.N.	T.C.	T.M.	L_2	Loss	CPU B.M.	(min)
✗	✗	✗	✗	✗	5.81×10^{-2}	2.28×10^{-4}	5.6	45k
✓	✓	✗	✗	✗	4.97×10^{-2}	3.9×10^{-5}	8.5	35k
✓	✗	✗	✗	✗	5.35×10^{-2}	2.2×10^{-4}	8.12	45k
✗	✗	✗	✓	✗	5.33×10^{-2}	2.15×10^{-3}	6.71	50k
✓	✓	✗	✓	✗	4.77×10^{-2}	7.79×10^{-4}	8.71	35k
✗	✗	✓	✗	✗	2.5×10^{-2}	3.2×10^{-4}	6.96	50k
✗	✗	✗	✗	✓	1.96×10^{-2}	7.71×10^{-4}	20.1	25k
✓	✓	✓	✓	✓	9.76×10^{-3}	1.6×10^{-4}	35.7	50k

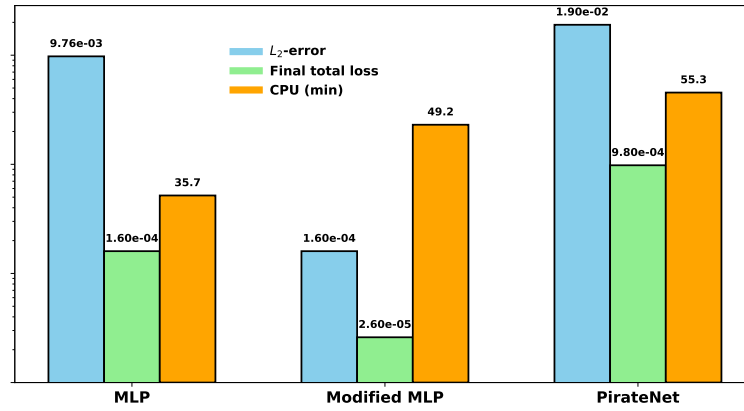


Figure 7.6 Comparison of PINN model performance using different architectures (MLP, Modified MLP (Wang *et al.*, 2021a), and PirateNet (Wang *et al.*, 2024a)) for the BCJ-R equation. Metrics shown include the L_2 -error, total loss, and CPU time, with all features enabled and the SOAP optimizer used across all experiments.

the PINN model is placed on each device, each with its own set of collocation points. During the forward pass, each device computes its local loss independently. Then, during the backward pass, the gradients from all devices are averaged (synchronized), and the resulting global gradient is used to update the shared model parameters. These updated parameters are then broadcast back to all devices, continuing the training loop. We evaluate the weak and strong scaling behavior of this parallel training strategy. Weak scaling refers to increasing the number of GPUs while proportionally increasing the total number of collocation points. In our case, we assign 4096 collocation points per GPU. This helps evaluate how well the training time remains constant as the

problem size and compute resources grow together. On the other hand, strong scaling keeps the total number of collocation points fixed while increasing the number of GPUs, assessing how efficiently the workload is distributed to reduce computation time. For weak scaling, the problem size increases proportionally with the number of GPUs, whereas for strong scaling, the problem size remains fixed.

The weak and strong scaling efficiencies are computed as follows:

$$e_{\text{weak}}^n = \frac{t_1}{t_n}, \quad \text{and} \quad e_{\text{strong}}^n = \frac{t_1}{n t_n}, \quad (7.44)$$

where t_n denotes the time required to complete the task using n GPUs.

The results are shown in Figure 7.7. Both weak and strong scaling behave well up to four GPUs. Weak scaling efficiency stays close to ideal for up to three GPUs, indicating that the increasing data size per GPU is handled effectively. A slight drop at four GPUs suggests growing communication overhead or load imbalance. For strong scaling, efficiency gradually declines as more GPUs are added, which is expected due to fixed workload size and the increasing impact of synchronization and communication costs.

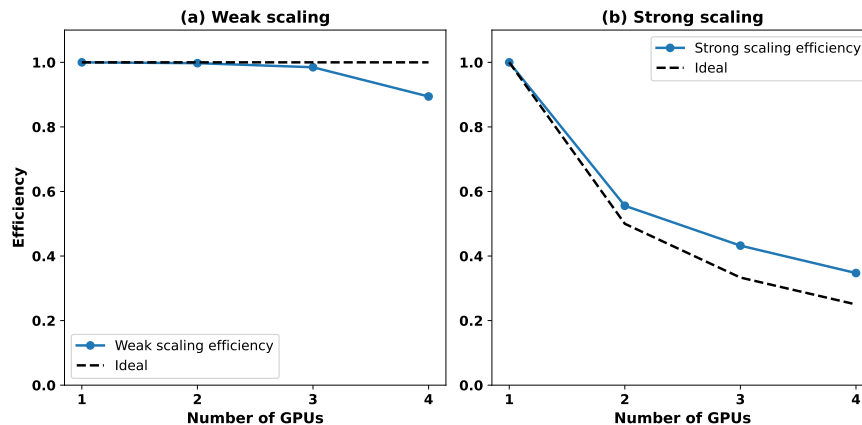


Figure 7.7 Weak and strong scaling efficiencies of the data-parallel training strategy. We use an MLP architecture with all features enabled and measure the execution time over 10,000 Adam iterations. For weak scaling, the number of residual points increases proportionally with the number of GPUs ($n \times 4096$), while for strong scaling, the total number of residual points is fixed at 8192.

CHAPTER 8 Physics-informed neural operators for efficient modeling of infiltration in porous media

Abstract

The development of efficient irrigation systems relies on accurately solving the water flow model described by the highly nonlinear Richards equation. Many forward simulations are required to evaluate various settings, such as irrigation duration, emitter flux rate, initial soil moisture distribution, and the type of irrigation system employed. This often leads to slow and cumbersome decision-making processes. In this study, we propose an efficient and accurate deep learning solver for modeling water flow in unsaturated soils. The model is a physics-informed deep operator network (DeepONet) with an improved architecture designed to accurately solve the highly nonlinear Richards equation. The aim is to train the DeepONet model to learn the mapping between different initial soil moisture conditions and emitter fluxes to the corresponding water content profile in the time-space domain. By incorporating physical constraints, we reduce the need for large amounts of labeled data while maintaining accuracy. Once trained, the DeepONet model provides soil moisture predictions within fractions of a second at any desired time or space location and with any desired resolution, given any initial conditions and fluxes.

The reliability of the DeepONet model is evaluated through numerical experiments on two-dimensional soil geometries with different soil types. The results highlight the efficiency of DeepONet predictions, as they closely match reference solutions. Additionally, the model's extrapolation capability was evaluated, showing that fine-tuning with physical constraints or available soil moisture data improved its prediction accuracy. This study represents a step towards developing an efficient and rapid physics-informed surrogate model for quick decision-making in irrigation strategies.

8.1 Introduction

The rapid increase in population growth worldwide necessitates continuing research for more efficient use of natural resources. Water is the central element of life for all creatures on Earth. Water scarcity, droughts, environmental pollution, and other factors reinforce the need for careful attention to the efficient management of water resources, especially in the era of climate change. In this study, we focus on the unsaturated zone, which lies between the soil surface and the water table. Modeling water dynamics in the unsaturated zone is the cornerstone for many applications. For instance, one can analyze water distribution and root water uptake under different irrigation strategies (Karandish and Šimůnek, 2019). In general, water infiltration in the unsaturated zone is a complex phenomenon involving various processes. Water moves within soil either by gravity forces or capillary action. Water may leave the soil domain through evaporation

from the soil surface, uptake by plants, or infiltration into groundwater (deep percolation). The primary approach for modeling water infiltration mechanisms involves using physical laws encoded in partial differential equations (PDEs). Several mathematical models have been proposed in the literature for modeling water infiltration through soils. A commonly used model for describing water flow in unsaturated soils is the Richards equation ([Richards, 1931](#)), which integrates Darcy’s law with the continuity equation to account for capillary and gravity forces. Numerous studies have sought to extend the Richards equation and develop continuum models for unsaturated flow that better capture hydrodynamic instabilities and fingering patterns observed during soil infiltration (for an overview, see, e.g., ([Eliassi and Glass, 2002](#); [Cueto-Felgueroso and Juanes, 2009](#); [Xiong, 2014](#); [DiCarlo, 2013](#))). Recently, Beljadid, Cueto-Felgueroso, and Juanes proposed a continuum model for unstable infiltration in porous media, endowed with an entropy function ([Beljadid *et al.*, 2020](#)). In this study, we adopt the Richards equation as the governing model for water infiltration.

The Richards equation is a widely used PDE for modeling water flow in soils. This equation is particularly challenging due to the interdependence of various variables. The main variable can be the pressure head, water content, or unsaturated hydraulic conductivity, each of which depends nonlinearly on the others. The dependency is characterized by water retention curves and hydraulic conductivity functions ([Van Genuchten, 1980](#)). These functions are highly nonlinear, leading to solutions with steep gradients at infiltration fronts. Therefore, to solve the Richards equation efficiently, algorithms designed to handle its nonlinear nature should be employed.

Various approaches have been proposed to solve the Richards equation. Numerical techniques include the finite element methods ([Celia *et al.*, 1990b](#); [Keita *et al.*, 2021c](#); [Toutlini *et al.*, 2025a](#); [Kamil *et al.*, 2024b](#)), finite volume method ([Manzini and Ferraris, 2004](#)), finite difference method ([Chávez-Negrete *et al.*, 2018](#)), and meshless techniques ([Boujoudar *et al.*, 2023b](#)). Numerical methods involve discretizing the computational domain into small elements and approximating the solution at the vertices or the center of these elements. The time is typically handled using finite difference methods with specialized numerical schemes to approximate the spatial operator. To solve the nonlinear problem arising from time and space discretizations, linearization techniques must be employed. For comprehensive details on the numerical solutions of the Richards equation, we refer to the literature review ([Farthing and Ogden, 2017](#)).

Another promising area is deep learning algorithms (DLs), where a neural network serves as a parametric model to represent the solution of a given PDE. DLs can operate either in a purely data-driven manner, be constrained by physics, or follow a hybrid approach ([Raissi *et al.*, 2019](#)). The purely data-driven approach is data-hungry, generally requiring a vast amount of labeled data. However, obtaining such extensive data on hydrological variables is often impractical due to its scarcity and the high costs associated with laboratory data collection in terms of time, resources, and personnel. In contrast, physics-constrained deep learning methods require only initial and bound-

ary conditions at a limited number of points, making them well-suited for practical applications. Soil moisture and/or pressure head data are required only when solving inverse problems.

Physics-informed DLs, also known as physics-informed neural networks (PINNs), have demonstrated significant applications in subsurface modeling. For instance, in studies such as (Bandai and Ghezzehei, 2021, 2022), researchers have applied PINNs for inverse modeling of water infiltration in both homogeneous and layered soils. The focus was on identifying water retention curves, hydraulic conductivity functions, and unknown soil surface fluxes, where PINNs have demonstrated efficiency in solving these inverse problems. In (Haruzi and Moreno, 2023a; Wang *et al.*, 2023b), researchers coupled the Richards equation with solute or heat transport for multiphysics modeling within an inverse scenario where PINNs again demonstrated effectiveness. In our previous study (Kamil *et al.*, 2024d), we proposed an improved PINN approach enhanced by transfer learning to model coupled water flow and multiple solute dynamics in unsaturated soils. This method yielded excellent results for this complex multiphysics problem. Other studies have proposed different approaches for solving either inverse problems (Depina *et al.*, 2022; Lan *et al.*, 2024b; Kamil *et al.*, 2024e) or conducting forward simulations using PINNs (Chen *et al.*, 2023b). This makes PINNs a useful alternative to standard methods for solving the Richards equation in both forward and inverse problems. PINNs offer several advantages over standard numerical solvers. One advantage is that PINNs are meshless techniques, making them highly flexible for complex geometries and high-dimensional problems when appropriate training scheme is implemented. Another advantage of PINNs over numerical techniques is their precise computation of temporal and spatial derivatives using automatic differentiation techniques (Baydin *et al.*, 2018), which are implemented in all major deep learning libraries (e.g., JAX, TensorFlow, PyTorch). This approach is more efficient than discretization methods, which can introduce additional numerical artifacts within the soil domain (Celia *et al.*, 1990b). Additionally, PINNs can solve the inverse problem in the same manner as the forward problem by adding a term to the loss function to incorporate available data. Both numerical techniques and physics-informed DLs are computationally expensive and task-specific. This means they are typically focused on a specific version of the addressed PDE, requiring rerunning the numerical solver or retraining the deep learning algorithm for each change in the PDE parameters, which is a common necessity due to varying conditions in practical applications. In the context of designing efficient irrigation systems, multiple simulations are essential to calibrate and optimize various parameters such as soil hydraulic parameters, irrigation duration, emitter flux rate, and others. Adjusting these parameters allows practitioners to evaluate their specific impacts on soil water movement in the root zone. Therefore, there is a strong demand for developing accurate and fast surrogate solvers capable of addressing the multi-task problem of water flow in unsaturated soils.

To address the issue of task-specific solvers in DL algorithms and numerical methods,

deep operator learning (Lu *et al.*, 2021a) and transfer learning (Chen *et al.*, 2021a) have emerged as efficient approaches. Our focus in this study is on the operator learning approach. The core concept involves establishing a mapping between the parameter space of a PDE and its corresponding solutions. The offline stage comprises training the deep operator model across various scenarios, which is subsequently leveraged in the online stage to rapidly predict solutions for new scenarios in a fraction of a second. Various architectures have been developed to implement this idea, including deep operator networks (DeepONets) (Lu *et al.*, 2021a), Fourier neural operators (FNO) (Li *et al.*, 2020), multiple-input DeepONet (Jin *et al.*, 2022), and other models (He *et al.*, 2024; Patel *et al.*, 2024). Deep operator learning approaches have found successful applications in many engineering problems (Cai *et al.*, 2021b; Goswami *et al.*, 2022b). However, in the context of subsurface and groundwater flow modeling, applications are relatively limited. For instance, in (Taccari *et al.*, 2024), researchers proposed a data-driven convolutional neural network based on DeepONet to map the hydraulic conductivity field profile to the hydraulic head in confined, heterogeneous aquifers. This emulator demonstrated high accuracy and achieved rapid predictions within fractions of a second for various scenarios. Similarly, in (Diab *et al.*, 2024), a physics-informed DeepONet was introduced as a fast and accurate simulator for mapping the flux function to solutions within the Buckley-Leverett PDE. More recently, Huang *et al.* (Huang *et al.*, 2024) proposed a CNN-based DeepONet for learning solution operators of parametric reactive transport equations in porous media. As of the current writing, there has been no development of a fast and multitask emulator, such as DeepONet, for the Richards equation.

A key limitation of purely data-driven DeepONet is its need for large training datasets, which are often impractical to obtain in subsurface hydrology due to the challenges in measuring state variables. Similar motivations in PINNs have inspired the development of a physics-informed variant of the DeepONet model (Wang *et al.*, 2021b). In this approach, the DeepONet is trained in a self-supervised manner, requiring only initial and boundary conditions, thereby eliminating the need for extensive labeled training data. Incorporating physical constraints into the loss function of the DeepONet model enhances its generalization by restricting the solution space to functions that satisfy the governing physical laws, while also mitigating overfitting. Therefore, in this study, we employ a physics-informed DeepONet to address the data scarcity typically encountered in subsurface modeling. When the task involves inverse problems or extrapolation, available soil moisture or matric head data can be integrated to further improve convergence.

Another challenge for machine learning models is extrapolation, where the test data may lie outside the distribution of the training data. To improve the predictive accuracy of a pre-trained DeepONet in such scenarios, fine-tuning with physical constraints or available observational data can be employed (Zhu *et al.*, 2023). This approach allows the model to adapt and enhance its performance beyond the original training domain.

The main objective of the present study is to develop an efficient deep learning solver for modeling water flow in unsaturated soils. The Richards equation serves as the PDE for modeling water infiltration, and a physics-informed DeepONet with a modified architecture (Wang *et al.*, 2022b) is proposed to enhance the efficiency of solving this highly nonlinear equation. Various scenarios were tested, including mapping from initial soil moisture and emitter flux to corresponding pressure head profiles. These scenarios are crucial for evaluating different irrigation strategies and understanding the impact of initial soil moisture and emitter discharge on water distribution. The accuracy of DeepONet is assessed against a finite element method (Celia *et al.*, 1990b) as a reference solution. We studied three soil types and two geometries to evaluate the sensitivity and accuracy of the DeepONet model. The fine-tuning strategy is adopted in cases involving extrapolation using either physics or available data. The primary contributions of this study are as follows:

1. The development of an efficient deep learning solver for modeling water flow in unsaturated soils based on physics-informed deep operators; and
2. Enhancement of the model's extrapolation capabilities through fine-tuning with physical constraints or available soil moisture data.

The paper is structured as follows: Section 8.2 presents the governing model of water flow in unsaturated soils along with the constitutive relationships. In Section 8.3, the proposed DeepONet is outlined. The main results are presented in Section 8.4. Finally, the limitations and the main conclusions of the study are discussed in Sections 8.5 and 8.6.

8.2 The mathematical model

The modeling of water flow in unsaturated soils commonly employs the Richards equation (Richards, 1931), which results from integrating Darcy's law with the continuity equation. It is represented as:

$$\begin{cases} \frac{\partial \theta}{\partial t} + \nabla \cdot \mathbf{q} = 0 & \text{in } (0, T] \times \Omega, \\ \mathbf{q} = -K \nabla (\Psi + z) & \text{in } (0, T] \times \Omega. \end{cases} \quad (8.1)$$

In system (8.1), θ indicates the volumetric water content, which we also refer to as soil moisture, Ψ denotes the pressure head, z represents the vertical dimension (gravity axis) directed upwards, $\mathbf{q}$ stands for the water flux, and $K = K_s k_r$ represents the unsaturated hydraulic conductivity, where K_s signifies the saturated hydraulic conductivity and k_r denotes the water relative permeability. The time coordinate is represented by t and spatial coordinates are denoted by $\mathbf{x} = (x, z)$, where x denotes the horizontal position. Additionally, $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$, represents the soil, and the time interval is denoted by $I = (0, T]$, with $T > 0$ being the final simulation time.

The Richards equation (8.1) is subjected to the following initial and boundary conditions:

$$\begin{cases} \theta = \theta_0 & \text{at } t = 0, \\ \theta = \theta_D & \text{in } I \times \Gamma_D \subset I \times \partial\Omega, \\ \mathbf{q} \cdot \mathbf{n} = q_0 & \text{in } I \times \Gamma_N \subset I \times \partial\Omega, \end{cases} \quad (8.2)$$

where θ_0 , θ_D , and q_0 denote the initial soil moisture, Dirichlet boundary condition, and the water inlet as a Neumann boundary condition, respectively. Additionally, $\partial\Omega$ represents the boundary of Ω , which is partitioned into the Dirichlet boundary denoted by Γ_D , and the Neumann boundary denoted by Γ_N . The outward unit normal vector to the boundary Γ_N is denoted by $\mathbf{n}$. The water inlet q_0 is prescribed to be negative in scenarios of water infiltration due to rainfall or irrigation, strictly positive in evaporation cases, and zero where no flux or evaporation occurs at the soil surface. It should be noted that one can define the initial and Dirichlet conditions using the pressure head Ψ as a state variable. In this study, the following terms will be used to describe the following boundary conditions:

$$\begin{cases} \text{No flux or no evaporation boundary condition: } q_0 = 0 & \text{in } I \times \Gamma_N, \\ \text{Free drainage boundary condition: } q_0 = -K & \text{in } I \times \Gamma_N. \end{cases} \quad (8.3)$$

The free drainage boundary condition describes water infiltration downward due to gravitational forces, allowing water to exit the soil domain. It is typically applied at the bottom boundary of the soil domain when there is no groundwater nearby.

Remark 9 *In this study, the boundary $\partial\Omega$ consists solely of the Neumann boundary Γ_N , which comprises four distinct boundaries: the soil surface, the soil bottom, and the two lateral boundaries. Further details regarding these boundary conditions will be provided in the following sections.*

To complete the description of the Richards equation (8.1), explicit functions relating water content θ and unsaturated hydraulic conductivity K in terms of pressure head Ψ must be provided. Various constitutive relationships can be employed for this purpose. These functions are highly nonlinear for realistic soils and can be provided either by a closed-form model or as tabular data. In this study, we utilize the van Genuchten model (Van Genuchten, 1980) to characterize θ and the van Genuchten-Mualem model (Mualem, 1976) to describe K . The function θ represents water retention curves, while K denotes hydraulic conductivity functions. The complete model is presented as follows:

$$\theta = \begin{cases} \theta_r + (\theta_s - \theta_r) \left(1 + (\alpha_v |\Psi|)^{n_v} \right)^{-m_v} & \text{if } \Psi \leq 0, \\ \theta_s & \text{if } \Psi > 0, \end{cases} \quad (8.4a)$$

$$K = \begin{cases} K_s \frac{\left[1 - (\alpha_v |\Psi|)^{n_v-1} \left(1 + (\alpha_v |\Psi|)^{n_v}\right)^{-m_v}\right]^2}{\left(1 + (\alpha_v |\Psi|)^{n_v}\right)^{m_v/2}} & \text{if } \Psi \leq 0, \\ K_s & \text{if } \Psi > 0, \end{cases} \quad (8.4b)$$

$$m_v = 1 - \frac{1}{n_v}, \quad n_v > 1, \quad (8.4c)$$

where θ_s and θ_r represent the saturated and residual volumetric water contents, respectively, and n_v , m_v and α_v are the van Genuchten empirical parameters.

In this study, we investigate three soil types: loam, sandy loam, and sand. These soils differ primarily in their texture, structure, and ability to retain water. Loam has a balanced mix of sand, silt, and clay, offering good water retention and moderate drainage. Sandy loam has a higher sand content, leading to faster drainage and lower water-holding capacity. Sand, being coarser, exhibits the least water retention and promotes rapid infiltration and drainage, often resulting in rapid water infiltration and sharp wetting fronts.

Remark 10 *The van Genuchten parameters for each soil type generally need to be calibrated using experimental data, which is often a critical task. This calibration is commonly addressed through inverse problems using available soil moisture data or alternative techniques such as hierarchical pedotransfer functions (Schaap et al., 2001). In our study, we utilized the HYDRUS database for the parameterization of these soil types.*

The complexity of the Richards equation increases notably for very coarse soils (e.g., sandy soils) compared to fine-textured soils. This complexity arises from the dominance of gravitational forces over capillary forces, resulting in an advection-dominated problem. The pore-size distribution parameter α_v can be used to measure the dominance of gravitational forces over capillary forces. Sandy soils have the highest α_v values compared to loam and sandy loam soils. The constitutive relationships for each soil are plotted in Figure 8.1, where sharper constitutive relationships are observed for sandy soil compared to other soils. The parameters used to generate Figure 8.1 are defined in Table 8.1. The water retention curves depicted in Figure 8.1 highlight the intricacy of the Richards equation. They exhibit highly nonlinear behavior, particularly near the saturation regions, where minor adjustments in water saturation levels can result in significant changes in corresponding pressure heads. These characteristics contribute to the potential degeneracy of Richards equation solutions, complicating accurate numerical simulations. For a more detailed theoretical discussion on the degeneracy of the solution of the Richards equation with respect to soil parameters, we refer to (Paniconi et al., 1991).

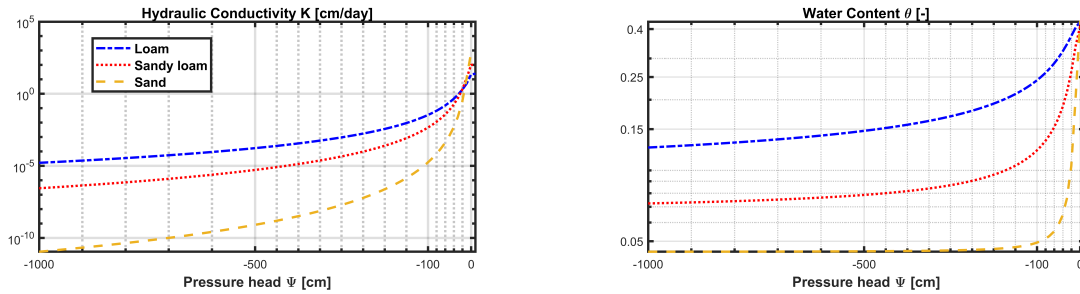


Figure 8.1 Constitutive relationships for three soil types (loam, sandy loam, and sandy soil) generated using the Mualem-van Genuchten model, displayed on a log-log scale. The left panel shows the unsaturated hydraulic conductivity (K) as a function of pressure head (Ψ), while the right panel presents the water content (θ) as a function of pressure head (Ψ). Soil parameters used in the model are provided in Table 8.1.

Table 8.1 Parameters of three types of soil used for the Mualem-van Genuchten model.

Parameter	Loam	Sandy loam	Sand
θ_r [$\text{m}^3 \text{m}^{-3}$]	0.078	0.065	0.045
θ_s [$\text{m}^3 \text{m}^{-3}$]	0.43	0.41	0.43
α_v [cm^{-1}]	0.036	0.075	0.145
n_v [-]	1.56	1.89	2.68
K_s [cm day^{-1}]	24.96	106.1	712.8

8.3 Methods

In this section, we introduce the proposed deep operator framework for the Richards equation. A brief introduction to deep operator learning and the DeepONet architecture is provided in Appendix 8.6.

8.3.1 DeepONet for the Richards equation

8.3.1.1 Methodology

In this study, we aim to develop an efficient DeepONet model for solving water flow problems in unsaturated soils with different parameter settings. The parameter space $\mathcal{P}$ will be varied to explore different scenarios relevant to real-world problems. We consider two specific scenarios:

- Mapping initial soil moisture conditions to the corresponding pressure head profiles; and
- Mapping water inlet boundary conditions to the corresponding pressure head profiles.

These scenarios have significant implications for various real-world applications. For instance, scenario (a) is crucial for assessing the impact of initial soil moisture on the infiltration and redistribution of water and solutes in unsaturated soils. This assessment is essential for understanding the dynamics of water and nutrient availability for plant growth and environmental processes. On the other hand, scenario (b) is particularly relevant for evaluating the effect of emitter discharge on the infiltration and redistribution of water in irrigation systems. This evaluation plays a vital role in designing efficient irrigation strategies, especially in semiarid and arid regions where water scarcity is a major concern. By understanding the relationship between water inlet boundary conditions and the resulting pressure head profiles, irrigation practitioners can optimize emitter placement, discharge rates, and scheduling to maximize water use efficiency and minimize deep percolation losses. What makes DeepONet efficient compared to standard numerical solvers is that once it is trained on a specific scenario, it can be used to make various predictions in a fraction of a second for unseen initial conditions or emitter fluxes. The only requirement is that the test data must lie within the support of the training data, meaning it must fall within the range of scenarios covered during training (i.e., interpolation). We will also test the extrapolation capabilities of DeepONet, discussing scenarios in which the pre-trained DeepONet should be retrained using a fine-tuning strategy (Zhu *et al.*, 2023).

To assess the sensitivity of DeepONet, we use three soil types, as detailed in Table 8.1: loam, sandy loam, and sandy soils. The complexity increases from loam to sand as gravity forces become more dominant over capillary forces. Additionally, two geometries are employed, as shown in Figure 8.2. In the first geometry, we model a scenario with a long duration of rainfall, where water infiltrates the soil domain uniformly over the entire surface. In the second geometry, the soil surface is divided into two portions: the left portion for water irrigation and the right portion for atmospheric conditions where we assume no evaporation. This scenario reflects a setting for surface drip irrigation where the dripper is located in a small portion of the soil surface. For both soil geometries, the lateral boundaries are assumed to have no flux ($q_0 = 0$), and the bottom boundary is set to free drainage ($q_0 = -K$). In the first scenario (a), we fixed the water inlet q_0 for the portion of water infiltration in both geometries to -5 cm/day. In scenario (b), the water inlet q_0 will vary, while the initial soil moisture will be fixed at $\theta_0 = 0.20$. Detailed descriptions of the two studied scenarios is provided in the results section.

Remark 11 *For scenario (b), the soil surface flux refers to the value of q_0 at the location of the water inlet zone. This zone encompasses the entire soil surface for geometry 1, whereas for geometry 2, it only includes the left portion of length r_0 . Throughout all the papers, we use the terms “water fluxes” and “water inlet conditions” interchangeably to maintain consistency.*

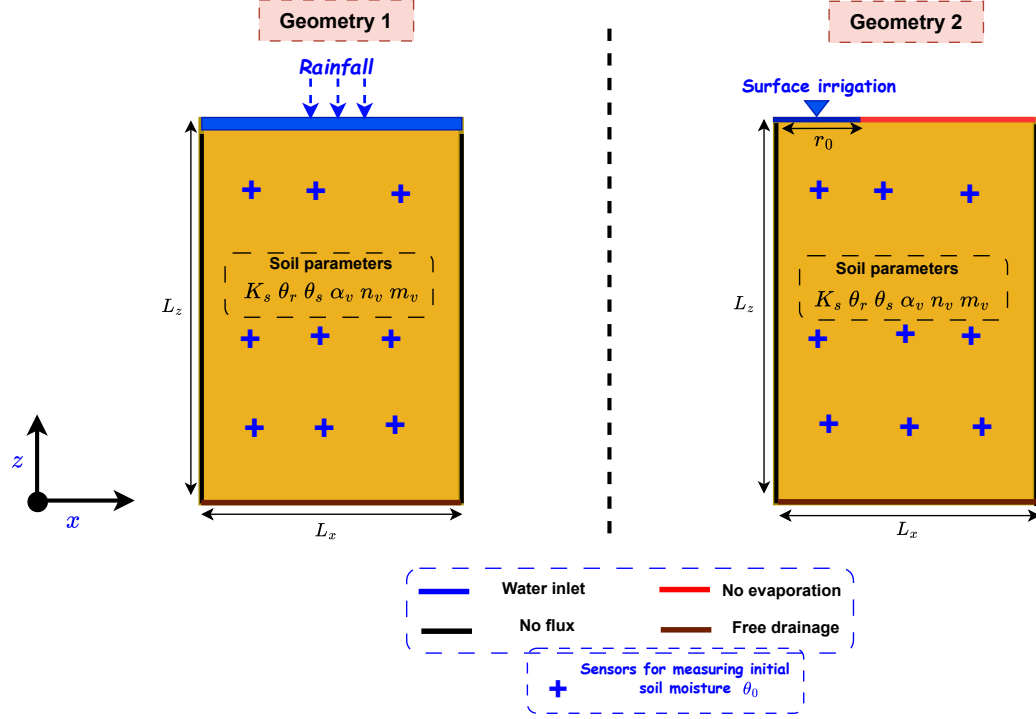


Figure 8.2 Two-dimensional soil geometry configurations used in this study. The parameters are: $L_x = L_z = 1$ m for geometry 1, and $L_x = L_z = 0.5$ m, $r_0 = 5$ cm for geometry 2. The final simulation times are $T = 24$ hours for geometry 1 and $T = 6$ hours for geometry 2. The soil geometries are represented as rectangular domains, with boundary types indicated using specific colors.

8.3.1.2 DeepONet architecture

In the following, we explore the architecture of the adopted DeepONet used to model water infiltration for both mapping scenarios: initial conditions and water inlet to the corresponding pressure head solution.

Let the training data be the given of N_{samples} in the space of parameters $\mathcal{P}$: $\{u^{(i)}\}_{i=1}^{N_{\text{samples}}}$, each evaluated in a given set of m sensors $(s_j)_{j=1}^m$. In our case, $u^{(i)}$ represents either the initial soil moisture $\theta_0^{(i)}$ or a water inlet $q_0^{(i)}$. The sensors for the initial soil moisture $\theta_0^{(i)}$ are the space coordinates $s_j = (x_j, z_j) \in \Omega$, and for the water inlet $q_0^{(i)}$ are the time coordinates $s_j = t_j \in (0, T]$. We note that the sensors are not forced to be equi-spaced, but they are the same for all samples $u^{(i)}$.

In addition, we need time-space collocation points as inputs for the trunk network $\{\mathbf{y}_j^{(i)} = (t_j^{(i)}, x_j^{(i)}, z_j^{(i)})\}_{j=1}^P$. Here we focus on notation, where it is important that the data $\{\mathbf{y}_j^{(i)}\}_{j=1}^P$ are in general different across the samples $\{u^{(i)}\}_{i=1}^{N_{\text{samples}}}$; in other words, each $u^{(i)}$ has its own collocation points $\{\mathbf{y}_j^{(i)}\}_{j=1}^P$ for all $i = 1, \dots, N_{\text{samples}}$.

The choice of architecture for both the branch and trunk networks is crucial and de-

depends upon the specific problem at hand. In the original DeepONet paper (Lu *et al.*, 2021a), the authors utilized a standard multilayer perceptron (MLP). In (Wang *et al.*, 2022b), a modified version of MLP was adopted to enhance training efficiency and model accuracy, inspired by the attention mechanism adopted in transformers (Vaswani *et al.*, 2017). In this study, we use the modified DeepONet architecture proposed by (Wang *et al.*, 2022b). The modified version is explained as follows: The inputs of the branch u and the trunk $\mathbf{y}$ are first encoded in a high-dimensional latent space by employing two distinct encoders, U_B and U_T , given by:

$$U_B = \tanh(W_u u + b_u), \quad U_T = \tanh(W_y \mathbf{y} + b_y), \quad (8.5)$$

where $\Theta_0 = (W_u, b_u, W_y, b_y)$ are the trainable parameters of the two encoders. The outputs of the two encoders are then combined with each hidden layer of an MLP using point-wise multiplication to compute the forward pass. Algorithm 7 provides the forward pass of the DeepONet model.

Algorithm 7 Modified DeepONet architecture

Inputs: $B_0 = u$, $T_0 = \mathbf{y}$, two encoders U_B and U_T defined by equation (8.5).
for each hidden layer $l = 1, \dots, L$ **do**
 First transformation:
 $B_l^* = \tanh(W_u^l B_{l-1} + b_u^l)$,
 $T_l^* = \tanh(W_y^l T_{l-1} + b_y^l)$,
 Hidden layer output:
 $B_l = \tanh((1 - B_l^*) \odot U_B + B_l^* \odot U_T)$,
 $T_l = \tanh((1 - T_l^*) \odot U_B + T_l^* \odot U_T)$,
end for
Final output: $G_{\Theta}(u)(\mathbf{y}) = \sigma(B_L \cdot T_L + c_0)$,
Note: Notation $\cdot$ denotes the inner product, and $\odot$ denotes the point-wise product.

In Algorithm 7, the inputs to the branch and trunk networks are represented by B_0 and T_0 , respectively. B_l^* and T_l^* denote the l -th hidden layer of the branch and trunk networks before combining with the output of the two encoders, and B_l and T_l denote the l -th hidden layer of the branch and trunk networks, respectively, with L being the total number of layers. The trainable parameters of the branch and the trunk networks are $\Theta_B = (W_u^l, b_u^l)_{l=1}^L$ and $\Theta_T = (W_y^l, b_y^l)_{l=1}^L$, respectively. Thus, the overall trainable parameters of the modified DeepONet model are collected in $\Theta = (\Theta_0, \Theta_B, \Theta_T, c_0)$. The flowchart of the DeepONet for modeling water flow in unsaturated soils is illustrated in Figure 8.3. This flowchart outlines various aspects of the model, which will be explained in detail in the following sections.

Remark 12 We adopt $\tanh$ as the activation function for the hidden layers due to

its smoothness. This choice is motivated by the need to compute derivatives—up to second order with respect to spatial coordinates and first order with respect to time, in the Richards equation.

8.3.1.3 DeepONet training phase

Here we describe the training procedure of the DeepONet model. The pressure head variable Ψ is chosen as the output of the DeepONet. Given that the pressure head is negative in unsaturated zones, we adopt $-\exp$ as the output function σ to adhere to this physical constraint. After each forward pass, the following loss function is

computed:

$$\mathcal{L}(\Theta) = \alpha_{\text{data}} \mathcal{L}_{\text{data}}(\Theta) + \alpha_{\text{physics}} \mathcal{L}_{\text{physics}}(\Theta) + \alpha_{\text{IC}} \mathcal{L}_{\text{IC}}(\Theta) + \alpha_{\text{BC}} \mathcal{L}_{\text{BC}}(\Theta), \quad (8.6)$$

where $\mathcal{L}_{\text{data}}$, $\mathcal{L}_{\text{physics}}$, $\mathcal{L}_{\text{IC}}$, and $\mathcal{L}_{\text{BC}}$ represent the losses associated with available data, physical constraints, initial and boundary conditions, respectively, and α_{data} , α_{physics} , α_{IC} , α_{BC} are the corresponding weights. The above losses are defined as follows:

$$\mathcal{L}_{\text{data}}(\Theta) = \frac{1}{N_{\text{samples}} N_{\text{data}}} \sum_{i=1}^{N_{\text{samples}}} \sum_{j=1}^{N_{\text{data}}} \left| \theta(\mathbf{y}_{\text{data}}^{(j)}) - \theta\left(G_{\Theta}(u^{(i)})(\mathbf{y}_{\text{data}}^{(j)})\right) \right|^2, \quad (8.7a)$$

$$\mathcal{L}_{\text{physics}}(\Theta) = \frac{1}{N_{\text{samples}} N_{\text{physics}}} \sum_{i=1}^{N_{\text{samples}}} \sum_{j=1}^{N_{\text{physics}}} \left| \mathcal{R}\left(G_{\Theta}(u^{(i)})(\mathbf{y}_{\text{physics}}^{(j)})\right) \right|^2, \quad (8.7b)$$

$$\mathcal{L}_{\text{IC}}(\Theta) = \frac{1}{N_{\text{samples}} N_{\text{IC}}} \sum_{i=1}^{N_{\text{samples}}} \sum_{j=1}^{N_{\text{IC}}} \left| f^{(i)}(\mathbf{y}_{\text{IC}}^{(j)}) - \theta\left(G_{\Theta}(u^{(i)})(\mathbf{y}_{\text{IC}}^{(j)})\right) \right|^2, \quad (8.7c)$$

$$\mathcal{L}_{\text{BC}}(\Theta) = \frac{1}{N_{\text{samples}} N_{\text{BC}}} \sum_{i=1}^{N_{\text{samples}}} \sum_{j=1}^{N_{\text{BC}}} \left| g^{(i)}(\mathbf{y}_{\text{BC}}^{(j)}) - \mathcal{B}\left(G_{\Theta}(u^{(i)})(\mathbf{y}_{\text{BC}}^{(j)})\right) \right|^2. \quad (8.7d)$$

Here, the output of the DeepONet is transformed to the water content variable using the van Genuchten model (8.4a) to compute the data loss function $\mathcal{L}_{\text{data}}$ (8.7a). However, we can also have data in terms of the pressure head, eliminating the need for an output transformation. While several existing instruments can effectively measure water content and pressure head, devices that measure water content are generally more applicable (Robinson *et al.*, 2008). This is due to the higher correlation of water content with many easily measured signals, such as electrical conductivity, density, and dielectric permittivity (Zha *et al.*, 2013). Therefore, it is advisable to evaluate data loss in terms of water content measurement.

The operator $\mathcal{R}$ in equation (8.7b) defines the Richards equation operator:

$$\mathcal{R}\left(G_{\Theta}(u^{(i)})(\mathbf{y}_{\text{physics}}^{(j)})\right) = \frac{\partial \theta\left(G_{\Theta}(u^{(i)})(\mathbf{y}_{\text{physics}}^{(j)})\right)}{\partial t} - \nabla \cdot \left[K\left(G_{\Theta}(u^{(i)})(\mathbf{y}_{\text{physics}}^{(j)})\right) \nabla \left(G_{\Theta}(u^{(i)})(\mathbf{y}_{\text{physics}}^{(j)}) + z\right) \right]. \quad (8.8)$$

The function $f^{(i)}$ in equation (8.7c) can be either the sample $u^{(i)}$, which represents an initial water content in scenario (a), or the fixed water content θ in scenario (b). On the other hand, the definition of the boundary operator $\mathcal{B}$ in equation (8.7d) depends on the type of boundary used. The function $g^{(i)}$ is either the corresponding imposed condition in scenario (a), or the associated water inlet in scenario (b).

In general, the choice of weights in (8.6) is crucial for training the DeepONet model where four different loss functions should be minimized simultaneously. In this study, a trial-and-error procedure is used to find the best combination. When $\alpha_{\text{data}} = 0$, this is known as a data-free, physics-informed DeepONet, while when $\alpha_{\text{physics}} = \alpha_{\text{IC}} =$

$\alpha_{\text{BC}} = 0$, this represents a purely data-driven DeepONet. We used a data-free physics-informed DeepONet with $\alpha_{\text{physics}} = 1$, while other weights were test-dependent. In the case of integrating available data into the loss function $\mathcal{L}$, we set $\mathcal{L}_{\text{data}} = 1$.

The DeepONet algorithm for modeling water flow in unsaturated soils is presented in Algorithm 8.

Remark 14 *The proposed modified DeepONet model adopts an MLP as its base architecture for two primary reasons (Azizzadenesheli et al., 2024):*

1. **Mesh-independent property:** *After training, the model can predict pressure head and soil moisture profiles at any desired resolution and at arbitrary points in the time-space domain;*
2. **Differentiability:** *The MLP architecture allows for straightforward computation of the derivative of the output with respect to the input. This capability is crucial for enforcing physical constraints in the learning process using automatic differentiation techniques.*

These characteristics make the MLP-based DeepONet particularly suitable for applications where physical constraints and flexible resolution in predictions are important considerations.

Remark 15 *The loss function of the DeepONet, as defined in (8.6), follows a soft-constraint approach. In this framework, the Monte Carlo method is used to approximate the integrals of the PDE, as well as the initial, data, and boundary terms, by projecting them onto a set of sampled points. While this is the primary approach, it is worth noting that other formulations, such as the variational loss function (Goswami et al., 2022b) can also be adopted. When it comes to implementing initial and boundary conditions, a soft implementation is typically used in the DeepONet framework. However, it is important to note that these conditions can also be imposed exactly (Sukumar and Srivastava, 2022). The choice between soft and hard implementation often depends on the specific requirements of the problem. In general, Neumann boundary conditions are used to impose water flux at the soil boundaries. It is worth mentioning that the hard imposition of these kinds of conditions is not as straightforward as implementing Dirichlet boundary terms. For cases where stronger enforcement of constraints is desired, alternative approaches can be considered. One such method is the augmented Lagrangian approach (Basir and Senocak, 2022). This method, as demonstrated in the case of the Richards equation in (Lan et al., 2024a), adds stronger penalties for the initial and boundary conditions, potentially leading to more accurate adherence to these physical constraints during the training process.*

Algorithm 8 DeepONet for mapping from the initial soil moisture/water inlet to the pressure head solution of the Richards equation.

1. Inputs:

- (a) Soil parameters $(K_s, n_v, m_v, \alpha_v, \theta_r, \theta_s)$.
- (b) The boundary conditions, the final time of simulation T , and available water content or pressure head data.
- (c) N_{samples} initial soil moisture samples (scenario (a)) or water inlet samples (scenario (b)) $\{u^{(i)}\}_{i=1}^{N_{\text{samples}}}$.
- (d) P collocation points $\{\mathbf{y}_j^{(i)}\}_{j=1}^P$ for each sample $u^{(i)}$.

2. Initialization:

- (a) Randomly initialize the parameters of the DeepONet (branch, trunk, and the two encoders) using the Xavier algorithm.
- (b) Apply the hyperbolic tangent activation function for hidden layers.
- (c) Set the output activation to $-\exp(x)$.

3. Training: Train the DeepONet model using the Adam optimizer algorithm.

4. Output:

- (a) Predicted pressure head, water content, and water flux.
-

8.3.1.4 Extrapolation of the DeepONet model

In real-world applications, one may encounter initial soil moisture profiles or water inlet conditions that fall outside the range of values used during the training phase of the DeepONet model. For example, in scenario (a), we trained the DeepONet on initial soil moisture conditions that are spatially uniform and evaluated its generalization by feeding it initial soil moisture conditions that vary with space. This situation is known as extrapolation, where the data lies outside the support of the training data. Consequently, the predictions of the DeepONet may not adhere to the physical laws or the initial/boundary conditions. Several studies have tried to propose efficient approaches capable of handling extrapolation scenarios (e.g., [Bhatt *et al.* \(2023\)](#); [Zhu *et al.* \(2023\)](#)). In this study, to address the extrapolation scenario, we opted for fine-tuning the pre-trained DeepONet model using available knowledge, which could include physical laws or some labeled field data. This approach was proposed in ([Zhu *et al.*, 2023](#)), where various strategies were also analyzed.

The fine-tuning process works as follows: for each new dataset u_{new} , we fix the input of the branch network to u_{new} , resample collocation points for the trunk network, and

retrain the pre-trained DeepONet using the same loss function defined in (8.6). In this case, the DeepONet functions as a PINN solver, specifically aimed at solving the Richards equation for one particular parameter u_{new} . It is important to keep in mind that the training of the DeepONet does not start from scratch; instead, the knowledge gained from previous training scenarios is used to initialize the parameters of the new model. Additionally, it is not trivial to measure the extrapolation complexity for each new dataset to determine the necessary fine-tuning time needed to achieve the desired accuracy. This remains an open challenge, as the measures proposed in (Zhu *et al.*, 2023) are not straightforward enough to extend to subsurface hydrological modeling problems. The extrapolation algorithm of the pretrained DeepONet model is illustrated in 9. For a comprehensive overview of the DeepONet surrogate solver's application in modeling water infiltration in unsaturated soils, we again refer to Figure 8.3.

Remark 16 *We note that there are various approaches to fine-tuning a pre-trained DeepONet model. In (Zhu *et al.*, 2023), the authors explored several strategies, including fine-tuning the entire model or selectively tuning the branch or trunk parameters while keeping the other parameters frozen. Moreover, the fine-tuning process can incorporate available physical knowledge or utilize data from various sources. The fine-tuning approach can be broadly categorized into two main types:*

1. *Physics-based fine-tuning: This involves imposing physical constraints, such as the Richards equation residual along initial and boundary conditions; and*
2. *Data-driven fine-tuning: This approach incorporates new soil moisture data to refine the model's predictions.*

In our study, we opted to fine-tune all parameters of the DeepONet model. To ensure physical consistency during extrapolation, we enforce the Richards equation residual alongside initial and boundary conditions. In cases of inverse problems, we also integrate available soil moisture data to further enhance the model's predictive accuracy.

Remark 17 *Fine-tuning is a concept related to transfer learning (Pan and Yang, 2009), aimed at enhancing the generalizability of a machine learning algorithm without starting the training process from scratch. However, transfer learning methods are generally developed to handle covariate shift, label shift, and conditional shift, not specifically tailored for extrapolation tasks (Zhu *et al.*, 2023).*

8.3.1.5 Algorithm implementation

The implementation of the DeepONet was performed using JAX (Bradbury *et al.*, 2018), a modern Python toolkit developed by Google for high-performance machine learning research. JAX offers powerful features such as advanced automatic differentiation and just-in-time compilation (JIT). The parameters of the DeepONet were

Algorithm 9 Extrapolation of the DeepONet model for predicting pressure head and soil moisture 2D profiles.

1. Inputs:

- (a) A pre-trained DeepONet model G_{Θ} .
- (b) Soil parameters $(K_s, n_v, m_v, \alpha_v, \theta_r, \theta_s)$.
- (c) Initial/boundary conditions, and the final simulation time T .
- (e) N_{samples} extrapolation samples of initial soil moisture or water flux $\{u^{(i)}\}_{i=1}^{N_{\text{samples}}}$.
- (f) P collocation points $\{\mathbf{y}_j^{(i)}\}_{j=1}^P$ for each sample $u^{(i)}$.
- (g) Number of epochs for fine-tuning.

2. Fine-Tuning:

- (a) **Physics-based fine-tuning:**
 - i. Incorporate the Richards equation residual as a loss term.
 - ii. Apply initial and boundary conditions.
- (b) **Data-driven fine-tuning** (if applicable):
 - i. Integrate available soil moisture data into the loss function.
 - ii. Adjust the DeepONet model parameters using the Adam optimizer to minimize the combined loss (Richards equation residual + data loss + initial and boundary conditions).

3. Extrapolation:

For each extrapolation sample $u^{(i)}$, where $i = 1, 2, \dots, N_{\text{samples}}$:

- (a) Fix the branch network input to $u^{(i)}$.
- (b) Fine-tune G_{Θ} with physical constraints and any available soil moisture data.
- (c) Predict the corresponding pressure head and soil moisture profiles.
- (d) Store the predicted pressure head and soil moisture profiles for the sample $u^{(i)}$.

4. Output: Predicted pressure head and soil moisture profiles for each extrapolation sample.

initialized using the Xavier algorithm (Glorot and Bengio, 2010a), a technique that prevents the activation function from becoming saturated at the beginning of the training phase. This initialization was applied to the weights, while the biases were initialized to 0. The Adam optimizer (Kingma and Ba, 2014) was employed to minimize the total

loss function $\mathcal{L}$ (8.6). It was configured with an exponential decay starting at 10^{-3} , a decay rate of 0.95, and a decay step of 1000, while all other parameters were set to their default values. This setup was chosen to improve the stability of DeepONet training. Additionally, a mini-batching technique was implemented due to its proven efficiency in handling stiff PDEs in the context of PINNs (McClenny and Braga-Neto, 2023). The training process stopped upon reaching a predefined maximum number of epochs.

The collocation points were uniformly generated within the time-space domain, as illustrated schematically in Figure 8.4. Different colors represent various types of collocation points. Blue points denote residual collocation points generated inside the time-space domain, which are used to enforce the Richards residual term. The other collocation points are dedicated to the initial and boundary conditions. Figure 8.4 also depicts a mesh used for the reference solution in Geometry 2, based on the linear Galerkin finite element method (FEM). This method was chosen for its widespread application in solving the Richards equation. For more details about the implementation and convergence analysis of the FEM scheme for the Richards equation, refer to Chapter 2. The following error measurement was adopted to quantify the discrepancy between the DeepONet prediction and the FEM reference solution:

$$\text{Relative-}L^2 \text{ error : } \left(\frac{\sum_{\mathbf{y}} (\hat{\Psi}(\mathbf{y}) - \Psi(\mathbf{y}))^2}{\sum_{\mathbf{y}} \Psi(\mathbf{y})^2} \times 100 \right) \%, \quad (8.9)$$

where $\hat{\Psi}$ and Ψ denote the predicted and the reference solutions, respectively. The sums are over all the collocation points in the time-space domain $I \times \Omega$. This is for one sample, and it is calculated over all samples $u^{(i)}$, and the average and standard deviation across all samples are reported.

All settings, including the architecture of DeepONet (number of hidden layers and neurons per layer), the number of collocation points P , and the number of samples N_{samples} , depend on each specific test case and will be clarified accordingly.

Remark 18 *After training, we evaluate the DeepONet predictions on test points that are equally spaced in the time-space domain, which are different from the randomly uniform sampled points used during training.*

8.4 Results

In this section, we discuss the performance of the proposed DeepONet model in solving the two scenarios. Numerical experiments were conducted using a single NVIDIA A100 GPU and an NVIDIA V100 Volta GPU. These experiments were hosted on the Graham and Narval clusters, which are part of the Compute Canada clusters. The source code is available on the GitHub repository: <https://github.com/Hamza-Kamil/DeepONet-WaterFlow-Soils>.

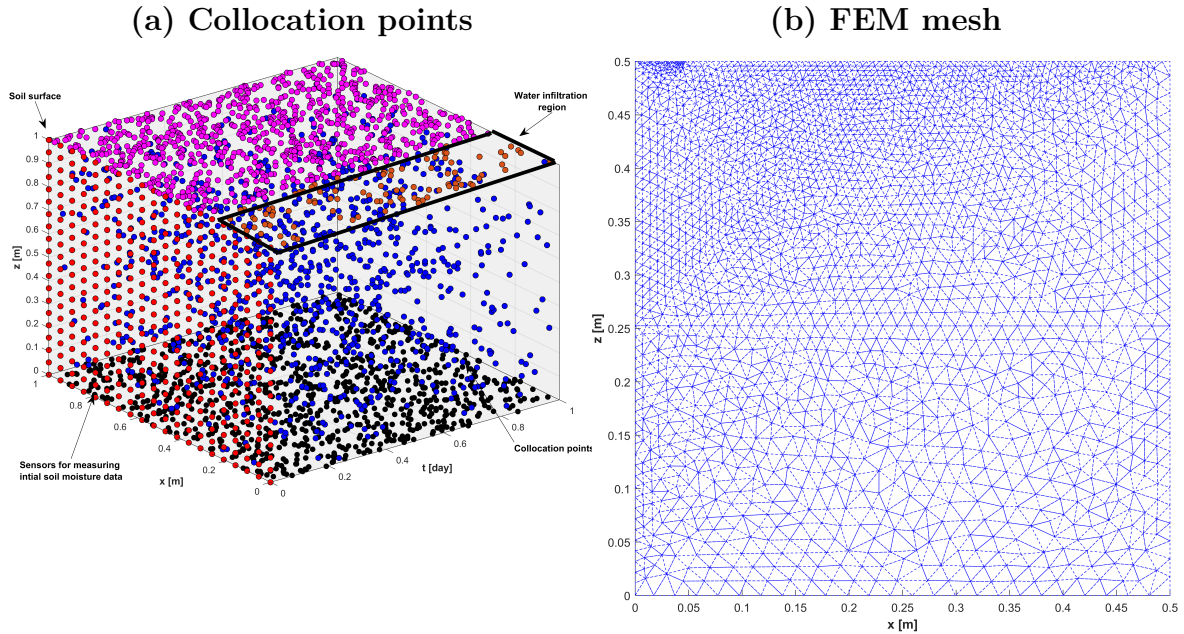


Figure 8.4 (a) Visualization of collocation points for DeepONet. (b) Unstructured mesh employed in the FEM reference solution. Both correspond to Geometry 2.

The structure of the results section is as follows: For each scenario, we first detail the setup of the DeepONet model, including its architectures and the size of the training data. Following this, we delve into the interpolation and extrapolation results, along with additional findings from scenarios involving inverse problems.

8.4.1 Scenario (a): Mapping initial soil moisture to the corresponding pressure head profile

In scenario (a), the objective is to map various initial soil moisture profiles to their corresponding pressure head solutions. Parametrizing this problem is challenging, as generating representative scenarios is complex. Therefore, we explore two types of parametrization. The first type is a non-parametric approach, where we generate the same initial condition in terms of soil moisture repeated N_{samples} times. This means that the input to the Branch network is only one initial condition. In this scenario, we aim to evaluate whether DeepONet can effectively capture the non-parametric situation. In this context, DeepONet is equivalent to the PINN approach. For each soil type, the initial condition is chosen within the interval (θ_r, θ_s) and uniform in space. This numerical test is referred to as scenario (a)-non-parametric initial conditions.

The second parametrization involves generating N_{samples} different initial soil moisture values within the interval (θ_r, θ_s) for each soil type. In this approach, the samples are assumed to be uniform over space, reflecting common practical scenarios when the soil is supposed to be homogeneous. This numerical test is referred to as scenario (a)-parametric initial conditions. In all these parametrizations, we have set the water inlet to $q_0 = -5$ cm/day. A visualization of both parametrizations is presented in Appendix 8.6.

8.4.1.1 Scenario (a)-non-parametric initial conditions

Settings. The configuration and architecture used in this scenario are summarized in Table 8.5. Table 8.2 presents the initial soil moisture values for each soil type and geometry. Different initial conditions were chosen for each geometry to evaluate the model’s performance under varying moisture conditions. Notably, DeepONet requires only sparse measurements of initial soil moisture for training, in contrast to the dense data typically needed by standard numerical solvers.

Interpolation results. Snapshots of the results obtained from the DeepONet and the FEM reference solution are depicted in Figures 8.5 and 8.6. The DeepONet model demonstrates good agreement with the reference solution obtained from FEM for the three soils and the two geometries. It can be observed that the infiltration front in sandy soil is much steeper compared to the other two soils. For sandy soil and geometry 2, we observe a poor prediction of the sharp front by DeepONet, whereas it produces accurate results for other soil types. For more details, we report the average relative error in Table 8.3. The lowest error is observed for loamy soil, followed by sandy loam and sandy soils. This reinforces the conclusion that complexity increases when the scenario involves coarser-textured soils and a more complex boundary condition (geometry 2). Despite the higher average error for sandy soils, DeepONet has an error lower than 4.5% in all cases, demonstrating its reliability in this scenario. We also note that with the FEM scheme, both a small mesh size and a small time step are needed to ensure stable and accurate results, especially for sandy soils and geometry 2.

The reduced performance of the DeepONet model for sandy soil and complex boundary conditions was expected, as DeepONet is essentially a generalization of the vanilla PINN (Raissi *et al.*, 2019), inheriting their limitations. Accurately capturing advection-dominated problems and discontinuous solutions remains a significant challenge for these architectures (Fuks and Tchelepi, 2020). To address this, we implemented a different methodology to improve the accuracy of the DeepONet model, aiming for a relative L^2 error of less than 1%. We first trained the model on loamy soil, then applied a transfer learning strategy by using the learned parameters from the DeepONet solver for loamy soil to initialize the model for sandy soil. This approach increased the accuracy of the DeepONet by 42% (results not shown here), demonstrating the effectiveness of transfer learning, where a simpler task is tackled first, followed by a more difficult one. This is also called curriculum training. A similar methodology was adopted in (Wang *et al.*, 2023a) to solve incompressible Navier–Stokes equations with high Reynolds numbers.

In terms of CPU time, DeepONet’s inference time is 50 times faster than that of the FEM

Table 8.2 Scenario (a) – The initial soil moisture θ_0 [m^3/m^3] for each type of soil and geometry. For scenario (a) – parametric initial conditions, we specify the range (min–max) within which we randomly generate the soil moisture values.

Soil type	Non-parametric initial conditions		Parametric initial conditions	
	Geometry 1	Geometry 2	Geometry 1	Geometry 2
Loam	0.25	0.24	$\theta_0^{\min} = 0.18, \theta_0^{\max} = 0.35$	$\theta_0^{\min} = 0.18, \theta_0^{\max} = 0.25$
Sandy loam	0.23	0.12	$\theta_0^{\min} = 0.1, \theta_0^{\max} = 0.25$	$\theta_0^{\min} = 0.1, \theta_0^{\max} = 0.25$
Sand	0.05	0.05	$\theta_0^{\min} = 0.05, \theta_0^{\max} = 0.1$	$\theta_0^{\min} = 0.05, \theta_0^{\max} = 0.1$

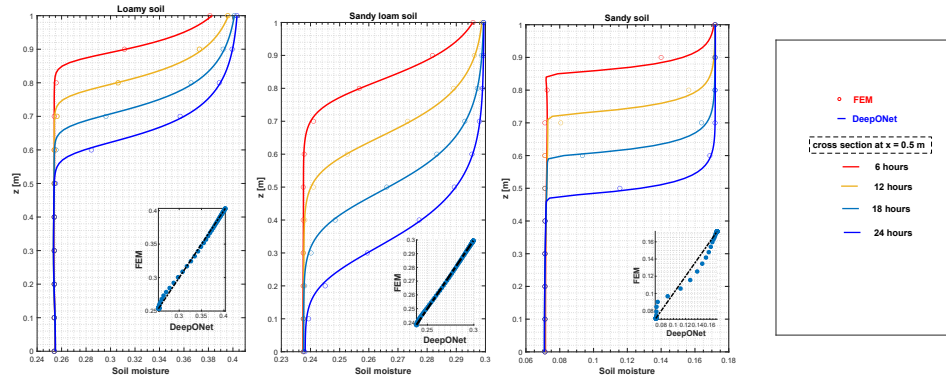
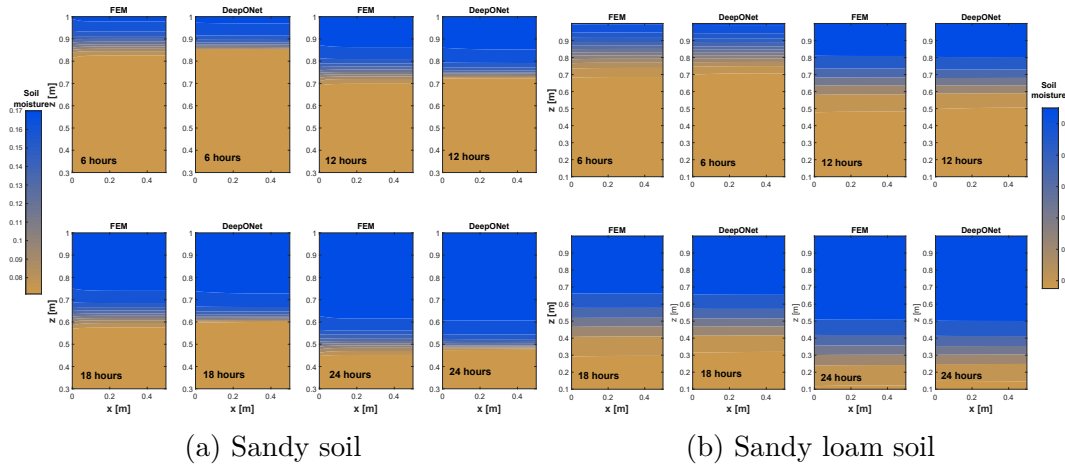


Figure 8.5 Scenario (a)-non-parametric /geometry 1. Top: 2D snapshots from DeepONet and FEM at different times for sandy and sandy loam soils. Bottom: Cross-sections at $x = 0.5$ m at different times for loam, sandy loam, and sandy soils.

simulation for a single sample of initial soil moisture. As we introduce more initial conditions, the time required for the FEM simulation will exponentially increase, while DeepONet's inference time remains relatively small. This indicates that DeepONet serves as an efficient surrogate model for rapid predictions, which is suitable for optimization and uncertainty quantification tasks.

Extrapolation results. We evaluate DeepONet's ability to generalize by introducing new unseen soil moisture data as initial conditions. This test data is generated as slight variations of the training data, defined as $\theta_0^{\text{test}} = \theta_0^{\text{train}} \pm \varepsilon_0$, where ε_0 takes values of 0.001, 0.01, and 0.1. This scenario involves extrapolation, prompting us to fine-tune the pre-trained DeepONet with physics to assess whether additional training can yield more accurate results compared to using the DeepONet without fine-tuning. The results are depicted in Figure 8.7 for all cases. The accuracy of the DeepONet remains satisfactory when the perturbation is small, even without any additional fine-tuning. However, as the perturbation increases, the accuracy decreases, and fine-tuning the pre-trained DeepONet can lead to more accurate results. We note that the fine-tuning phase took around 60 seconds on a single Nvidia A100 GPU.

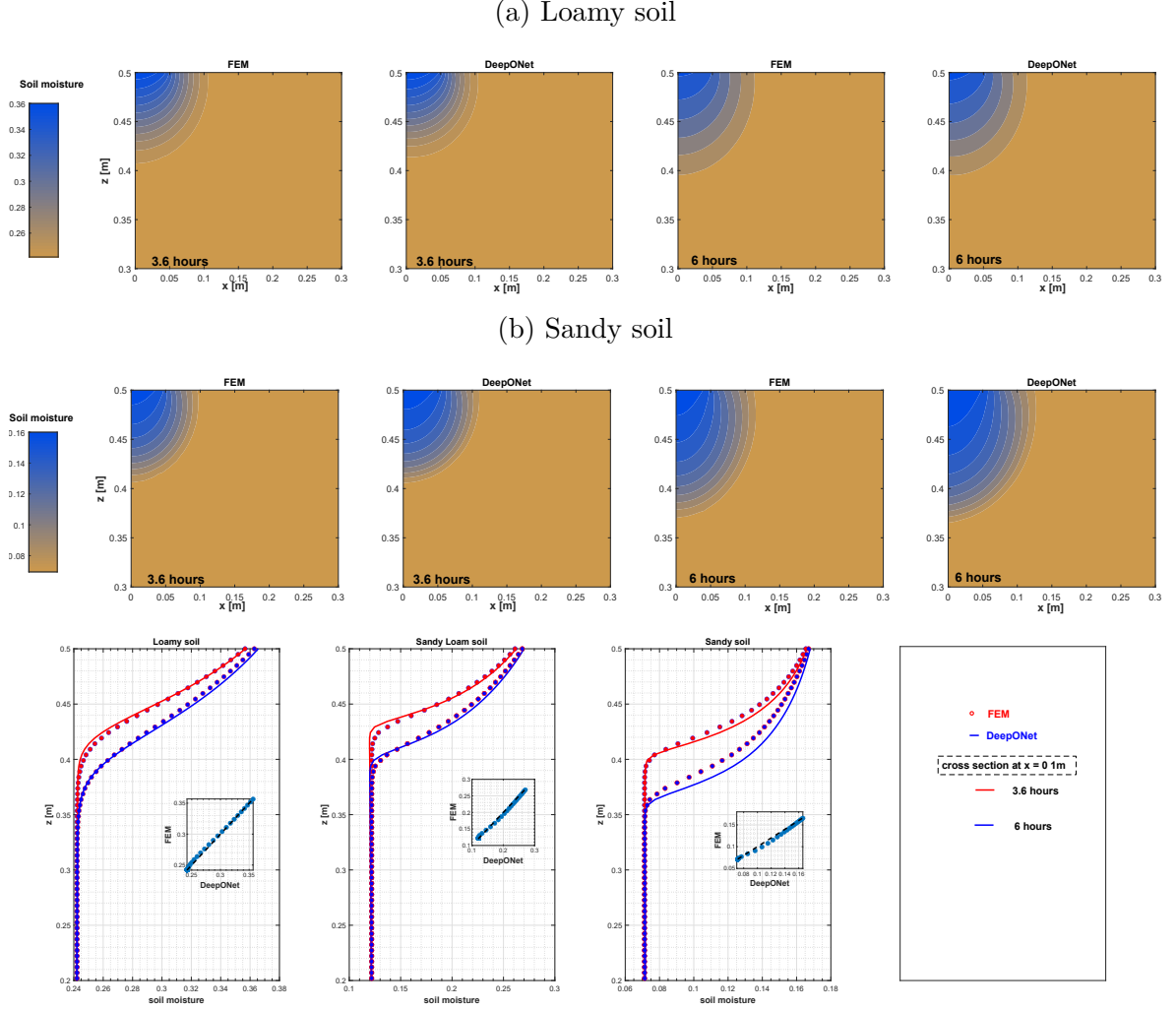


Figure 8.6 Scenario (a)-non-parametric /geometry 2. Top: 2D snapshots from DeepONet and FEM at different times for sandy soil and sandy loam soil. Bottom: Cross-sections at $x = 0.5$ m at different times for loam, sandy loam, and sandy soils.

Inverse problem results. Now, we aim to test an inverse problem scenario where the soil parameters and the initial soil moisture θ_0 for each soil are perturbed with Gaussian random noise, as follows:

$$\mathbf{p}_{\text{noisy}} = \mathbf{p}_{\text{origin}} + \mathcal{N}(0, \sigma), \quad (8.10)$$

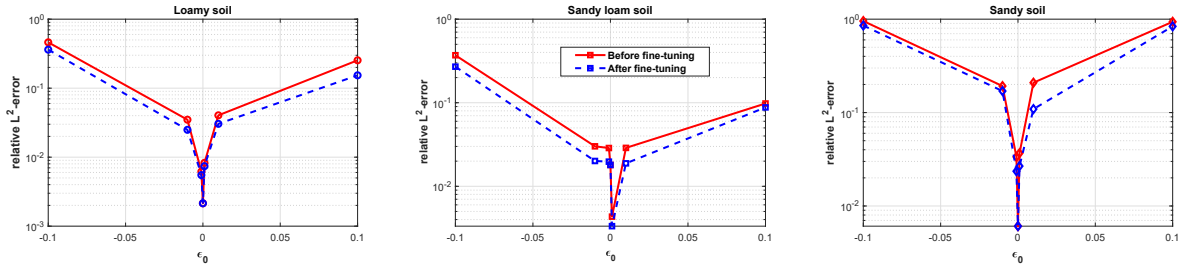
where $\mathbf{p}_{\text{origin}}$ represents a soil parameter or initial soil moisture, i.e., $\mathbf{p}_{\text{origin}} \in \{\theta_r, \theta_s, \alpha_v, n_v, m_v, K_s, \theta_0\}$. Here, $\sigma = 2\%$ indicates the magnitude of the noise added.

This scenario is an ill-posed problem and, at the same time, an extrapolation scenario as we change the data used in the training phase, including soil hydraulic parameters and initial soil moisture profile. Thus, in this situation, we need to guide the training of DeepONet in the fine-tuning stage by adding available soil moisture data. For this purpose, we place four sensors in the soil domain at the following depths: $z = 3$ cm, 10 cm, 17 cm, 25 cm, 40 cm. We

Table 8.3 Scenario (a)-Average and standard deviation of the relative L^2 -error. $\cdot|\cdot$ indicates training | test errors.

Soil type	Non-parametric initial conditions		Parametric initial conditions			
	Geometry 1	Geometry 2	Geometry 1		Geometry 2	
Loam	$0.21 \pm 2\%$	$0.56 \pm 1.5\%$	$1.32 \pm 0.85\%$	$2.52 \pm 0.7\%$	$1.45 \pm 1.46\%$	$3.15 \pm 2.16\%$
Sandy loam	$0.63 \pm 1.2\%$	$1.65 \pm 3.5\%$	$2.98 \pm 0.55\%$	$4.2 \pm 1.2\%$	$4.18 \pm 4.55\%$	$6.38 \pm 1.2\%$
Sand	$2.87 \pm 5.5\%$	$4.28 \pm 4.3\%$	$8.09 \pm 0.36\%$	$10.2 \pm 2.16\%$	$9.43 \pm 5.76\%$	$9.56 \pm 2.26\%$

(a) Geometry 1



(b) Geometry 2

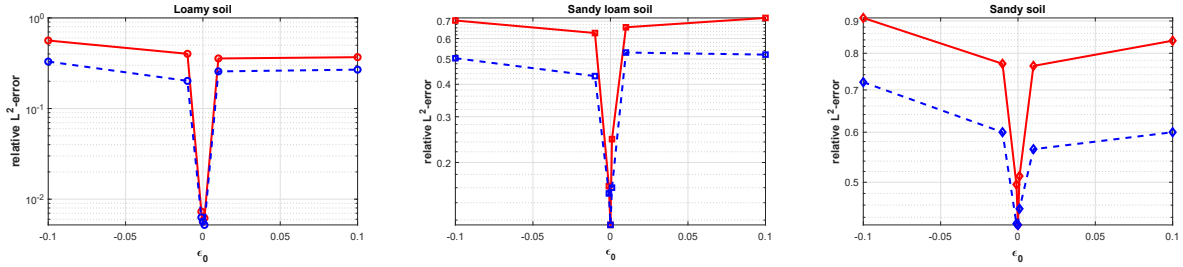


Figure 8.7 Scenario (a)-non-parametric : The relative L^2 -error of DeepONet for each soil type and geometry. We plot both the accuracy before and after fine-tuning the pre-trained DeepONet with physics.

choose the sensors close to the soil surface due to the high change in soil moisture near the surface. We record soil moisture data each hour.

The results obtained from DeepONet before and after fine-tuning, along with the FEM reference solution, are depicted in Figure 8.8. We note that DeepONet predictions without fine-tuning may be erroneous due to the ill-posed nature of the problem. The DeepONet was trained on specific soil parameters and initial conditions, while the test data represents a different setting, highlighting the necessity for additional data to refine its predictions. The fine-tuning with physics and available data improved the accuracy of DeepONet and helped the pre-trained model to capture the real soil moisture profile for loamy and sandy loam soils. Unfortunately, sandy soil remains challenging for DeepONet; however, the accuracy was improved by 23% for both geometries compared to the pre-trained DeepONet prediction. An

important observation from Figure 8.8 is that the DeepONet’s predictions deviate more from the true profile as soil texture becomes coarser (e.g., in sandy soils). This demonstrates the complexity of modeling coarse-textured soils and suggests that more sophisticated architectures should be integrated into the DeepONet model to improve its extrapolation capabilities.

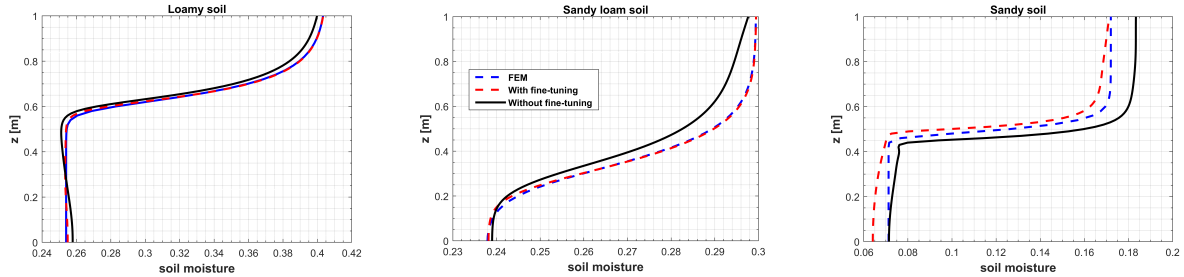


Figure 8.8 Scenario (a)-non-parametric : Comparison of soil moisture cross-sections from DeepONet (with and without fine-tuning) and the FEM reference solution. The cross-section is taken from Geometry 1 at position $x = 0.5$ m, observed at the final simulation time of 1 day.

8.4.1.2 Scenario (a)-parametric initial conditions

Settings. The configuration and architecture used in this scenario are summarized in Table 8.5. The dry and wet soil moisture values (minimum and maximum values) for each soil type are listed in Table 8.2.

Interpolation results. The results obtained for different initial soil moisture conditions from the DeepONet and the FEM reference solution are depicted in Figure 8.9. For each soil type and geometry, we plot two samples from the training data set and two samples from a test data set containing 50 additional initial soil moisture conditions. The DeepONet 2D profiles agree well with those obtained from the FEM solution. The average relative L^2 -error is reported in Table 8.3. The DeepONet produces accurate results for loamy and sandy loam soils, while the error is significantly greater for sandy soil. This discrepancy is due to the gravity-dominated nature of coarse soils and the reliance on collocation points in the DeepONet approach. A variational formulation of the physical loss function $\mathcal{L}$ may enhance the accuracy of DeepONet for solving gravity-dominant infiltration problems. In our case, which involves irrigation systems, the water velocity is generally low, and the most challenging aspect is capturing the sharp front of the solution.

Another important observation from this numerical test is that accuracy decreases when the initial soil moisture is either too close to saturation θ_s or too close to the residual moisture θ_r . This challenge is also observed with the FEM scheme (Celia *et al.*, 1990b). The main reason is that infiltration into dry soils generates highly nonlinear wetting fronts, leading to rapid changes in spatial gradients of soil hydraulic properties (Farthing and Ogden, 2017). It is important to note that the accuracy of DeepONet decreases for geometry 2 due to the complexity of boundary conditions, which are imposed only in a small portion of the soil surface. These infiltration boundary conditions lead to varying soil moisture profiles in both

the x and z -directions. However, for geometry 1, the soil moisture profile is simpler due to the one-dimensional nature of the problem, where DeepONet achieves better accuracy. Figure 8.17 presents a visual comparison of CPU times between FEM and DeepONet, highlighting the substantial difference between the two solvers. DeepONet consistently outperforms FEM in terms of CPU time while maintaining reasonable accuracy.

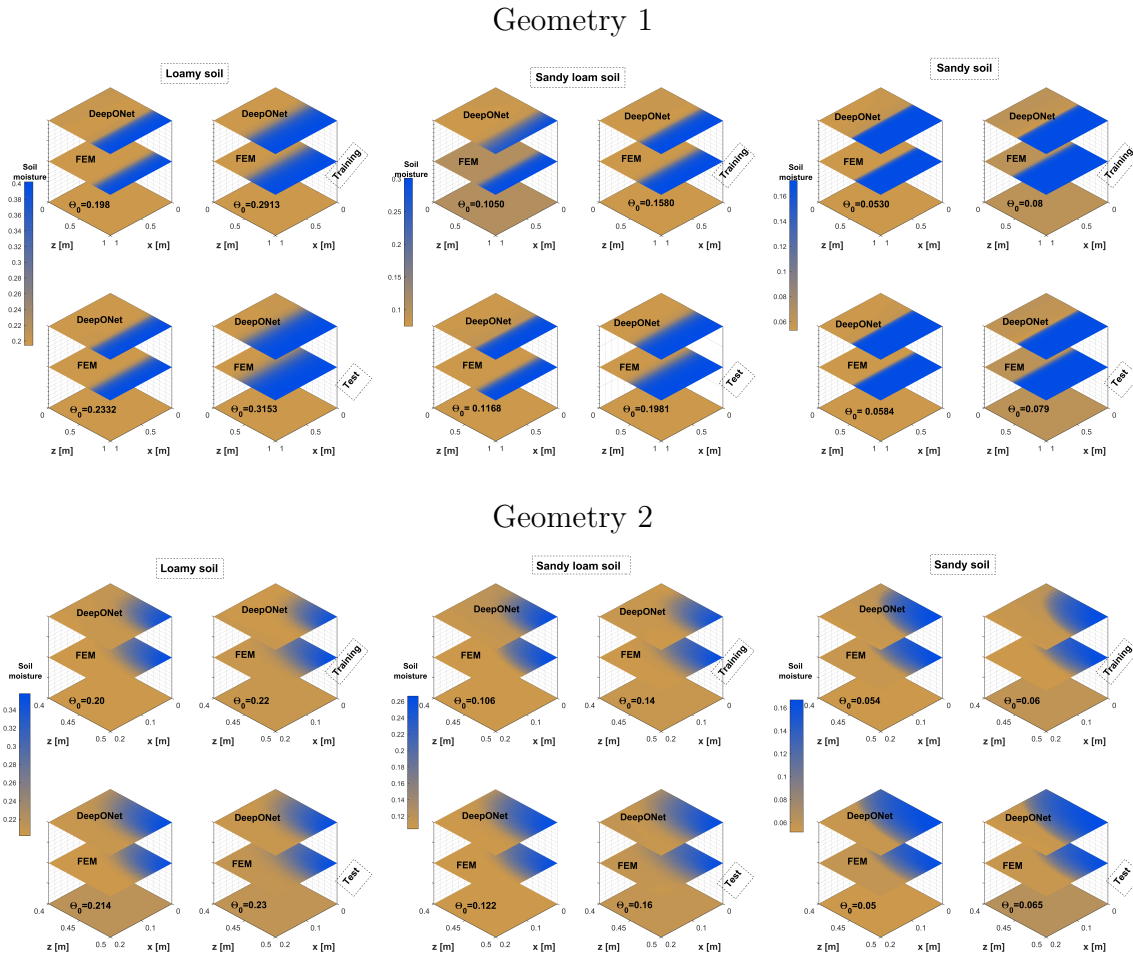


Figure 8.9 Scenario (a)-parametric: 2D snapshots from DeepONet and FEM at the final simulation time for loam, sandy loam, and sandy soils. The figures include two samples from the training data and two samples from the test data. Each panel shows the initial condition at the bottom, followed by the predictions from the FEM and DeepONet models.

Extrapolation results. In this parametrization scenario, we trained the DeepONet on uniform initial soil moisture conditions in space. To test its efficacy in extrapolation, we consider a scenario where the initial soil moisture depends on space, which varies linearly with depth from a predefined value at the soil surface to the bottom, while remaining uniform in the x -direction. We limit the analysis to loamy soil and geometry 1. The initial soil moisture profile is defined as (refer to Figure 8.10) $\theta_0(z) = (\theta_{\text{surface}} - \theta_{\text{bottom}})z + \theta_{\text{bottom}}$, where the

soil surface is located at $z = 1$. Here, $\theta_{\text{surface}} = 0.18$ and $\theta_{\text{bottom}} = 0.22$. This results in dry soil moisture at the soil surface with increasing water content as depth in the soil profile increases. Such a profile is typical when the water table is near the domain of interest, making a hydrostatic initial condition a suitable choice. We ensured that $\theta_0(z) \in (\theta_r, \theta_s)$ for all z . The results obtained are shown in Figure 8.10, comparing DeepONet with and without fine-tuning against the FEM reference solution. As expected, the pre-trained DeepONet failed to provide accurate results for this extrapolation scenario with a relative L^2 -error of 50%. Fine-tuning with physics improves the accuracy of the pre-trained model and predicts the sharp front well ($\sim 3.5\%$). However, despite the importance of the fine-tuning phase, we observed unstable training when increasing the gap ($\theta_{\text{surface}} - \theta_{\text{bottom}}$), which makes the extrapolation more complex. This underscores the necessity for more robust metrics to measure the complexity of extrapolation tasks related to subsurface hydrology problems.

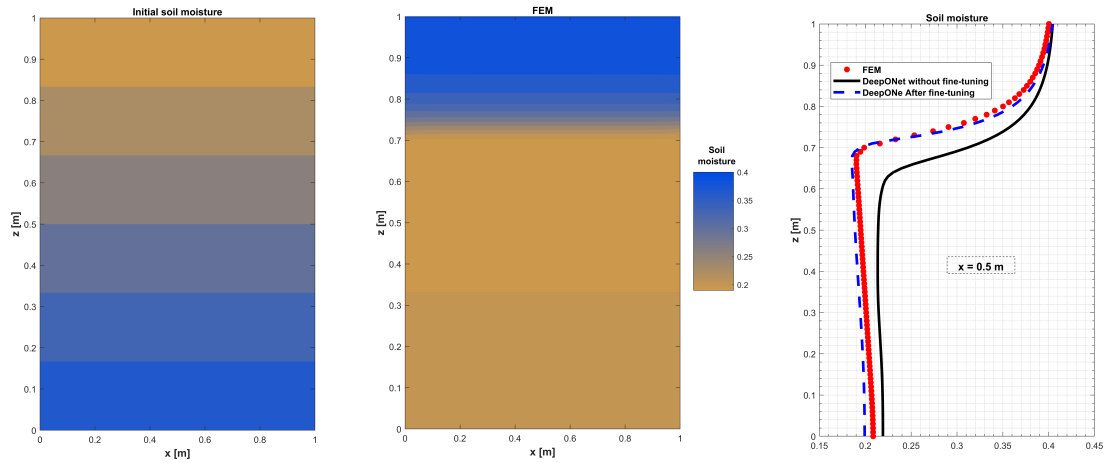


Figure 8.10 Scenario (a)-parametric: Results from DeepONet in an extrapolation scenario for loam soil in geometry 1, with and without fine-tuning. The initial soil moisture varies linearly from the surface to the bottom (left). Soil moisture profiles from FEM at $T = 1$ day (middle). Cross-sections from FEM and DeepONet with and without fine-tuning (right).

8.4.2 Scenario (b): Mapping water inlet boundary conditions to the corresponding pressure head profiles

For scenario (b), we aim to map different surface water fluxes to the corresponding pressure head profiles. Three types of parameterization are discussed. In the first, we generate N_{samples} time-independent water fluxes, assuming that water will infiltrate the soil continuously until the final simulation time is reached. This parameterization is referred to as time-independent water flux. In the second parameterization, we consider that the infiltration or irrigation

time (the water pulse duration) is less than the total simulation time:

$$q_0^{(i)}(t) = \begin{cases} q_0, & \text{if } t \leq t_i < T, \\ 0, & \text{else.} \end{cases} \quad (8.11)$$

Here, N_{samples} points t_i are uniformly generated from the simulation interval I to produce different pulse durations. In this scenario, water infiltrates the soil domain for a duration t_i , after which the boundary condition switches immediately to a no flux condition, leading to redistribution of water within the soil domain. This parameterization is referred to as pulse duration.

The final parameterization represents a more general and realistic scenario involving a drying-wetting cycle, simulating an infiltration-evaporation situation. The fluxes are generated as follows:

$$q_0^{(i)}(t) = \begin{cases} q_0^j, & \text{if } t \in [t_j, t_{j+1}), \text{ for } j = 0, 1, 2, 3, \\ \text{with } t_0 = 0, t_1 = 0.25T, t_2 = 0.5T, t_3 = 0.75T, t_4 = T, \end{cases} \quad (8.12)$$

In the parameterization (8.12), the simulation domain I is divided into four subintervals, each corresponding to a water inlet q_0^j . The values of these fluxes are randomly generated from the range $[-0.75K_s, 0)$ for each soil texture. In this scenario, we have four water irrigation events, each with a specific water inlet. This parameterization is referred to as a drying-wetting cycle with different fluxes.

In all these parametrizations, we have set the initial soil moisture to $\theta_0 = 0.20$, and we exclude sandy soil from the analysis. All these parametrizations are illustrated in Appendix 8.6. Table 8.6 summarizes the configuration and architectural details used in each scenario.

8.4.2.1 Scenario (b)-Time independent water flux

Results. The results are depicted in Figure 8.11, where we plot two samples: one from the training data and one from the test data for each geometry for loamy soil. The DeepONet provides a good prediction of water movement, and the cross-sections show that the sharp front is well predicted. The average relative L^2 -error obtained for both training and test datasets is reported in Table 8.4. In this parameterization scenario, the accuracy of DeepONet matches that of the first scenario, although it decreases for sandy loam soil due to the dominance of gravity forces. The FEM solution also necessitates a finer mesh near the water infiltration region for numerical stability. The DeepONet model generalizes well in the interpolation context across both soils and geometries, exhibiting no signs of overfitting. Notably, DeepONet outperforms FEM in terms of computational efficiency, being 60 times faster at predicting 50 new unseen surface water fluxes. The main advantage is that DeepONet predicts all the water fluxes simultaneously.

This parameterization is simplistic and not realistic due to the duration of the infiltration pulse time. To better reflect actual field scenarios, subsequent parameterizations will incorporate abrupt changes in the water flux at the boundary, enabling a more precise simulation of the dynamic nature of infiltration events.

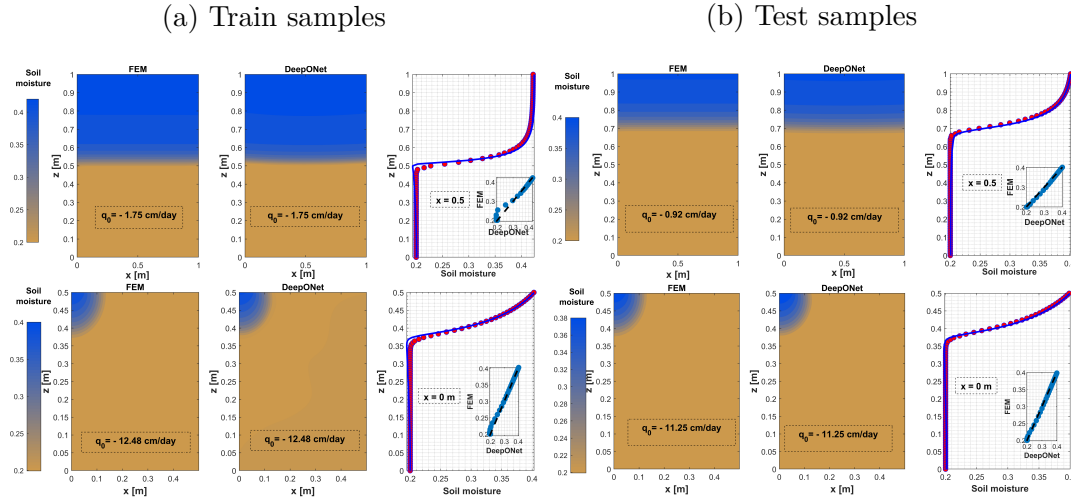


Figure 8.11 Scenario (b)-time-independent water flux: 2D snapshots from DeepONet and the FEM reference solution at the final simulation time for loamy soil. Each snapshot includes one surface water flux from the training data and one from the test data for each geometry. The solid line represents DeepONet, while the red circles represent the FEM solution in the cross-sections.

Table 8.4 Scenario (b)-Average and standard deviation of the relative L^2 -error for scenario (b). $\cdot|\cdot$ indicates train | test errors.

Soil type	Time-independent water flux				Pulse duration			
	Geometry 1		Geometry 2		Geometry 1		Geometry 2	
Loam	$0.93 \pm 1.09\%$	$1.28 \pm 1.89\%$	$5.43 \pm 1.00\%$	$6.98 \pm 2.15\%$	$3.96 \pm 4.24\%$	$4.57 \pm 3.75\%$	$5.31 \pm 2.22\%$	$6.2 \pm 1.22\%$
Sandy loam	5.25 ± 4.85	$6.32 \pm 8.44\%$	$8.30 \pm 2.30\%$	$9.36 \pm 1.31\%$	$5.64 \pm 1.00\%$	$7.64 \pm 2.12\%$	$8.23 \pm 3.00\%$	$9.83 \pm 3.00\%$
Soil type	Drying-wetting cycle with different fluxes							
	Geometry 1				Geometry 2			
Loam	$1.63 \pm 0.5\%$		$3.94 \pm 1.53\%$		$4.22 \pm 0.20\%$		$5.7 \pm 2.0\%$	
Sandy loam	$8.45 \pm 1.00\%$		$9.83 \pm 3.00\%$		$12.45 \pm 1.00\%$		$13.2 \pm 1.05\%$	

8.4.2.2 Scenario (b)-Pulse duration

Results. Figure 8.12 presents the results obtained from the DeepONet model and the FEM reference solution for three samples. The figure includes 2D profiles, cross-sections, and the soil surface water flux for geometry 1 and the loamy soil. DeepONet performs well in this parametric scenario. One significant challenge of this problem is the discontinuity in the surface flux boundary condition, which standard neural networks struggle to accurately capture due to the utilization of continuous activation functions ($\tanh$). The surface water flux predicted by DeepONet closely approximates the exact imposed flux. However, accurately imposing discontinuous boundary conditions in DeepONet, which is based on a collocation method, presents challenges. Nonetheless, accuracy can be enhanced by adapting DeepONet's output to directly enforce boundary conditions using appropriate distance functions to the

boundaries, as proposed in (Sukumar and Srivastava, 2022). However, the time-dependent Neumann boundary condition for water flux poses challenges for exact enforcement using the hard constraint approach.

We report the average relative L^2 -error in Table 8.4. The results demonstrate the good performance of DeepONet, with errors less than 10% for all cases, albeit slightly larger for sandy loam soil. Again, in terms of computational cost during the inference phase, DeepONet outperforms the FEM solver. In this test scenario, when providing a new pulse duration, DeepONet produces results in 50 ms, whereas the FEM solver requires 54 seconds. This performance advantage becomes more significant for long-duration simulations.

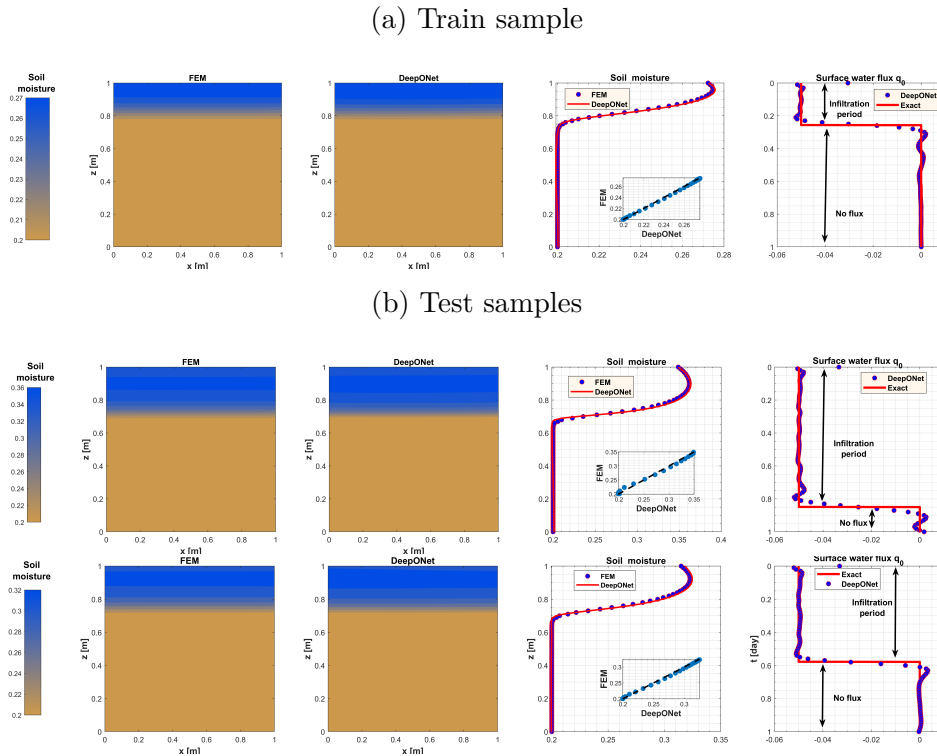


Figure 8.12 Scenario (b)-pulse duration: 2D snapshots along with cross-sections from DeepONet and FEM at the final simulation times for geometry 1 for the loamy soil. The plots include one sample from the training data and two samples of test data.

8.4.2.3 Scenario (b)-drying-wetting cycle with different fluxes

Results. Figure 8.13 illustrates the results obtained from DeepONet and FEM for four samples. This scenario is more complex due to the inclusion of various discontinuities within the time-dependent surface water flux. Despite this additional complexity, DeepONet can capture the discontinuities without any additional training effort. The average relative L^2 -error is reported in Table 8.4. DeepONet achieves good accuracy ($\sim 5\%$) for loamy soil, while larger errors ($\sim 10\%$) occur for sandy loam soil in both geometries. The error is larger for geometry 2 due to the complexity of boundary conditions, especially the discontinuities in

the time-dependent surface water flux located in the left portion of the soil surface.

We extend our analysis of this drying-wetting parameterization by fixing two observation points $\mathbf{x}_1$ and $\mathbf{x}_2$, with $\mathbf{x}_1$ is located at the soil surface, and $\mathbf{x}_2$ is at a depth of 5 cm below the surface. Observing these points is a common strategy to track changes in soil moisture and water flux near the emitter over time. Such analysis aids in developing efficient irrigation systems that maintain optimal water content near the root zone while mitigating deep percolation issues.

The results obtained from DeepONet and the FEM solution for loamy soil in geometry 1 are depicted in Figure 8.14. The figure illustrates the variations in soil moisture and water flux at the two fixed observation points. We plot three soil surface water fluxes: one from the training data, one from the test data, and the third from an extrapolation scenario. In the extrapolation scenario, we force the surface water flux to have two values: either an irrigation event ($q_0 = -5$ cm/day) or no infiltration ($q_0 = 0$).

For both the training and test samples, DeepONet shows good performance at both observation points, effectively capturing soil moisture dynamics and water flux, even during abrupt changes in surface water flux. The relative L^2 -error is less than 1.5%.

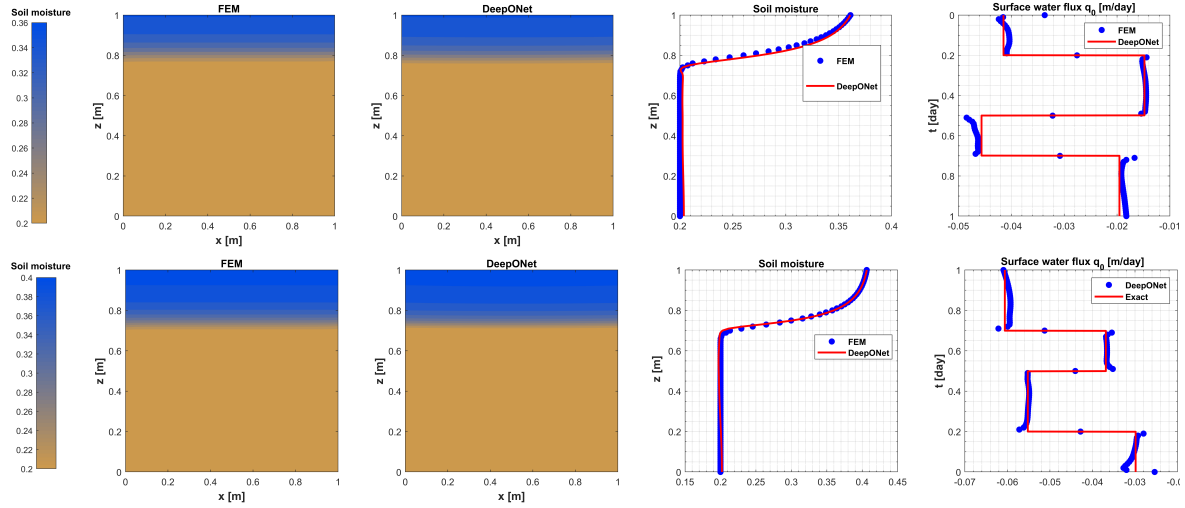
For the extrapolation scenario, we present the results before and after fine-tuning the pre-trained DeepONet model (see Figure 8.14-c). In the fine-tuning process, we gathered some scarce data from two sensors placed below the soil surface at depths of 10 cm and 20 cm, recording data every hour. To simulate real-world measurement errors, the data were perturbed with Gaussian noise of magnitude 1% and zero mean. As expected, fine-tuning improved the accuracy of DeepONet by 53%. The fine-tuning process took less than one minute on a single Nvidia A100 GPU card. Even with noisy data, the DeepONet training benefited from the additional information to improve its prediction accuracy while respecting the physical constraints. A significant advantage of this approach is the straightforward inclusion of data into the DeepONet model, making it an attractive solver. Unlike standard solvers that require repeated forward simulations coupled with optimizers during the calibration stage, DeepONet can efficiently integrate new data for improved performance.

8.5 Discussion

Here, we comprehensively examine various aspects, covering both advantages and limitations, and propose enhancements that could be integrated into the DeepONet for modeling water flow in unsaturated soils.

The development of an efficient simulator for subsurface water modeling is a key step, and it is very demanding. In this study, we propose a physics-informed DeepONet for fast and accurate prediction of soil moisture distribution within unsaturated 2D soils for different parametric scenarios. We obtained promising results, although several limitations need to be addressed in future research. Specifically, we focused on homogeneous soils where soil parameters are assumed to be spatially independent, which is unrealistic for many real-world soil conditions. It is well known that PINNs struggle with solving PDEs that involve discontinuities, which are common in heterogeneous soils where water saturation gradients can be very steep and discontinuities occur at interfaces between soil layers. To address this, specialized architectures are required. For instance, approaches like the extended PINN developed in (Jagtap

(a) Train samples



(b) Test samples

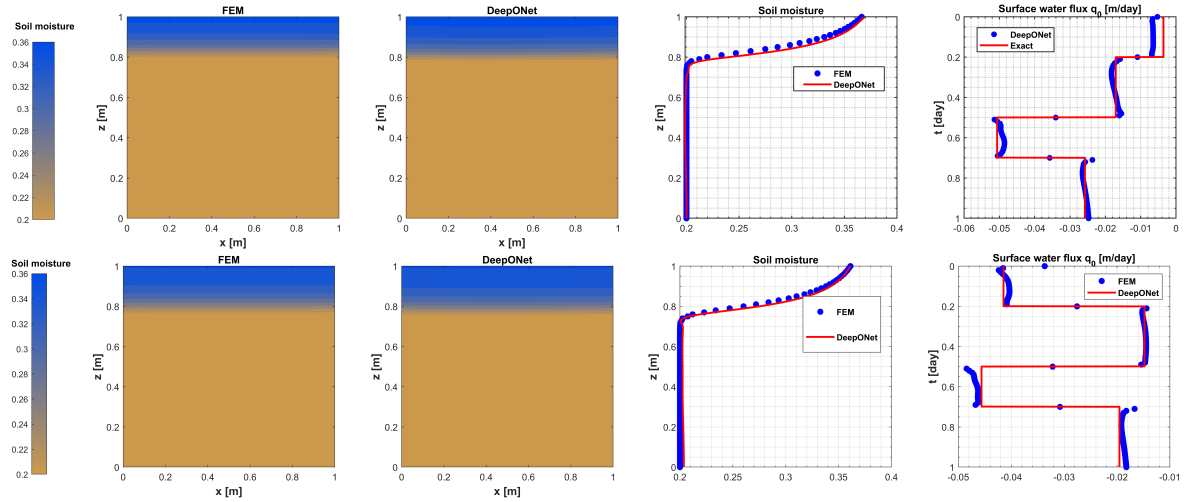


Figure 8.13 Scenario (b)-drying-wetting cycle with different fluxes: 2D snapshots from DeepONet and FEM at the final time of simulation for loamy soil and geometry 1, including two surface water fluxes from training data and two surface water fluxes from test data.

and Karniadakis, 2020) could be adapted or extended to suit the DeepONet model. Alternatively, encoding the governing PDE using its weak formulation (Goswami *et al.*, 2022a) could enhance DeepONet's ability to accurately capture heterogeneous soils and discontinuities in soil surface flux. In our previous study (Kamil *et al.*, 2024e), we proposed a mixed-PINN approach for modeling the coupled dynamics of water and multiple solutes in layered soils. The technique was based on solving each PDE with its mixed formulation by deriving both the solution and the corresponding flux. Accurate results were obtained, and this approach can be extended to the DeepONet model in future studies.

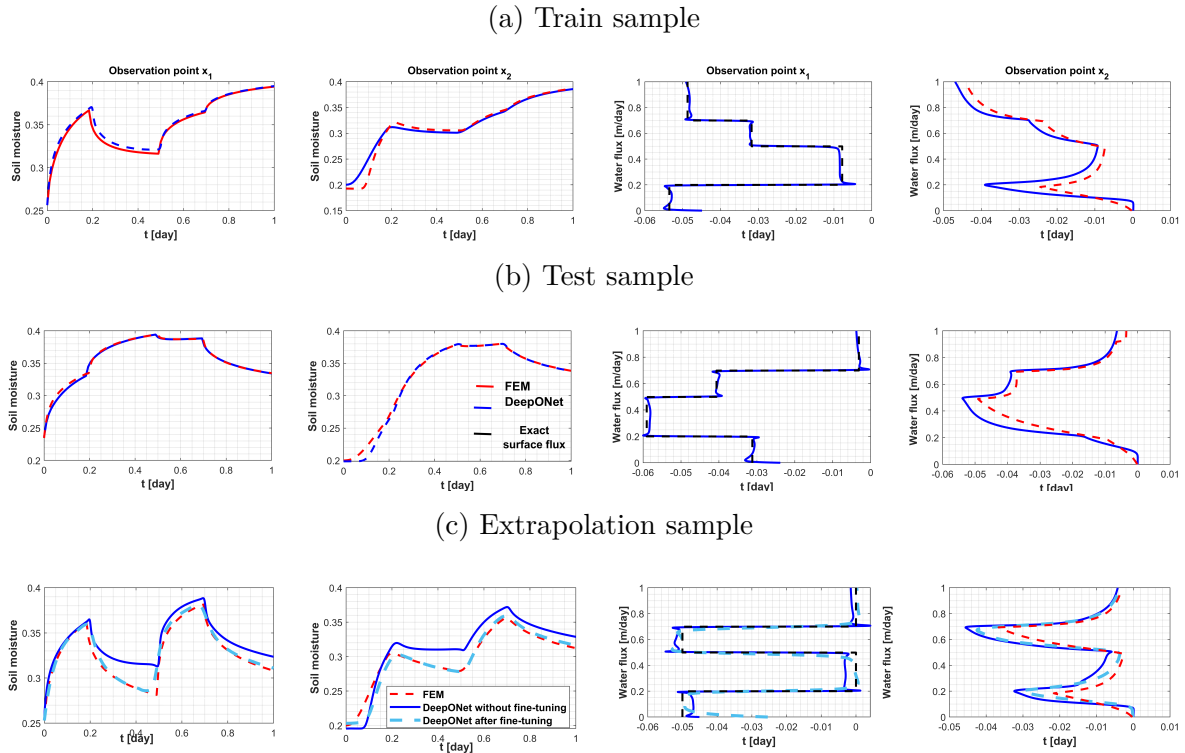


Figure 8.14 Scenario (b)-drying-wetting cycle with varying fluxes: Temporal variation of soil moisture and water flux at observation depths of $z = 0$ cm (observation points $\mathbf{x}_1$) and $z = 5$ cm (observation points $\mathbf{x}_2$) for training, test, and extrapolation surface water flux scenarios.

In this study, we have demonstrated that for very coarse soils (e.g., sand), training could be unstable due to the dominance of gravity, which leads to rapid changes within the soil domain. Collocation methods are generally unsuited for advection-dominated problems, necessitating reformulation of the loss function in the DeepONet model to ensure accurate predictions. This challenge becomes more complex for long-term simulations and heterogeneous soils. To address this issue, we propose a transfer learning methodology that leverages the knowledge gained from solving a related problem. Initially, we train a DeepONet model on a loamy soil as the source task, which allows the model to learn the underlying patterns and relationships in soil moisture dynamics. Subsequently, we tackle the sandy soil problem as the target task by initializing the model's parameters with those obtained from the pre-trained loamy soil DeepONet solver. This approach enables the model to adapt and fine-tune its knowledge to the specific characteristics of sandy soil, thereby improving the efficiency and accuracy of the simulation.

In the final parametrization scenario, we analyzed the simulation of wetting-drying cycles for different soil surface fluxes, highlighting the efficiency of the DeepONet model in such scenarios. However, incorporating additional terms into the Richards equation, such as root water uptake, may introduce further challenges. The inclusion of root water uptake leads

directly to the need for atmospheric boundary conditions, which are not given a priori and should be determined based on the equilibrium between the soil surface and atmospheric conditions (Feddes *et al.*, 1976). This coupling between soil moisture dynamics and atmospheric conditions introduces further complexity to the modeling process, as it requires the accurate representation of the interactions between the soil, plant, and atmosphere. To address this challenge, future research could explore the integration of root water uptake and atmospheric boundary conditions into the DeepONet framework, potentially leveraging transfer learning techniques to build upon the knowledge gained from simpler scenarios. By incorporating these additional factors, the DeepONet model could provide a more comprehensive and realistic simulation of soil moisture dynamics in complex, real-world environments.

In general, relying solely on the soil moisture profile is inadequate for efficient fertigation systems. It is essential to incorporate additional PDEs to simulate the transport of fertilizers (such as urea, ammonium, nitrate, phosphorus, $\dots$) and contaminants. This necessitates addressing multiphysics simulation problems, where adapting the DeepONet model to infer multiple solutions simultaneously can be quite challenging. Proposed architectures for multiphysics, like DeepM&Mnet (Cai *et al.*, 2021b), need to be extended to subsurface modeling to efficiently handle water flow and multiple solute transport species in the unsaturated zone. This approach would be valuable for using a well-trained DeepONet model to rapidly predict soil moisture profiles and solute concentrations for any new irrigation and fertigation strategy. The transfer learning PINN approach proposed in (Kamil *et al.*, 2024e) can be used to develop a DeepONet for solving multiphysics problems.

Extrapolation and inverse modeling pose significant challenges in scientific modeling. We address this issue by fine-tuning a pre-trained DeepONet to predict the output of test data beyond the support of the training data. This is particularly relevant in cases where initial conditions vary spatially, while the pre-trained model was trained under uniform conditions and other scenarios. Fine-tuning with physics or available data can enhance prediction accuracy. However, assessing the complexity of the extrapolation task requires a suitable metric. Existing metrics proposed in (Zhu *et al.*, 2023) may not directly apply to subsurface problems. It is important to note that fine-tuning does not always ensure convergence or stable training, especially for complex extrapolation tasks. Therefore, theoretical research is crucial to develop more reliable strategies for addressing extrapolation challenges. A recent notable effort within the PINNs framework addressed time extrapolation in (Kim *et al.*, 2021), proposing a new regularization strategy to ensure effective minimization of the PDE residual term. However, extrapolation is multidimensional, extending beyond time, where any deviation from the PDE parameters outside the training support constitutes an extrapolation task.

In this study, we focus solely on a single-input DeepONet, where the operator maps from a single input function, such as the initial condition or surface water flux to the corresponding solution. In reality, multiple parameters can be changed simultaneously, including soil hydraulic parameters, initial/boundary conditions, and others. An efficient surrogate model for such conditions is the use of multiple-input DeepONet (Jin *et al.*, 2022). This approach allows for the analysis of different settings, although it requires efficient training of the model, as the number of trainable parameters will significantly increase.

In each numerical test, we have provided details on the number of trainable parameters and

the training time on a GPU device. While the number of trainable parameters is significantly smaller compared to large language models (the GPT-3 model has 175 billion parameters (Floridi and Chiriatti, 2020)), efficient training and acceleration can still be achieved. For DeepONet, a single GPU card suffices, but for optimal performance, exploring parallel training techniques such as model parallelism or data parallelism is advisable. These methods can lead to rapid training, which is particularly beneficial when dealing with multi-input DeepONet models.

The architecture of the proposed DeepONet model is based on a modified MLP with two encoders. Using architectures suitable for time-dependent data, such as long-short-term memory (LSTM) networks, and gated recurrent units (GRUs), could yield very accurate results, but the challenge is how to efficiently calculate the physical constraints. Several attempts have already been made in this direction (Zhang *et al.*, 2020), although more research should be conducted. Another important architecture is the convolutional neural network (CNN), which is better suited for handling 2D spatially varying inputs. For example, when using random initial soil moisture profiles as input data, a standard MLP may fail to capture fine-scale features accurately, whereas a CNN can better preserve and extract spatial patterns.

In this study, we have demonstrated the efficiency of the DeepONet model for accurate and rapid prediction of soil moisture profiles within a fraction of a second. While the accuracy of DeepONet cannot yet be compared to numerical methods (e.g., FEM), which have benefited from at least 60 years of development, scientific machine learning is in its infancy and rapidly improving due to advancements in CPUs, GPUs, and efficient programming languages (e.g., Julia). Finding an intermediate solution that combines the speed of deep operator learning with the high accuracy of numerical methods is a promising direction for future research.

8.6 Conclusion

In this study, we propose an accurate and rapid deep learning solver for modeling water flow in unsaturated soils. The water flow is modeled using the highly nonlinear Richards equation along with the corresponding initial and boundary conditions. The deep learning model is a physics-informed DeepONet designed to learn the mapping between various initial soil moisture conditions and surface water fluxes to the corresponding pressure head profile across the entire time-space domain. To enhance the accuracy of the DeepONet model, we incorporate a modified architecture that includes a multilayer perceptron network along with two encoders (Wang *et al.*, 2022b).

Several numerical tests were conducted to assess the reliability of the proposed model, all of which demonstrated its good agreement with a finite element reference solution. For fine and medium-fine textured soils, DeepONet provided efficient predictions, although it struggled with coarse soils due to the gravity-dominated nature of the problem. Inverse modeling and extrapolation tasks were also analyzed, showing that fine-tuning a pre-trained DeepONet model with either physics constraints or available soil moisture data improves the model's efficiency.

Our next objective is to extend the DeepONet model to include various aspects such as soil heterogeneity, root water uptake, and multiple solute dynamics under field boundary conditions (e.g., atmospheric boundary conditions). This extension will enable a more realistic

assessment of the DeepONet model and allow us to compare its accuracy with available field or laboratory data, ultimately aiding in the development of efficient fertigation strategies for sustainable agriculture.

Code availability

The source code used in this study is publicly available from the GitHub repository <https://github.com/Hamza-Kamil/DeepONet-WaterFlow-Soils>.

Appendix A. Deep operator learning

Here we provide a brief description of deep operator learning, focusing on the architectures proposed in (Lu *et al.*, 2021a; Wang *et al.*, 2021b), which are employed in this study.

The main idea behind deep operator learning is to learn the mapping between two functional spaces using a surrogate model, such as a neural network (NN). More precisely, consider the following operator:

$$\begin{aligned} G : \quad \mathcal{P} &\rightarrow \mathcal{S} \\ u &\mapsto G(u), \end{aligned} \tag{8.13}$$

where $\mathcal{P}$ and $\mathcal{S}$ are two Banach spaces. For example, $\mathcal{P}$ could denote the space of parameters for a specific PDE, while $\mathcal{S}$ might represent the space of corresponding solutions to that PDE. The goal is to train a surrogate model G_Θ that can map input data $u \in \mathcal{P}$ (source terms, initial/boundary conditions, etc.) to the corresponding solution $G_\Theta(u) \in \mathcal{S}$, aiming to approximate the operator G . Here, Θ represents the trainable parameters of the surrogate model, which are learned using an optimization algorithm (e.g., Adam). Such a mapping is not trivial to learn, especially if the operator G is highly nonlinear, which is often the case. According to the theoretical results presented in (Chen and Chen, 1995), which refer to the universal approximation theorem for operators, it is proven that a NN with a single hidden layer of sufficient width can accurately approximate any nonlinear continuous operator. This result underscores the potential of deep neural networks in learning continuous and complex operators.

DeepONets. Lu *et al.* (Lu *et al.*, 2021a) introduced deep operator networks (DeepONets) as an efficient method to implement the theoretical insights from the universal approximation theorem described in Chen *et al.* (Chen and Chen, 1995). A DeepONet G_Θ consists of two NNs, namely, the branch and the trunk networks aimed at approximating the operator G (8.13). The branch network is responsible for encoding the input functions u from the parameter space $\mathcal{P}$, at predetermined sensor points $\{s_i\}_{i=0}^m$, while the trunk network encodes the domain geometry (the time-space coordinates), $\{\mathbf{y}_i\}_{i=0}^P$, which are used to evaluate the resulting solution $G_\Theta(u)(\mathbf{y})$. Specifically, the branch network takes as input the function $u \in \mathcal{P}$, represented in a discrete form as $[u(s_1), u(s_2), \dots, u(s_m)]$, and outputs a latent representation $[b_1, b_2, \dots, b_q]$. In parallel, the trunk network takes a time-space coordinate $\mathbf{y} \in I \times \Omega$ and generates a latent representation $[c_1, c_2, \dots, c_q]$. The final output of a DeepONet model is given by:

$$G_\Theta(u)(\mathbf{y}) = \sigma \left(\sum_{i=1}^q b_i c_i + c_0 \right). \tag{8.14}$$

where σ represents an output activation function chosen based on the specific problem, with the identity function being a common choice. Additionally, c_0 is a bias term added at the output layer to improve the generalization of the DeepONet model.

Remark 19 *The notation $G_\Theta(u)(\mathbf{y})$ denotes the evaluation of the surrogate model G_Θ at an input function $u \in \mathcal{P}$, assessed at the time-space coordinate $\mathbf{y} \in I \times \Omega$.*

The trunk network can be seen as a network responsible for constructing basis functions to represent the solution. The branch network can be regarded as a network tasked to compute the coefficients for these basis functions (Patel *et al.*, 2024). To evaluate the dot product between the latent representations from the branch and trunk networks, they both must be of the same output dimension q .

The training of a DeepONet model can be conducted either through a purely data-driven approach (Lu *et al.*, 2021a) or by incorporating physical constraints, resulting in what is known as a physics-informed DeepONet (Wang *et al.*, 2021b). The type of NN used for both the branch and the trunk depends on the situation at hand. In this study, we adopted a modified multilayer perceptron.

Appendix B. DeepONet settings for each numerical test

Table 8.5 DeepONet training setup for scenario (a).

Parameter	Non-parametric case	Parametric case
Number of samples	$N_{\text{samples}} = 100$	$N_{\text{samples}} = 100$
Sensors	$m = 20 \times 20$	$m = 20 \times 20$
Collocation points	$P = 11,500$	$P = 11,500$
Branch/Trunk architecture	4 layers $\times$ 50 neurons	5 layers $\times$ 100 neurons
Trainable parameters	60,900	152,000
Loss weights	$\alpha_{\text{IC}} = \alpha_{\text{BC}} = 10$	$\alpha_{\text{IC}} = \alpha_{\text{BC}} = 100$
Optimizer	Adam	Adam
Epochs	50,000	200,000
Batch size	10,000	10,000
Training time (A100)	0.26 h	2.95 h
Training time (V100)	0.39 h	3.43 h

Table 8.6 DeepONet training setup for Scenario (b). Architecture, optimizer, and training settings are identical across all scenarios: a 6-layer $\times$ 100-neuron network, 172,200 trainable parameters, Adam optimizer, 200,000 epochs, batch size 10,000, and loss weights $\alpha_{\text{IC}} = 1$, $\alpha_{\text{BC}} = 10$.

Parameter	Scenario 1	Scenario 2	Scenario 3
Training samples	100	500	500
Testing samples	50	100	100
Input type	Constant flux q_0	Pulse time $t_i \in (0, T)$	Time-dependent $q_0^{(i)}$
Sensors	100	1000	1000
Collocation points	10,500	10,500	20,000
Training time	3.18 h	4.37 h	5.08 h

Appendix C. Overview of the two studied scenarios

Here we present a visualization of the training data employed in each scenario. For comprehensive details on the results, refer to the corresponding sections. Figure 8.15 showcases selected samples from each scenario.

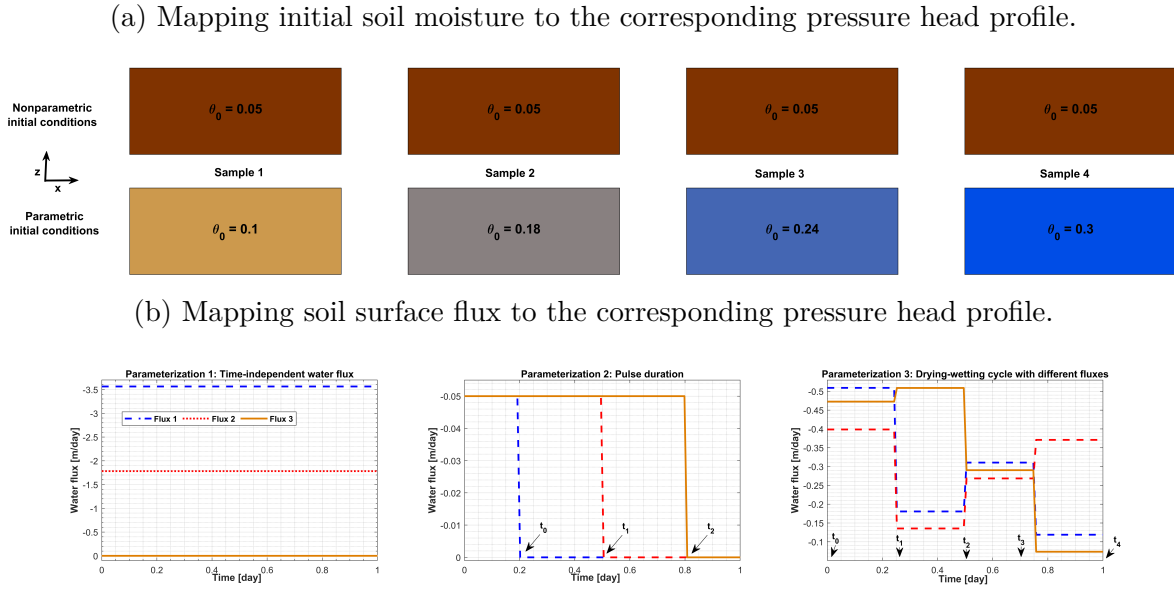


Figure 8.15 The two scenarios explored in this study: four snapshots for scenario (a) and three snapshots for scenario (b) used for training the DeepONet model.

Appendix D. Supplementary results

This section presents additional results. First, we plot the total loss of the DeepONet after training for each scenario. The results are depicted in Figure 8.16.

In the following discussion, we compare the computational costs of the FEM and DeepONet models. To ensure a fair comparison, both solvers were run on a local machine using only the CPU environment. We focus exclusively on the prediction phase for both FEM and DeepONet. For DeepONet, this corresponds to the stage after training is completed. For FEM, it corresponds to the point after selecting the time step, mesh size, and mesh structure, and after performing a time-space convergence analysis to ensure stable and converged solutions. Since these preparation steps belong to the offline stage, they are excluded from the runtime comparison. This approach allows for a fair assessment of the speed and efficiency of each method when making predictions.

We present results for scenario (a)-parametric parametric initial conditions-geometry 1 and loamy soil. The FEM solution was computed using a time step of $\Delta t = 10^{-2}$ and a 50×50 mesh grid. The predictions of the DeepONet are made on a uniform grid consisting of $100 \times 100 \times 100$ time-space coordinates. The results are depicted in Figure 8.17.

For each initial soil moisture condition, we solve the corresponding Richards equation using

the FEM solver and simultaneously predict the solution with the trained DeepONet model. Thus, if there are n initial soil moisture conditions, the Richards equation is solved n times with the FEM solver, while n forward passes are required for the DeepONet model. It is important to note that to ensure fairness, parallel prediction is not employed during the forward passes with DeepONet.

In terms of CPU time, DeepONet outperforms the FEM solver, a trend that becomes more pronounced with additional initial soil moisture conditions. The primary computational burden in the FEM solver arises from solving the linear system, which is repeated iteratively until convergence is met for each time step. In contrast, DeepONet requires only the initial soil moisture data and performs a forward pass based on optimized parameters from its training phase, as described in Algorithm 7, to generate predictions. A key distinction between the two solvers is that FEM operates sequentially in time, requiring computation from the beginning to the final time step, while DeepONet utilizes a time-space approach, predicting across the entire time-space domain.

In conclusion, the DeepONet is 52 times faster than the FEM solver for this test scenario.

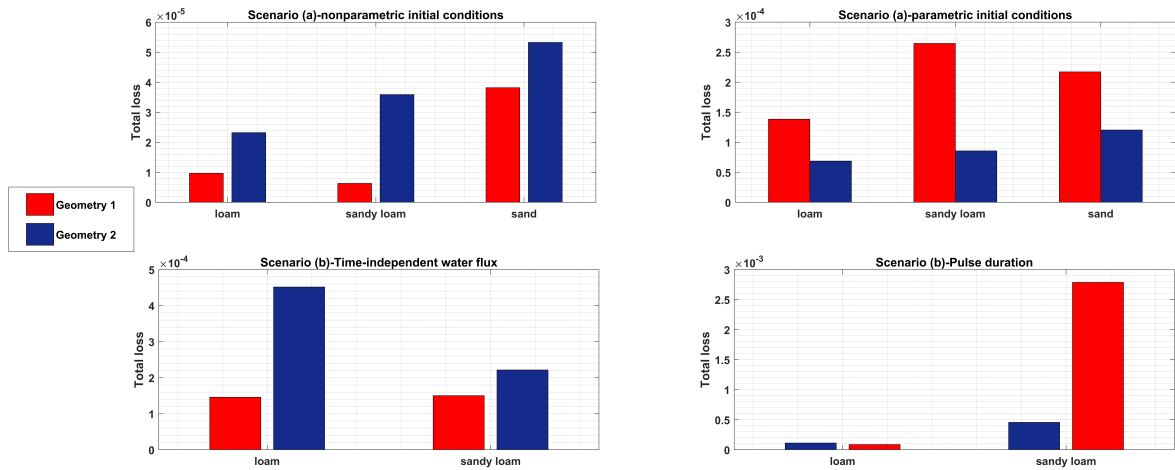


Figure 8.16 The total loss of DeepONet for four parametrizations used in both scenarios.

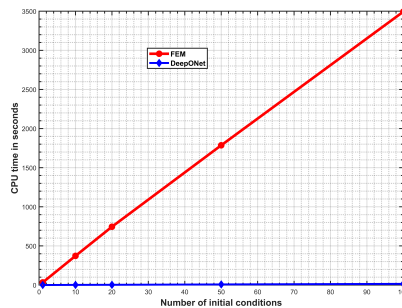


Figure 8.17 CPU time comparison of FEM and DeepONet solutions for Richards equation across varying initial soil moisture conditions.

CHAPTER 9 Conclusion

The current thesis project is devoted to the development of efficient predictive algorithms for modeling water flow and single or multi-species solute transport in unsaturated porous media. Two main infiltration models are considered: the Richards equation is adopted as the governing model for stable infiltration, while the fourth-order BCJ-R equation (Beljaid *et al.*, 2020) is employed to capture unstable infiltration phenomena such as fingering flow and saturation overshoot. These infiltration models are particularly challenging to solve due to their strong nonlinearity, steep solution gradients, and discontinuities arising in heterogeneous soils. To simulate solute transport, the infiltration models are coupled with advection–dispersion–reaction equations. These equations account for key biogeochemical processes such as first-order decay of nitrogen species, adsorption, immobilization, and denitrification, capturing the complexity of real-world fertilizer transport in unsaturated soils. The overarching goal of this thesis is to design computationally efficient and accurate numerical models capable of handling the multiphysics nature of this coupled system for forward simulations and for inverse modeling tasks involving noisy and limited data.

In Chapter 2, we developed an efficient FEM to simulate the coupled processes of water flow and solute transport in porous media. The proposed framework is based on a semi-implicit, second-order mixed FEM formulation that circumvents the need for iterative nonlinear solvers, thereby eliminating convergence issues commonly encountered when solving the Richards equation. The proposed numerical model was extended to account for both single-species and multi-species solute transport, incorporating relevant physical processes such as advection, dispersion, and first-order decay. To ensure its accuracy and reliability, the numerical scheme was rigorously validated against analytical solutions, benchmark numerical models, and experimental datasets. Beyond serving as a stand-alone predictive tool, the FEM solver plays a dual role in this thesis: it acts as a high-fidelity reference for assessing the accuracy of PINN-based models and functions as a data generator for training and validating inverse modeling approaches.

In Chapter 3, we presented a PINN model for solving the coupled system of water flow and solute transport in unsaturated porous media. The PINN framework was designed for a semi-supervised, data-free training regime, relying solely on initial and boundary condition data, while incorporating the governing partial differential equations as soft constraints in the loss function. This approach enables the model to infer physically consistent solutions without the need for labeled data across the domain. The predictive performance of the PINN model was systematically evaluated by comparing it to the high-fidelity FEM solver developed in Chapter 2. The results obtained demonstrated good agreement between the two models, confirming the reliability and potential of PINNs for forward modeling of infiltration and solute transport in porous media.

In Chapter 4, we present an extensive comparative study of multiple PINN strategies for modeling coupled water–solute transport processes. Both forward and inverse problems were addressed, with a particular emphasis on parameter identification from sparse and noisy measurements, reflecting realistic conditions encountered in vadose-zone hydrology. Among

the evaluated approaches, the sequential PINN framework emerged as the most effective, consistently delivering accurate predictions for both water flow and solute concentration dynamics. Its robustness was demonstrated across a diverse set of benchmark problems and further validated against experimental datasets, highlighting its suitability for data-limited, physics-constrained modeling tasks.

In Chapter 5, we extend the sequential PINN framework by integrating a transfer learning strategy. This approach leverages knowledge acquired from earlier components of the coupled PDE system, such as water flow solutions, to enhance the training of subsequent solute transport models. The incorporation of transfer learning significantly accelerated convergence and improved the stability of the training process. These benefits were particularly evident in multi-species solute transport scenarios, where the model demonstrated good performance in both forward simulations and inverse parameter estimation tasks, even under conditions of sparse and noisy observational data.

In Chapter ??, we generalize the sequential training PINN approach to a broader class of one-way coupled PDE systems. We introduced novel PINN architectures specifically designed to preserve the directional structure inherent in such systems. These architectures were evaluated on the water–multispecies solute transport problem as a representative case study. The proposed models demonstrated superior accuracy and training stability, particularly in scenarios involving multispecies solute transport. This work underscores the importance of aligning the neural network architecture with the underlying PDE structure when designing PINN models for complex multiphysics applications.

In Chapter 7, we focused on modeling unstable infiltration governed by the fourth-order BCJ-R equation. We demonstrated that standard PINN models are insufficient to capture saturation overshoot and other fine-scale features of this equation. A range of architectural and training enhancements were implemented, leading to a PINN model that successfully captured complex dynamics and agreed with both experimental and numerical references. Moreover, the model was shown to satisfy the entropy dissipation law in closed systems, offering a novel verification criterion for such physics-informed networks.

Finally, in Chapter 8, we proposed a physics-informed mesh-independent neural operator for multi-task learning of water infiltration dynamics in porous media. Trained in a self-supervised manner, this physics-informed DeepONet model learns the operator mapping from varying initial and boundary conditions to the full moisture distribution. Fine-tuning with sparse physical measurements improved generalization in out-of-distribution settings, making the model a viable surrogate for real-time predictive simulations.

Natural extensions of the current thesis can be listed as follows:

1. **Extension to 3D flow and transport.** The current thesis project addresses one- and two-dimensional problems. Extending the proposed predictive algorithms to fully 3D heterogeneous media is a natural next step to enable more realistic and field-scale simulations.
2. **Real-time data assimilation.** Integration of real-time data assimilation is a promising future direction. By incorporating real-time sensor measurements into PINNs or neural operator frameworks, the models can dynamically update their predictions to reflect evolving field conditions. This enables continuous estimation of soil moisture and

solute concentrations, supporting timely and data-informed decisions in agricultural water management and contamination control.

3. **Hybrid modeling frameworks.** Future work could explore hybrid approaches that combine physics-based models (e.g., finite element methods) with machine learning surrogates, leveraging the accuracy and reliability of the former and the speed and flexibility of the latter. Such approaches are particularly promising for complex scenarios such as unstable infiltration modeled by the BCJ-R equation, where traditional numerical methods face challenges due to the equation's fourth-order nature and the presence of steep gradients and multiple spatial-temporal scales. Hybrid frameworks could significantly enhance efficiency in large-scale inverse modeling or uncertainty quantification tasks while maintaining physical fidelity.
4. **Incorporation of stochasticity and uncertainty.** Future extensions could focus on developing physics-informed models that explicitly account for uncertainties in system parameters, initial and boundary conditions, and observational data. Approaches such as Bayesian PINNs or stochastic neural operators offer promising frameworks for capturing these uncertainties, thereby improving the robustness, interpretability, and reliability of model predictions in real-world applications.
5. **Multi-fidelity learning from heterogeneous data sources.** Future work could explore integrating multi-fidelity datasets, such as combining high-resolution laboratory measurements, in-situ soil probes, and lower-resolution remote sensing data, within a unified learning framework. This would allow models to effectively leverage the strengths of each data source while handling the inherent sparsity, noise, and heterogeneity present in real-world measurements.
6. **Application to coupled surface–subsurface systems.** Extending the proposed framework to incorporate surface flow dynamics, such as overland or shallow water flow, coupled with subsurface infiltration and solute transport would enable comprehensive modeling of integrated hydrological systems. This is particularly relevant for simulating real-world scenarios involving rainfall-runoff processes, irrigation, or contaminant transport across soil–water interfaces.
7. **Inclusion of biological and chemical interactions.** Future extensions could incorporate complex biogeochemical processes, such as microbial activity, root water and nutrient uptake, and precipitation/dissolution reactions, to more accurately model nutrient cycling, contaminant fate, and soil–plant interactions in unsaturated porous media.
8. **Parallel training for computational acceleration.** To scale PINN models to larger, higher-resolution domains, future work should incorporate parallel training strategies on multi-core CPUs and GPUs. Leveraging data-parallel and model-parallel techniques can significantly reduce training times and make real-time or near-real-time simulation feasible for complex, high-dimensional problems.

PUBLICATIONS

1. **Kamil, H.**, Beljadid, A., Soulaïmani, A., and Bourgault, Y. (2024). Semi-implicit schemes for modeling water flow and solute transport in unsaturated soils. *Advances in Water Resources*, 193, 104835.
2. **Kamil, H.**, Soulaïmani, A., and Beljadid, A. (2024). Physics-informed neural network vs finite element method for modeling coupled water and solute flow in unsaturated soils. *16th World Congress on Computational Mechanics and 4th Pan American Congress on Computational Mechanics*, Vancouver, Canada, 21–26 July.
3. **Kamil, H.**, Soulaïmani, A., and Beljadid, A. (2025). A comparative study of physics-informed neural network strategies for modeling water and nitrogen transport in unsaturated soils. *Journal of Hydrology*, 661, 133624.
4. **Kamil, H.**, Soulaïmani, A., and Beljadid, A. (2024). A transfer learning physics-informed deep learning framework for modeling multiple solute dynamics in unsaturated soils. *Computer Methods in Applied Mechanics and Engineering*, 431, 117276.
5. **Kamil, H.**, Soulaïmani, A., and Beljadid, A. (2025). A comprehensive analysis of physics-informed neural networks for solving one-way coupled problems. *Submitted*.
6. **Kamil, H.**, Soulaïmani, A., and Beljadid, A. (2025). Physics-informed neural networks for modeling saturation overshoot in porous media. *Submitted*.
7. **Kamil, H.**, Soulaïmani, A., and Beljadid, A. (2025). Physics-informed neural operators for efficient modeling of infiltration in porous media. *Journal of Computational Physics*, 538, 114156.

CONFERENCES & WORKSHOPS

Communications

1. **Kamil, H.**, Soulaïmani, A., and Beljadid, A. (2024). Physics-informed neural network vs finite element method for modeling coupled water and solute flow in unsaturated soils. *16th WCCM & 4th PanACM*, Vancouver, Canada, 21–26 July.
2. **Kamil, H.**, Soulaïmani, A., and Beljadid, A. (2025). Deep neural operators for inversion and surrogate modeling in unsaturated flow. *3rd Digital Twins in Engineering Conference (DTE 2025) & AICOMAS 2025*, Paris, France, 17–21 February.
3. **Kamil, H.**, Soulaïmani, A., and Beljadid, A. (2024). Advanced physics-informed deep learning for inverse problems of water flow and solute transport in unsaturated soils. *UM6P Doctoral Days*, Benguerir, Morocco, 23–26 September.
4. **Kamil, H.**, Soulaïmani, A., and Beljadid, A. (2024). Physics-informed neural networks for modeling water flow and solute transport in unsaturated soils. *International Conference on Research in Applied Mathematics and Computer Science (ICRAMCS)*, Marrakech, Morocco, 25–27 April.
5. **Kamil, H.**, Soulaïmani, A., and Beljadid, A. (2024). Efficient modeling of unsaturated flow in soils using physics-informed deep neural operators. *I2CEAI 2024*, Khouribga, Morocco, 27–28 September.
6. Soulaïmani, A., **Kamil, H.**, Mullins, M., Fahsi, A., and Beljadid, A. (2025). Advancements in PINNs for Fluid Mechanics: The PirateNet Architecture and Training Strategies. *Accuracy & Efficiency in Scientific Machine Learning Workshop, Organized by IVADO and CRM*, Montréal, Canada, 25–27 June.
7. **Kamil, H.**, Beljadid, A., Soulaïmani, A., and Bourgault, Y. (2025). Semi-implicit second-order mixed finite element methods for modeling infiltration and solute transport in soils. *The Third Joint SIAM/CAIMS Annual Meetings (AN25), Conference on Applied and Computational Discrete Algorithms (ACDA25)*, Montréal, Canada, July 28 – August 1.

Participations

1. The 16th World Congress on Computational Mechanics and 4th Pan American Congress on Computational Mechanics, Vancouver, Canada, July 21–26, 2024.
2. The Third Joint SIAM/CAIMS Annual Meetings (AN25), Conference on Applied and Computational Discrete Algorithms (ACDA25), Montréal, Canada, July 28–August 1, 2025.
3. The third IACM Digital Twins in Engineering Conference (DTE 2025) & first ECCOMAS Artificial Intelligence and Computational Methods in Applied Science (AICOMAS 2025), Paris, France, February 17–21, 2025.

4. Accuracy & Efficiency in Scientific Machine Learning Workshop, Organized by IVADO and CRM, Montréal, Canada, June 25–27, 2025.
5. University Mohammed VI Polytechnic Doctoral Days, Benguerir, Morocco, September 23–26, 2024.
6. The first INTERNATIONAL CONFERENCE ON COMPUTER ENGINEERING AND ARTIFICIAL INTELLIGENCE (I2CEAI) 2024, Khouribga, Morocco, September 27–28, 2024.
7. International Conference on Research in Applied Mathematics and Computer Science (ICRAMCS), Marrakech, Morocco, April 25–27, 2024.

REFERENCES

- Abadi, Martín and Barham, Paul and Chen, Jianmin and Chen, Zhifeng and Davis, Andy and Dean, Jeffrey and Devin, Matthieu and Ghemawat, Sanjay and Irving, Geoffrey and Isard, Michael and others (2016). TensorFlow: A system for large-scale machine learning. *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. 265–283.
- Aboutalebi, Mahyar and Allen, L Niel and Torres-Rua, Alfonso F and McKee, Mac and Coopmans, Calvin (2019). Estimation of soil moisture at different soil levels using machine learning techniques and unmanned aerial vehicle (UAV) multispectral imagery. *Autonomous air and ground sensing systems for agricultural optimization and phenotyping IV*. SPIE, vol. 11008, 216–226.
- Ahmad, Sajjad and Kalra, Ajay and Stephen, Haroon (2010). Estimating soil moisture using remote sensing data: A machine learning approach. *Advances in water resources*, 33(1), 69–80.
- Alhubail, Ali and He, Xupeng and AlSinan, Marwa and Kwak, Hyung and Hoteit, Hussein (2022). Extended physics-informed neural networks for solving fluid flow problems in highly heterogeneous media. *International Petroleum Technology Conference*. IPTC, D031S073R001.
- Almajid, Muhammad M and Abu-Al-Saud, Moataz O (2022). Prediction of porous media fluid flow using physics informed neural networks. *Journal of Petroleum Science and Engineering*, 208, 109205.
- Alt, Hans Wilhelm and Luckhaus, Stephan (1983). Quasilinear elliptic-parabolic differential equations. *Math. Z*, 183(3), 311–341.
- Amini, Danial and Haghighat, Ehsan and Juanes, Ruben (2022). Physics-informed neural network solution of thermo–hydro–mechanical processes in porous media. *Journal of Engineering Mechanics*, 148(11), 04022070.
- Anagnostopoulos, Sokratis J and Toscano, Juan Diego and Stergiopoulos, Nikolaos and Karniadakis, George Em (2024). Residual-based attention in physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421, 116805.
- Antonopoulos, Vassilis Z. (1993). Simulation of water and nitrogen dynamics in soils during wastewater applications by using a finite element model. *Water Resources Management*, 7(3), 237–251.
- Arbogast, Todd and Obeyesekere, Mandri and Wheeler, Mary F (1993). Numerical methods for the simulation of flow in root-soil systems. *SIAM journal on numerical analysis*, 30(6), 1677–1702.

- Ascher, Uri M and Ruuth, Steven J and Wetton, Brian TR (1995). Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3), 797–823.
- Autovino, Dario and Rallo, Giovanni and Provenzano, Giuseppe (2018). Predicting soil and plant water status dynamic in olive orchards under different irrigation systems with Hydrus-2D: Model performance and scenario analysis. *Agricultural water management*, 203, 225–235.
- Azizzadenesheli, Kamyar and Kovachki, Nikola and Li, Zongyi and Liu-Schiaffini, Miguel and Kossaifi, Jean and Anandkumar, Anima (2024). Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 1–9.
- Bailey, Ryan T and Morway, Eric D and Niswonger, Richard G and Gates, Timothy K (2013). Modeling variably saturated multispecies reactive groundwater solute transport with MODFLOW-UZF and RT3D. *Groundwater*, 51(5), 752–761.
- Bandai, Toshiyuki and Ghezzehei, Teamrat 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 Resources Research*, 57(2), e2020WR027642.
- Bandai, Toshiyuki and Ghezzehei, Teamrat A (2022). Forward and inverse modeling of water flow in unsaturated soils with discontinuous hydraulic conductivities using physics-informed neural networks with domain decomposition. *Hydrology and Earth System Sciences*, 26(16), 4469–4495.
- Basir, Shamsulhaq and Senocak, Inanc (2022). Physics and equality constrained artificial neural networks: application to forward and inverse problems with multi-fidelity data fusion. *Journal of Computational Physics*, 463, 111301.
- Baydin, Atilim Gunes and Pearlmutter, Barak A and Radul, Alexey Andreyevich and Siskind, Jeffrey Mark (2018). Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18, 1–43.
- Bear, Jacob (2013). *Dynamics of fluids in porous media*. Courier Corporation.
- Beljadid, Abdelaziz and Cueto-Felgueroso, Luis and Juanes, Ruben (2020). A continuum model of unstable infiltration in porous media endowed with an entropy function. *Advances in Water Resources*, 144, 103684.
- Beljadid, Abdelaziz and Hanini, Amine (2023). An efficient semi-implicit friction source term treatment for modeling overland flow. *Advances in Water Resources*, 173, 104391.
- Beljadid, Abdelaziz and LeFloch, Philippe (2017). A central-upwind geometry-preserving method for hyperbolic conservation laws on the sphere. *Communications in Applied Mathematics and Computational Science*, 12(1), 81–107.

- Beljadid, Abdelaziz and LeFloch, Philippe G and Mohammadian, Abdolmajid (2019). Late-time asymptotic behavior of solutions to hyperbolic conservation laws on the sphere. *Computer Methods in Applied Mechanics and Engineering*, 349, 285–311.
- Beljadid, Abdelaziz and Mohammadian, Abdolmajid and Charron, Martin and Girard, Claude (2014). Theoretical and numerical analysis of a class of semi-implicit semi-lagrangian schemes potentially applicable to atmospheric models. *Monthly Weather Review*, 142(12), 4458–4476.
- Beljadid, Abdelaziz and Mohammadian, Abdolmajid and Qiblawey, Hazim M (2013). An unstructured finite volume method for large-scale shallow flows using the fourth-order Adams scheme. *Computers & Fluids*, 88, 579–589.
- Bergamaschi, Luca and Putti, Mario (1999a). Mixed finite elements and Newton-type linearizations for the solution of Richards’ equation. *International Journal for Numerical Methods in Engineering*, 45(8), 1025–1046.
- Bergamaschi, Luca and Putti, Mario (1999b). Mixed finite elements and Newton-type linearizations for the solution of Richards’ equation. *International journal for numerical methods in engineering*, 45(8), 1025–1046.
- Berlin, M and Suresh Kumar, G (2018). Numerical modelling on sorption kinetics of nitrogen species in wastewater-applied agricultural field. *Applied Water Science*, 8, 1–16.
- Daan Bertels and Patrick Willems (2023). Physics-informed machine learning method for modelling transport of a conservative pollutant in surface water systems. *Journal of Hydrology*, 619, 129354.
- Bhatt, Pratyush and Kumar, Yash and Soulaïmani, Azzeddine (2023). Deep convolutional architectures for extrapolative forecasts in time-dependent flow problems. *Advanced Modeling and Simulation in Engineering Sciences*, 10(1), 17.
- Bhattacharya, Kaushik and Hosseini, Bamdad and Kovachki, Nikola B and Stuart, Andrew M (2020). Model reduction and neural networks for parametric PDEs. *arXiv preprint arXiv:2005.03180*.
- Bona, JL and Dougalis, VA and Karakashian, OA and McKinney, WR and Smith, FT (1995). Conservative, high-order numerical schemes for the generalized Korteweg—de Vries equation. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, 351(1695), 107–164.
- Boujoudar, Mohamed and Beljadid, Abdelaziz and Taik, Ahmed (2021a). Localized MQ-RBF meshless techniques for modeling unsaturated flow. *Engineering analysis with boundary elements*, 130, 109–123.
- Boujoudar, Mohamed and Beljadid, Abdelaziz and Taik, Ahmed (2021b). Modelling of unsaturated flow through porous media using meshless methods. *Canadian Society of Civil Engineering Annual Conference*. Springer, 565–576.

- Boujoudar, Mohamed and Beljadid, Abdelaziz and Taik, Ahmed (2023a). Localized RBF methods for modeling infiltration using the Kirchhoff-transformed Richards equation. *Engineering Analysis with Boundary Elements*.
- Boujoudar, Mohamed and Beljadid, Abdelaziz and Taik, Ahmed (2023b). Localized RBF methods for modeling infiltration using the Kirchhoff-transformed Richards equation. *Engineering Analysis with Boundary Elements*, 152, 259–276.
- Boujoudar, Mohamed and Beljadid, Abdelaziz and Taik, Ahmed (2024). Implicit EXP-RBF techniques for modeling unsaturated flow through soils with water uptake by plant roots. *Applied Numerical Mathematics*, Positive review.
- Bradbury, James and Frostig, Roy and Hawkins, Peter and Johnson, Matthew James and Leary, Chris and Maclaurin, Dougal and Necula, George and Paszke, Adam and VanderPlas, Jake and Wanderman-Milne, Skye and others (2018). JAX: composable transformations of Python+ NumPy programs.
- Brendel, Oliver (2021). The relationship between plant growth and water consumption: a history from the classical four elements to modern stable isotopes. *Annals of Forest Science*, 78(2), 47.
- Brooks, Royal Harvard and Corey, Arthur T (1966). Properties of porous media affecting fluid flow. *Journal of the irrigation and drainage division*, 92(2), 61–88.
- Buckingham, Edgar (1907). Studies on the movement of soil moisture. *US Dept. Agric. Bur. Soils Bull.*, 38.
- Byrd, Richard H and Lu, Peihuang and Nocedal, Jorge and Zhu, Ciyu (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5), 1190–1208.
- Cai, Shengze and Mao, Zhiping and Wang, Zhicheng and Yin, Minglang and Karniadakis, George Em (2021a). Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12), 1727–1738.
- Cai, Shengze and Wang, Zhicheng and Lu, Lu and Zaki, Tamer A and Karniadakis, George Em (2021b). DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, 436, 110296.
- Caiqiong, Yi and Jun, Fan (2016). Application of HYDRUS-1D model to provide antecedent soil water contents for analysis of runoff and soil erosion from a slope on the Loess Plateau. *Catena*, 139, 1–8.
- Cao, Qianying and Goswami, Somdatta and Karniadakis, George Em (2024). Laplace neural operator for solving differential equations. *Nature Machine Intelligence*, 6(6), 631–640.
- Cao, Wenbo and Zhang, Weiwei (2025). An analysis and solution of ill-conditioning in physics-informed neural networks. *Journal of Computational Physics*, 520, 113494.

- Casulli, Vincenzo and Zanolli, Paola (2010). A nested Newton-type algorithm for finite volume methods solving Richards' equation in mixed form. *SIAM Journal on Scientific Computing*, 32(4), 2255–2273.
- Celia, Michael A. and Bouloutas, Efthimios T. and Zarba, Rebecca L. (1990a). A general mass-conservative numerical solution for the unsaturated flow equation. *Water Resources Research*, 26(7), 1483–1496.
- Celia, Michael A. and Bouloutas, Efthimios T. and Zarba, Rebecca L. (1990b). A general mass-conservative numerical solution for the unsaturated flow equation. *Water resources research*, 26(7), 1483–1496.
- Chávez-Negrete, C and Domínguez-Mota, FJ and Santana-Quinteros, D (2018). Numerical solution of Richards' equation of water flow by generalized finite differences. *Computers and Geotechnics*, 101, 168–175.
- Chen, Jungang and Gildin, Eduardo and Killough, John E (2023a). Transfer learning-based physics-informed convolutional neural network for simulating flow in porous media with time-varying controls. *arXiv preprint arXiv:2310.06319*.
- Chen, Min and Willgoose, Garry R and Saco, Patricia M (2014). Spatial prediction of temporal soil moisture dynamics using HYDRUS-1D. *Hydrological processes*, 28(2), 171–185.
- Chen, Nanxi and Cui, Chuanjie and Ma, Rujin and Chen, Airong and Wang, Sifan (2025). Sharp-PINNs: staggered hard-constrained physics-informed neural networks for phase field modelling of corrosion. *arXiv preprint arXiv:2502.11942*.
- Chen, Shuai and Mao, Xiaomin and Barry, David Andrew and Yang, Jian (2019a). Model of crop growth, water flow, and solute transport in layered soil. *Agricultural Water Management*, 221, 160–174.
- Chen, Shuai and Mao, Xiaomin and Barry, David Andrew and Yang, Jian (2019b). Model of crop growth, water flow, and solute transport in layered soil. *Agricultural Water Management*, 221, 160–174.
- Chen, Tianping and Chen, Hong (1993). Approximations of continuous functionals by neural networks with application to dynamic systems. *IEEE Transactions on Neural networks*, 4(6), 910–918.
- Chen, Tianping and Chen, Hong (1995). Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4), 911–917.
- Chen, Tianqi and Goodfellow, Ian and Shlens, Jonathon (2015). Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*.

- Chen, Xinhai and Gong, Chunye and Wan, Qian and Deng, Liang and Wan, Yunbo and Liu, Yang and Chen, Bo and Liu, Jie (2021a). Transfer learning for deep neural network-based partial differential equations solving. *Advances in Aerodynamics*, 3(1), 1–14.
- Chen, Xiao-Xuan and Zhang, Pin and Yin, Zhen-Yu (2024). Physics-Informed neural network solver for numerical analysis in geoen지니어ing. *Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards*, 1–19.
- Chen, Yang and Xu, Yongfu and Wang, Lei and Li, Tianyi (2023b). Modeling water flow in unsaturated soils through physics-informed neural network with principled loss function. *Computers and Geotechnics*, 161, 105546.
- Chen, Zhao and Liu, Yang and Sun, Hao (2021b). Physics-informed learning of governing equations from scarce data. *Nature communications*, 12(1), 6136.
- Choubineh, Abouzar and Chen, Jie and Wood, David A and Coenen, Frans and Ma, Fei (2023). Fourier neural operator for fluid flow in small-shape 2D simulated porous media dataset. *Algorithms*, 16(1), 24.
- Chuang, Pi-Yueh and Barba, Lorena A (2022). Experience report of physics-informed neural networks in fluid simulations: pitfalls and frustration. *arXiv preprint arXiv:2205.14249*.
- Cote, Claire M and Bristow, Keith L and Charlesworth, Philip B and Cook, Freeman J and Thorburn, Peter J (2003). Analysis of soil wetting and solute transport in subsurface trickle irrigation. *Irrigation Science*, 22(3-4), 143–156.
- Cueto-Felgueroso, L and Juanes, R (2009). A phase field model of unsaturated flow. *Water resources research*, 45(10).
- Dastorani, Mohammad T and Heshmati, Mohammad and Sadeghzadeh, Mohammad A (2010). Evaluation of the efficiency of surface and subsurface irrigation in dryland environments. *Irrigation and Drainage: The journal of the International Commission on Irrigation and Drainage*, 59(2), 129–137.
- Davis, Timothy A (2004). Algorithm 832: UMFPACK V4. 3—an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software (TOMS)*, 30(2), 196–199.
- Dazzi, Susanna (2024). Physics-informed neural networks for the augmented system of shallow water equations with topography. *Water Resources Research*, 60(10), e2023WR036589.
- DeCaria, Victor and Schneier, Michael (2021). An embedded variable step IMEX scheme for the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 376, 113661.
- Dekhovitch, Aleksandr and Sluiter, Marcel HF and Tax, David MJ and Bessa, Miguel A (2025). iPINNs: Incremental learning for Physics-informed neural networks. *Engineering with Computers*, 41(1), 389–402.

- Delgado, AE and Sun, Da-Wen (2001). Heat and mass transfer models for predicting freezing processes—a review. *Journal of food engineering*, 47(3), 157–174.
- Depina, Ivan and Jain, Saket and Mar Valsson, Sigurdur and Gotovac, Hrvoje (2022). Application of physics-informed neural networks to inverse problems in unsaturated groundwater flow. *Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards*, 16(1), 21–36.
- Dharmaraja, Sohan (2007). *An analysis of the TR-BDF2 integration scheme*. Msc thesis, Massachusetts Institute of Technology.
- Diab, Waleed and Chaabi, Omar and Alkobaisi, Shayma and Awotunde, Abeeb and Al Kobaisi, Mohammed (2024). Learning generic solutions for multiphase transport in porous media via the flux functions operator. *Advances in Water Resources*, 183, 104609.
- Diaw, EB and Lehmann, Francois and Ackerer, Ph (2001). One-dimensional simulation of solute transfer in saturated–unsaturated porous media using the discontinuous finite elements method. *Journal of contaminant hydrology*, 51(3-4), 197–213.
- DiCarlo, David A (2004). Experimental measurements of saturation overshoot on infiltration. *Water Resources Research*, 40(4).
- DiCarlo, David A (2013). Stability of gravity-driven multiphase flow in porous media: 40 years of advancements. *Water Resources Research*, 49(8), 4531–4544.
- Diersch, H-JG and Perrochet, Pierre (1999). On the primary variable switching technique for simulating unsaturated-saturated flows. *Advances in Water Resources*, 23(3), 271–301.
- Difonzo, Fabio V and Pellegrino, Sabrina F (2024). Convergence analysis of a spectral numerical method for a peridynamic formulation of Richards’ equation. *Mathematics and Computers in Simulation*, 223, 219–228.
- Donnelly, James and Daneshkhah, Alireza and Abolfathi, Soroush (2024). Physics-informed neural networks as surrogate models of hydrodynamic simulators. *Science of the Total Environment*, 912, 168814.
- Dougill, Andy J and Heathwaite, A Louise and Thomas, David SG (1998). Soil water movement and nutrient cycling in semi-arid rangeland: vegetation change and system resilience. *Hydrological Processes*, 12(3), 443–459.
- Ebrahimian, Hamed and Liaghat, Abdolmajid and Parsinejad, Masoud and Playán, Enrique and Abbasi, Fariborz and Navabian, Maryam (2013). Simulation of 1D surface and 2D subsurface water flow and nitrate transport in alternate and conventional furrow fertigation. *Irrigation Science*, 31, 301–316.
- Eivazi, Hamidreza and Tahani, Mojtaba and Schlatter, Philipp and Vinuesa, Ricardo (2022). Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations. *Physics of Fluids*, 34(7).

- Eliassi, Mehdi and Glass, Robert J (2001). On the continuum-scale modeling of gravity-driven fingers in unsaturated porous media: The inadequacy of the Richards equation with standard monotonic constitutive relations and hysteretic equations of state. *Water Resources Research*, 37(8), 2019–2035.
- Eliassi, Mehdi and Glass, Robert J (2002). On the porous-continuum modeling of gravity-driven fingers in unsaturated materials: Extension of standard theory with a hold-back-pile-up effect. *Water Resources Research*, 38(11), 16–1.
- Elmorsy, Mohamed and El-Dakhakhni, Wael and Zhao, Benzhong (2023). Rapid Permeability Upscaling of Digital Porous Media via Physics-Informed Neural Networks. *Water Resources Research*, 59(12), e2023WR035064.
- Eshaghi, Mohammad Sadegh and Anitescu, Cosmin and Thombre, Manish and Wang, Yizheng and Zhuang, Xiaoying and Rabczuk, Timon (2025). Variational physics-informed neural operator (VINO) for solving partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 437, 117785.
- Essaid, Hedef I and Bekins, Barbara A and Cozzarelli, Isabelle M (2015). Organic contaminant transport and fate in the subsurface: Evolution of knowledge and understanding. *Water Resources Research*, 51(7), 4861–4902.
- Ethier, Marc and Bourgault, Yves (2008). Semi-implicit time-discretization schemes for the bidomain model. *SIAM Journal on Numerical Analysis*, 46(5), 2443–2468.
- Eymard, Robert and Gutnic, Michaël and Hilhorst, Danielle (1999). The finite volume method for Richards equation. *Computational Geosciences*, 3, 259–294.
- Fahs, Marwan and Younes, Anis and Lehmann, François (2009). An easy and efficient combination of the Mixed Finite Element Method and the Method of Lines for the resolution of Richards' Equation. *Environmental Modelling & Software*, 24(9), 1122–1126.
- Fanaskov, Vladimir Sergeevich and Oseledets, Ivan V (2023). Spectral neural operators. *Doklady Mathematics*. Springer, vol. 108, S226–S232.
- FAO (2021). FAOSTAT: Statistical Reports of Plant Production. <http://www.fao.org/faostat/en/#data/QC>. Accessed on 27 January 2022.
- Faroughi, Salah A and Pawar, Nikhil M and Fernandes, Celio and Raissi, Maziar and Das, Subasish and Kalantari, Nima K and Kourosh Mahjour, Seyed (2024). Physics-guided, physics-informed, and physics-encoded neural networks and operators in scientific computing: Fluid and solid mechanics. *Journal of Computing and Information Science in Engineering*, 24(4), 040802.
- Faroughi, Salah A and Soltanmohammadi, Ramin and Datta, Pingki and Mahjour, Seyed Kourosh and Faroughi, Shirko (2023). Physics-informed neural networks with periodic activation functions for solute transport in heterogeneous porous media. *Mathematics*, 12(1), 63.

- Farthing, Matthew W and Ogden, Fred L (2017). Numerical solution of Richards' equation: A review of advances and challenges. *Soil Science Society of America Journal*, 81(6), 1257–1269.
- Feddes, Reinder A and Kowalik, Piotr and Kolinska-Malinka, Krystina and Zaradny, Henryk (1976). Simulation of field water uptake by plants using a soil water dependent root extraction function. *Journal of Hydrology*, 31(1-2), 13–26.
- Fengnan, Liu and Fukumoto, Yasuhide and Zhao, Xiaopeng (2020). A linearized finite difference scheme for the Richards equation under variable-flux boundary conditions. *Journal of Scientific Computing*, 83(1), 16.
- Feyen, Jan and Jacques, Diederik and Timmerman, Anthony and Vanderborght, Jan (1998). Modelling water flow and solute transport in heterogeneous soils: A review of recent approaches. *Journal of Agricultural Engineering Research*, 70(3), 231–256.
- Floridi, Luciano and Chiriatti, Massimo (2020). GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30, 681–694.
- Fraces, Cedric G and Tchelepi, Hamdi (2021). Physics informed deep learning for flow and transport in porous media. *SPE Reservoir Simulation Conference?* SPE, D011S006R002.
- Franco, Nicola Rares and Manzoni, Andrea and Zunino, Paolo (2023). Mesh-informed neural networks for operator learning in finite element spaces. *Journal of Scientific Computing*, 97(2), 35.
- Fuks, Olga and Tchelepi, Hamdi A (2020). Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1).
- Gärdenäs, AI and Hopmans, JW and Hanson, BR and Šimůnek, J (2005). Two-dimensional modeling of nitrate leaching for various fertigation scenarios under micro-irrigation. *Agricultural water management*, 74(3), 219–242.
- Gardner, WR (1958). Some steady-state solutions of the unsaturated moisture flow equation with application to evaporation from a water table. *Soil science*, 85(4), 228–232.
- Glass, Robert J and Steenhuis, Tammie S and Parlange, J-Yves (1989). Mechanism for finger persistence in homogeneous, unsaturated, porous media: Theory and verification. *Soil Science*, 148(1), 60–70.
- Glorot, Xavier and Bengio, Yoshua (2010a). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 249–256.
- Glorot, Xavier and Bengio, Yoshua (2010b). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 249–256. ISSN: 1938-7228.

- Gong, Xuezi and Zha, Yuanyuan (2025). Physics-informed neural network with enhanced parameter-state coupling for inverse modeling of unsaturated flow in heterogeneous soils. *Water Resources Research*, 61(9), e2025WR040040.
- Goswami, Somdatta and Anitescu, Cosmin and Chakraborty, Souvik and Rabczuk, Timon (2020). Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics*, 106, 102447.
- Goswami, Somdatta and Kontolati, Katiana and Shields, Michael D and Karniadakis, George Em (2022a). Deep transfer operator learning for partial differential equations under conditional shift. *Nature Machine Intelligence*, 4(12), 1155–1164.
- Goswami, Somdatta and Yin, Minglang and Yu, Yue and Karniadakis, George Em (2022b). A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*, 391, 114587.
- Grossmann, Tamara G and Komorowska, Urszula Julia and Latz, Jonas and Schönlieb, Carola-Bibiane (2023). Can physics-informed neural networks beat the finite element method? *arXiv preprint arXiv:2302.04107*.
- Quan Guo and Yue Zhao and Chunhui Lu and Jian Luo (2023). High-dimensional inverse modeling of hydraulic tomography by physics informed neural network (HT-PINN). *Journal of Hydrology*, 616, 128828.
- Haghighat, Ehsan and Adeli, Mohammad Hesam and Mousavi, S Mohammad and Juanes, Ruben (2025). STONet: A neural operator for modeling solute transport in micro-cracked reservoirs. *Advances in Water Resources*, 105046.
- Haghighat, Ehsan and Amini, Danial and Juanes, Ruben (2022). Physics-informed neural network simulation of multiphase poroelasticity using stress-split sequential training. *Computer Methods in Applied Mechanics and Engineering*, 397, 115141.
- Hanson, Blaine R and Šimůnek, Jirka and Hopmans, Jan W (2006). Evaluation of urea-ammonium-nitrate fertigation with drip irrigation using numerical modeling. *Agricultural water management*, 86(1-2), 102–113.
- Harandi, Ali and Moeineddin, Ahmad and Kaliske, Michael and Reese, Stefanie and Rezaei, Shahed (2024). Mixed formulation of physics-informed neural networks for thermo-mechanically coupled systems and heterogeneous domains. *International Journal for Numerical Methods in Engineering*, 125(4), e7388.
- Haruzi, P and Moreno, Z (2023a). Modeling Water Flow and Solute Transport in Unsaturated Soils Using Physics-Informed Neural Networks Trained With Geoelectrical Data. *Water Resources Research*, 59(6), e2023WR034538.
- Haruzi, P and Moreno, Z (2023b). Modeling Water Flow and Solute Transport in Unsaturated Soils Using Physics-Informed Neural Networks Trained With Geoelectrical Data. [Dataset]. Zenodo. <https://doi.org/10.5281/zenodo.7558746>.

- Haruzi, P. and Moreno, Z. (2023c). Modeling Water Flow and Solute Transport in Unsaturated Soils Using Physics-Informed Neural Networks Trained With Geoelectrical Data. *Water Resources Research*, 59(6), e2023WR034538.
- Haverkamp, Roland and Vauclin, Michel and Touma, Jaoudat and Wierenga, PJ and Vachaud, Georges (1977). A comparison of numerical simulation models for one-dimensional infiltration. *Soil Science Society of America Journal*, 41(2), 285–294.
- He, Junyan and Kushwaha, Shashank and Park, Jaewan and Koric, Seid and Abueidda, Diab and Jasiuk, Iwona (2024). Sequential Deep Operator Networks (S-DeepONet) for predicting full-field solutions under time-dependent loads. *Engineering Applications of Artificial Intelligence*, 127, 107258.
- He, QiZhi and Barajas-Solano, David and Tartakovsky, Guzel and Tartakovsky, Alexandre M (2020). Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Advances in Water Resources*, 141, 103610.
- Hecht, Frédéric (2012). New development in FreeFem++. *Journal of numerical mathematics*, 20(3-4), 251–266.
- Hornik, Kurt (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2), 251–257.
- Hornik, Kurt and Stinchcombe, Maxwell and White, Halbert (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5), 359–366.
- Howard, Amanda and Fu, Yucheng and Stinis, Panos (2024). A multifidelity approach to continual learning for physical systems. *Machine Learning: Science and Technology*, 5(2), 025042.
- Hu, Zheyuan and Shukla, Khemraj and Karniadakis, George Em and Kawaguchi, Kenji (2024). Tackling the curse of dimensionality with physics-informed neural networks. *Neural Networks*, 176, 106369.
- Huang, Pan and Leng, Yifei and Lian, Cheng and Liu, Honglai (2024). Porous-DeepONet: Learning the Solution Operators of Parametric Reactive Transport Equations in Porous Media. *Engineering*, 39, 94–103.
- Huang, Shudong and Feng, Wentao and Tang, Chenwei and He, Zhenan and Yu, Caiyang and Lv, Jiancheng (2025). Partial differential equations meet deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.
- Illiano, Davide and Pop, Iuliu Sorin and Radu, Florin Adrian (2021). Iterative schemes for surfactant transport in porous media. *Computational geosciences*, 25, 805–822.
- Im, Daniel Jiwoong and Tao, Michael and Branson, Kristin (2016). An empirical analysis of the optimization of deep network loss surfaces. *arXiv preprint arXiv:1612.04010*.

- Jacot, Arthur and Gabriel, Franck and Hongler, Clément (2018). Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31.
- Jagtap, Ameya D and Karniadakis, George Em (2020). Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5).
- Jagtap, Ameya D and Kawaguchi, Kenji and Karniadakis, George Em (2020). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404, 109136.
- Jin, Pengzhan and Meng, Shuai and Lu, Lu (2022). MIONet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6), A3490–A3514.
- Kaluarachchi, Jagath J and Parker, JC (1988). Finite element model of nitrogen species transformation and transport in the unsaturated zone. *Journal of Hydrology*, 103(3-4), 249–274.
- Kamil, Hamza and Beljadid, Abdelaziz and Soulaïmani, Azzeddine and Bourgault, Yves (2024a). Semi-implicit schemes for modeling water flow and solute transport in unsaturated Soils. *Advances in Water Resources*, Under Review.
- Kamil, Hamza and Beljadid, Abdelaziz and Soulaïmani, Azzeddine and Bourgault, Yves (2024b). Semi-implicit schemes for modeling water flow and solute transport in unsaturated soils. *Advances in Water Resources*, 104835.
- Hamza Kamil and Azzeddine Soulaïmani and Abdelaziz Beljadid (2024c). A transfer learning physics-informed deep learning framework for modeling multiple solute dynamics in unsaturated soils. *Computer Methods in Applied Mechanics and Engineering*, 431, 117276.
- Kamil, Hamza and Soulaïmani, Azzeddine and Beljadid, Abdelaziz (2024d). A transfer learning physics-informed deep learning framework for modeling multiple solute dynamics in unsaturated soils. *Computer Methods in Applied Mechanics and Engineering*, 431, 117276.
- Kamil, Hamza and Soulaïmani, Azzeddine and Beljadid, Abdelaziz (2025). A comparative study of physics-informed neural network strategies for modeling water and nitrogen transport in unsaturated soils. *Journal of Hydrology*, 133624.
- Kamil, Hamza and SSoulaïmani, Azzeddine and Beljadid, Abdelaziz (2024e). Physics-informed neural network vs finite element method for modeling coupled water and solute flow in unsaturated soils. *16th World Congress on Computational Mechanics and 4th Pan American Congress on Computational Mechanics, 21-26 July, Vancouver, Canada*.
- Kandelous, Maziar M and Simunek, Jiri and Van Genuchten, M Th and Malek, Keyvan (2011). Soil water content distributions between two emitters of a subsurface drip irrigation system. *Soil Science Society of America Journal*, 75(2), 488–497.

- Kandelous, Maziar M and Šimůnek, Jiri (2010). Numerical simulations of water movement in a subsurface drip irrigation system under field and laboratory conditions using HYDRUS-2D. *Agricultural Water Management*, 97(7), 1070–1076.
- Karandish, Fatemeh and Šimůnek, Jirka (2016). A comparison of numerical and machine-learning modeling of soil water content with limited input data. *Journal of Hydrology*, 543, 892–909.
- Karandish, Fatemeh and Šimůnek, Jirka (2019). A comparison of the HYDRUS (2D/3D) and SALTMed models to investigate the influence of various water-saving irrigation strategies on the maize water footprint. *Agricultural Water Management*, 213, 809–820.
- Karjoun, Hasan and Beljadid, Abdelaziz and LeFloch, Philippe G (2022). A structure-preserving algorithm for surface water flows with transport processes. *Advances in Computational Mathematics*, 48(1), 7.
- Karniadakis, George Em and Kevrekidis, Ioannis G and Lu, Lu and Perdikaris, Paris and Wang, Sifan and Yang, Liu (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3(6), 422–440.
- Kavetski, D. and Binning, P. and Sloan, S.W. (2001). Adaptive time stepping and error control in a mass conservative numerical solution of the mixed form of Richards equation. *Advances in Water Resources*, 24(6), 595–605.
- Kavetski, Dmitri and Binning, Philip and Sloan, Scott W (2002). Noniterative time stepping schemes with adaptive truncation error control for the solution of Richards equation. *Water Resources Research*, 38(10), 29–1.
- Keita, Sana and Beljadid, Abdelaziz and Bourgault, Yves (2021a). Efficient second-order semi-implicit finite element method for fourth-order nonlinear diffusion equations. *Computer Physics Communications*, 258, 107588.
- Keita, Sana and Beljadid, Abdelaziz and Bourgault, Yves (2021b). Implicit and semi-implicit second-order time stepping methods for the Richards equation. *Advances in Water Resources*, 148, 103841.
- Keita, Sana and Beljadid, Abdelaziz and Bourgault, Yves (2021c). Implicit and semi-implicit second-order time stepping methods for the Richards equation. *Advances in Water Resources*, 148, 103841.
- Keita, Sana and Beljadid, Abdelaziz and Bourgault, Yves (2021d). Mass-conservative and positivity preserving second-order semi-implicit methods for high-order parabolic equations. *Journal of Computational Physics*, 440, 110427.
- Khadijeh, Mahmoud and Cerqueglini, Veronica and Kasbergen, Cor and Erkens, Sandra and Varveri, Aikaterini (2025). Multistage physics informed neural network for solving coupled multiphysics problems in material degradation and fluid dynamics. *Engineering with Computers*, 1–31.

- Kharazmi, Ehsan and Zhang, Zhongqiang and Karniadakis, George Em (2019). Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873*.
- Kharazmi, Ehsan and Zhang, Zhongqiang and Karniadakis, George Em (2021). hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374, 113547.
- Khashaei, Farshad and Behmanesh, Javad and Rezaverdinejad, Vahid and Azad, Nasrin (2024). Field evaluation and numerical simulation of water and nitrate transport in subsurface drip irrigation of corn using HYDRUS-2D. *Irrigation Science*, 42(2), 327–352.
- Kim, Jungeun and Lee, Kookjin and Lee, Dongeun and Jhin, Sheo Yon and Park, Noseong (2021). DPM: a novel training method for physics-informed neural networks in extrapolation. *Proceedings of the AAAI Conference on Artificial Intelligence*. vol. 35, 8146–8154.
- Kim, Jihoon and Tchelepi, Hamdi A and Juanes, Ruben (2011). Stability and convergence of sequential methods for coupled flow and geomechanics: Fixed-stress and fixed-strain splits. *Computer Methods in Applied Mechanics and Engineering*, 200(13-16), 1591–1606.
- Kim, Jung-Hoon and Guo, Xuejun and Park, Hung-Suck (2008). Comparison study of the effects of temperature and free ammonia concentration on nitrification and nitrite accumulation. *Process Biochemistry*, 43(2), 154–160.
- Kingma, Diederik P and Ba, Jimmy (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Koto, Toshiyuki (2008). IMEX Runge–Kutta schemes for reaction–diffusion equations. *Journal of Computational and Applied Mathematics*, 215(1), 182–195.
- Kovachki, Nikola and Lanthaler, Samuel and Mishra, Siddhartha (2021). On universal approximation and error bounds for Fourier neural operators. *Journal of Machine Learning Research*, 22(290), 1–76.
- Krishnapriyan, Aditi and Gholami, Amir and Zhe, Shandian and Kirby, Robert and Mahoney, Michael W (2021). Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34, 26548–26560.
- Lagaris, Isaac E and Likas, Aristidis and Fotiadis, Dimitrios I (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5), 987–1000.
- Lan, Peng and Su, Jingjing and Zhang, Sheng (2024a). Surrogate modeling for unsaturated infiltration via the physics and equality-constrained artificial neural networks. *Journal of Rock Mechanics and Geotechnical Engineering*, 16(6), 2282–2295.
- Lan, Peng and Su, Jingjing and Zhu, Shuairun and Huang, Jinsong and Zhang, Sheng (2024b). Reconstructing unsaturated infiltration behavior with sparse data via physics-informed deep learning. *Computers and Geotechnics*, 168, 106162.

- Lanthaler, Samuel and Mishra, Siddhartha and Karniadakis, George E (2022). Error estimates for deepONets: A deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1), tnac001.
- Laubscher, Ryno (2021). Simulation of multi-species flow and heat transfer using physics-informed neural networks. *Physics of Fluids*, 33(8).
- Lehmann, F and Ackerer, PH (1998). Comparison of iterative methods for improved solutions of the fluid flow equation in partially saturated porous media. *Transport in porous media*, 31, 275–292.
- Lehmann, François and Fahs, Marwan and Alhubail, Ali and Hoteit, Hussein (2023). A mixed pressure-velocity formulation to model flow in heterogeneous porous media with physics-informed neural networks. *Advances in Water Resources*, 181, 104564.
- Leverett, MoC (1941). Capillary behavior in porous solids. *Transactions of the AIME*, 142(01), 152–169.
- Li, Changli and Han, Zheng and Li, Yange and Li, Ming and Wang, Weidong and Dou, Jie and Xu, Linrong and Chen, Guangqi (2023). Physical information-fused deep learning model ensembled with a subregion-specific sampling method for predicting flood dynamics. *Journal of Hydrology*, 620, 129465.
- Li, H and Farthing, MW and Dawson, CN and Miller, CT (2007). Local discontinuous Galerkin approximations to Richards' equation. *Advances in Water Resources*, 30(3), 555–575.
- Li, J and Zhang, J and Rao, M (2005). Modeling of water flow and nitrate transport under surface drip fertigation. *Transactions of the ASAE*, 48(2), 627–637.
- Li, Kai-Qi and Yin, Zhen-Yu and Zhang, Ning and Li, Jian (2024a). A PINN-based modelling approach for hydromechanical behaviour of unsaturated expansive soils. *Computers and Geotechnics*, 169, 106174.
- Li, Xiuneng and Wang, Hong and Qin, Shuhong and Lin, Lin and Wang, Xin and Cornelis, Wim (2024b). Evaluating ensemble learning in developing pedotransfer functions to predict soil hydraulic properties. *Journal of Hydrology*, 640, 131658.
- Li, Yong and Šimůnek, Jirka and Zhang, Zhentin and Jing, Longfei and Ni, Lixiao (2015). Evaluation of nitrogen balance in a direct-seeded-rice field experiment using Hydrus-1D. *Agricultural Water Management*, 148, 213–222.
- Li, Zongyi and Kovachki, Nikola and Azizzadenesheli, Kamyar and Liu, Burigede and Bhattacharya, Kaushik and Stuart, Andrew and Anandkumar, Anima (2020). Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.
- List, Florian and Radu, Florin A (2016). A study on iterative methods for solving Richards' equation. *Computational Geosciences*, 20(2), 341–353.

- Liu, Kun and Huang, Guanhua and Xu, Xu and Xiong, Yunwu and Huang, Quanzhong and Šimůnek, Jirka (2019). A coupled model for simulating water flow and solute transport in furrow irrigation. *Agricultural Water Management*, 213, 792–802.
- Liu, Yi and Ao, Chang and Zeng, Wenzhi and Srivastava, Amit Kumar and Gaiser, Thomas and Wu, Jingwei and Huang, Jiesheng (2021). Simulating water and salt transport in subsurface pipe drainage systems with HYDRUS-2D. *Journal of Hydrology*, 592, 125823.
- Llanas, Bernardo and Lantarón, Sagrario and Sáinz, Francisco J (2008). Constructive approximation of discontinuous functions by neural networks. *Neural Processing Letters*, 27(3), 209–226.
- Loy, KC and Bourgault, Y (2017). On efficient high-order semi-implicit time-stepping schemes for unsteady incompressible Navier-Stokes equations. *Computers & Fluids*, 148, 166–184.
- Lu, Lu and Jin, Pengzhan and Pang, Guofei and Zhang, Zhongqiang and Karniadakis, George Em (2021a). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3), 218–229.
- Lu, Lu and Meng, Xuhui and Mao, Zhiping and Karniadakis, George Em (2021b). DeepXDE: A deep learning library for solving differential equations. *SIAM review*, 63(1), 208–228.
- Ma, Yaoyao and Xu, Xiaoyu and Yan, Shuai and Ren, Zhuoxiang (2022). A preliminary study on the resolution of electro-thermal multi-physics coupling problem using physics-informed neural network (PINN). *Algorithms*, 15(2), 53.
- Machado, Gaspar J. and Pereira, Rui M.S. and Clain, Stéphane and Araújo, Nuno and Lopes, Sofia O. (2022). A new stabilised scheme for the Richards’ equation with evapotranspiration. *Groundwater for Sustainable Development*, 17, 100736.
- Manzini, Gianmarco and Ferraris, Stefano (2004). Mass-conservative finite volume methods on 2-D unstructured grids for the Richards’ equation. *Advances in Water Resources*, 27(12), 1199–1215.
- Marinoschi, Gabriela (2006). *Functional approach to nonlinear models of water flow in soils*, vol. 21. Springer.
- Mattey, Revanth and Ghosh, Susanta (2022). A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390, 114474.
- McClenny, Levi D and Braga-Neto, Ulisses M (2023). Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474, 111722.
- McNab Jr, WW and Narasimhan, TN (1993). A multiple species transport model with sequential decay chain interactions in heterogeneous subsurface environments. *Water resources research*, 29(8), 2737–2746.

- Merdasi, Arshia and Ebrahimi, Saman and Yang, Xiang and Kunz, Robert (2023). Physics Informed Neural Network application on mixing and heat transfer in combined electroosmotic-pressure driven flow. *Chemical Engineering and Processing-Process Intensification*, 193, 109540.
- Millington, RJ and Quirk, JP (1961). Permeability of porous solids. *Transactions of the Faraday Society*, 57, 1200–1207.
- Mishra, Siddhartha and Molinaro, Roberto (2020). Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating a class of inverse problems for PDEs. *arXiv preprint arXiv:2007.01138*.
- Mishra, Siddhartha and Molinaro, Roberto (2023). Estimates on the generalization error of physics-informed neural networks for approximating PDEs. *IMA Journal of Numerical Analysis*, 43(1), 1–43.
- Mitchell, Robert J and Mayer, Alex S (1998). A numerical model for transient-hysteretic flow and solute transport in unsaturated porous media. *Journal of contaminant hydrology*, 30(3-4), 243–264.
- Moghadas, Davood and Badorreck, Annika (2019). Machine learning to estimate soil moisture from geophysical measurements of electrical conductivity. *Near Surface Geophysics*, 17(2), 181–195.
- Monaco, Simone and Apiletti, Daniele (2023). Training physics-informed neural networks: One learning to rule them all? *Results in Engineering*, 18, 101023.
- Morellos, Antonios and Pantazi, Xanthoula-Eirini and Moshou, Dimitrios and Alexandridis, Thomas and Whetton, Rebecca and Tziotzios, Georgios and Wiebensohn, Jens and Bill, Ralf and Mouazen, Abdul M (2016). Machine learning based prediction of soil total nitrogen, organic carbon and moisture content by using VIS-NIR spectroscopy. *Biosystems Engineering*, 152, 104–116.
- Moreno, Ziv and Paster, Amir (2018). Prediction of pollutant remediation in a heterogeneous aquifer in Israel: Reducing uncertainty by incorporating lithological, head and concentration data. *Journal of Hydrology*, 564, 651–666.
- Morianou, Giasemi and Kourgialas, Nektarios N and Karatzas, George P (2023). A review of HYDRUS 2D/3D applications for simulations of water dynamics, root uptake and solute transport in tree crops under drip irrigation. *Water*, 15(4), 741.
- Mualem, Yechezkel (1976). A new model for predicting the hydraulic conductivity of unsaturated porous media. *Water resources research*, 12(3), 513–522.
- Muhammad, Ihsan and Lv, Ju Zhi and Yang, Li and Ahmad, Shakeel and Farooq, Saqib and Zeeshan, Muhammad and Zhou, Xun Bo (2022). Low irrigation water minimizes the nitrate nitrogen losses without compromising the soil fertility, enzymatic activities and maize growth. *BMC Plant Biology*, 22(1), 159.

- Mullins, Mathieu and Kamil, Hamza and Fahsi, Adil and Soulaïmani, Azzeddine (2025). Physics-informed neural networks for solving moving interface flow problems using the level set approach. *Physics of Fluids*, 37(10).
- Nakamura, Yo and Shiratori, Suguru and Takagi, Ryota and Sutoh, Michihiro and Sugihara, Iori and Nagano, Hideaki and Shimano, Kenjiro (2022). Physics-informed neural network applied to surface-tension-driven liquid film flows. *International Journal for Numerical Methods in Fluids*, 94(9), 1359–1378.
- Nazari, Luis Fernando and Camponogara, Eduardo and Seman, Laio Oriel (2022). Physics-informed neural networks for modeling water flows in a river channel. *IEEE Transactions on Artificial Intelligence*, 5(3), 1001–1015.
- Nielsen, DR and Th. Van Genuchten, M and Biggar, JW (1986). Water flow and solute transport processes in the unsaturated zone. *Water resources research*, 22(9S), 89S–108S.
- Niu, Jie and Xu, Wei and Qiu, Han and Li, Shan and Dong, Feifei (2023). 1-D coupled surface flow and transport equations revisited via the physics-informed neural network approach. *Journal of Hydrology*, 625, 130048.
- Nunez, Andres and Misra, Siddharth and Falola, Yusuf (2025). Transfer learning for geological carbon storage forecasting using neural operator. *Advances in Water Resources*, 199, 104948.
- Ohara, Y and Moteki, D and Muramatsu, S and Hayasaka, K and Yasuda, H (2024). Physics-informed neural networks for inversion of river flow and geometry with shallow water model. *Physics of Fluids*, 36(10).
- Pan, Sinno Jialin and Yang, Qiang (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10), 1345–1359.
- Pang, Min (2024). Efficient river hydrodynamics modelling in realistic river systems using a Fourier neural operator-based network. *Journal of Hydrology*, 637, 131345.
- Paniconi, Claudio and Aldama, Alvaro A and Wood, Eric F (1991). Numerical evaluation of iterative and noniterative methods for the solution of the nonlinear Richards equation. *Water Resources Research*, 27(6), 1147–1163.
- Patel, Dhruv and Ray, Deep and Abdelmalik, Michael RA and Hughes, Thomas JR and Oberai, Assad A (2024). Variationally mimetic operator networks. *Computer Methods in Applied Mechanics and Engineering*, 419, 116536.
- Perrin, Jérôme and Mascré, Cédric and Pauwels, Hélène and Ahmed, Shakeel (2011). Solute recycling: an emerging threat to groundwater quality in southern India? *Journal of Hydrology*, 398(1-2), 144–154.
- Pham, Khanh and Kim, Dongku and Le, Canh V and Won, Jongmuk (2023). Machine learning-based pedotransfer functions to predict soil water characteristics curves. *Transportation Geotechnics*, 42, 101052.

- Poch-Massegú, R and Jiménez-Martínez, Joaquín and Wallis, KJ and de Cartagena, F Ramírez and Candela, L (2014). Irrigation return flow and nitrate leaching under different crops and irrigation methods in Western Mediterranean weather conditions. *Agricultural Water Management*, 134, 1–13.
- Pop, Iuliu Sorin and Schweizer, Ben (2011). Regularization schemes for degenerate Richards equations and outflow conditions. *Mathematical Models and Methods in Applied Sciences*, 21(08), 1685–1712.
- Porporato, A and D’odorico, P and Laio, F and Rodriguez-Iturbe, I (2003). Hydrologic controls on soil carbon and nitrogen cycles. I. Modeling scheme. *Advances in water resources*, 26(1), 45–58.
- Provenzano, Giuseppe (2007). Using HYDRUS-2D simulation model to evaluate wetted soil volume in subsurface drip irrigation systems. *Journal of Irrigation and Drainage Engineering*, 133(4), 342–349.
- Qadem, Zohair and Tayfur, Gökmen and Kankal, Murat (2025). In-depth analysis of drought trend in semiarid Saïs Plateau and Middle Atlas region in Morocco. *Acta Geophysica*, 1–17.
- Quezada, Cristhian R and Clement, T Prabhakar and Lee, Kang-Kun (2004). Generalized solution to multi-dimensional multi-species transport equations coupled with a first-order reaction network involving distinct retardation factors. *Advances in water resources*, 27(5), 507–520.
- Radu, Florin Adrian and Bause, Marcus and Prechtel, A and Attinger, Sabine (2008). A mixed hybrid finite element discretization scheme for reactive transport in porous media. *Numerical Mathematics and Advanced Applications: Proceedings of ENUMATH 2007, the 7th European Conference on Numerical Mathematics and Advanced Applications, Graz, Austria, September 2007*. Springer, 513–520.
- Radu, Florin A and Pop, Iuliu Sorin (2010). Newton method for reactive solute transport with equilibrium sorption in porous media. *Journal of computational and applied mathematics*, 234(7), 2118–2127.
- Radu, Florin A and Pop, Iuliu Sorin and Attinger, Sabine (2010). Analysis of an Euler implicit-mixed finite element scheme for reactive solute transport in porous media. *Numerical Methods for Partial Differential Equations: An International Journal*, 26(2), 320–344.
- Radu, Florin A and Pop, Iuliu Sorin and Knabner, Peter (2006). Newton—type methods for the mixed finite element discretization of some degenerate parabolic equations. *Numerical Mathematics and Advanced Applications: Proceedings of ENUMATH 2005, the 6th European Conference on Numerical Mathematics and Advanced Applications Santiago de Compostela, Spain, July 2005*. Springer, 1192–1200.
- Rahaman, Nasim and Baratin, Aristide and Arpit, Devansh and Draxler, Felix and Lin, Min and Hamprecht, Fred and Bengio, Yoshua and Courville, Aaron (2019). On the spectral bias of neural networks. *International conference on machine learning*. PMLR, 5301–5310.

- Raissi, Maziar and Perdikaris, Paris and Karniadakis, George 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.
- Raj, Mayank and Kumbhar, Pramod and Annabattula, Ratna Kumar (2021). Physics-informed neural networks for solving thermo-mechanics problems of functionally graded material. *arXiv preprint arXiv:2111.10751*.
- Rajput, TBS and Patel, Neelam (2006). Water and nitrate movement in drip-irrigated onion under fertigation and irrigation treatments. *Agricultural water management*, 79(3), 293–311.
- Raonic, Bogdan and Molinaro, Roberto and Rohner, Tobias and Mishra, Siddhartha and de Bezenac, Emmanuel (2023). Convolutional neural operators. *ICLR 2023 workshop on physics for machine learning*.
- Rasht-Behesht, Majid and Huber, Christian and Shukla, Khemraj and Karniadakis, George Em (2022). Physics-informed neural networks (PINNs) for wave propagation and full wave-form inversions. *Journal of Geophysical Research: Solid Earth*, 127(5), e2021JB023120.
- Rathore, Pratik and Lei, Weimu and Frangella, Zachary and Lu, Lu and Udell, Madeleine (2024). Challenges in training PINNs: A loss landscape perspective. *arXiv preprint arXiv:2402.01868*.
- Rezaei, Shahed and Harandi, Ali and Moeineddin, Ahmad and Xu, Bai-Xiang and Reese, Stefanie (2022). A mixed formulation for physics-informed neural networks as a potential solver for engineering problems in heterogeneous domains: Comparison with finite element method. *Computer Methods in Applied Mechanics and Engineering*, 401, 115616.
- Richards, L. A. (1931). Capillary conduction of liquids through porous mediums. *Physics*, 1(5), 318–333.
- Ritzema, HP and others (1994). Subsurface flow to drains. *Drainage principles and applications*, 16, 263–304.
- Robinson, DA and Binley, Andrew and Crook, N and Day-Lewis, FD and Ferré, TPA and Grauch, VJS and Knight, R and Knoll, M and Lakshmi, V and Miller, R and others (2008). Advancing process-based watershed hydrological research using near-surface geophysics: A vision for, and review of, electrical and magnetic geophysical methods. *Hydrological Processes: An International Journal*, 22(18), 3604–3635.
- Roy, Arunabha M and Bose, Rikhi and Sundararaghavan, Veera and Arróyave, Raymundo (2023). Deep learning-accelerated computational framework based on Physics Informed Neural Network for the solution of linear elasticity. *Neural Networks*, 162, 472–489.
- Russo, David (2012a). Numerical analysis of solute transport in variably saturated bimodal heterogeneous formations with mobile–immobile-porosity. *Advances in water resources*, 47, 31–42.

- Russo, David (2012b). Numerical analysis of solute transport in variably saturated bimodal heterogeneous formations with mobile-immobile-porosity. *Advances in Water Resources*, 47, 31–42.
- Russo, David and Kurtzman, Daniel (2019). Using desalinated water for irrigation: Its effect on field scale water flow and contaminant transport under cropped conditions. *Water*, 11(4), 687.
- Russo, David and Zaidel, Jacob and Laufer, Asher (1998). Numerical analysis of flow and transport in a three-dimensional partially saturated heterogeneous soil. *Water Resources Research*, 34(6), 1451–1468.
- Russo, David and Zaidel, Jacob and Laufer, Asher (2001). Numerical analysis of flow and transport in a combined heterogeneous vadose zone-groundwater system. *Advances in Water Resources*.
- Saccon, Pierpaolo (2018). Water for agriculture, irrigation management. *Applied soil ecology*, 123, 793–796.
- Sahimi, Muhammad (2024). Physics-informed and data-driven discovery of governing equations for complex phenomena in heterogeneous media. *Physical Review E*, 109(4), 041001.
- Sander, GC and Braddock, RD (2005). Analytical solutions to the transient, unsaturated transport of water and contaminants through horizontal porous media. *Advances in Water Resources*, 28(10), 1102–1111.
- Sander, GC and Parlange, J-Y and Kühnel, V and Hogarth, WL and Lockington, D and O’kane, JPJ (1988). Exact nonlinear solution for constant flux infiltration. *Journal of Hydrology*, 97(3-4), 341–346.
- Schaap, Marcel G and Leij, Feike J and Van Genuchten, Martinus Th (2001). Rosetta: A computer program for estimating soil hydraulic parameters with hierarchical pedotransfer functions. *Journal of hydrology*, 251(3-4), 163–176.
- Schaeffer, Hayden (2017). Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197), 20160446.
- Schrefler, BA and D’Alpaos, Luigi and Zhan, XY and Simoni, Luciano (1994). Pollutant transport in deforming porous media. *European Journal of Mechanics. A, Solids;(France)*, 13(4).
- Selim, HM and Iskandar, IK (1981). MODELING NITROGEN TRANSPORT AND TRANSFORMATIONS IN SOILS: 1. THEORETICAL CONSIDERATIONS: 1. *Soil Science*, 131(4), 233–241.
- Shafeeq, Poo Madhathil and Aggarwal, Pramila and Krishnan, Prameela and Rai, Vikas and Pramanik, Pragati and Das, Tapas Kumar (2020). Modeling the temporal distribution of water, ammonium-N, and nitrate-N in the root zone of wheat using HYDRUS-2D under conservation agriculture. *Environmental Science and Pollution Research*, 27, 2197–2216.

- Shahraiyini, Hamid Taheri and Ataie-Ashtiani, Behzad (2012). Mathematical forms and numerical schemes for the solution of unsaturated flow equations. *Journal of Irrigation and Drainage Engineering*, 138(1), 63–72.
- Shin, Yeonjong and Darbon, Jerome and Karniadakis, George Em (2020). On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. *arXiv preprint arXiv:2004.01806*.
- Singh, Ajay and Panda, Sudhindra N and Uzokwe, Veronica NE and Krause, Peter (2019). An assessment of groundwater recharge estimation techniques for sustainable resource management. *Groundwater for Sustainable Development*, 9, 100218.
- Slodicka, Marián (2002). A robust and efficient linearization scheme for doubly nonlinear and degenerate parabolic problems arising in flow in porous media. *SIAM journal on scientific computing*, 23(5), 1593–1614.
- Song, Wenxiang and Shi, Liangsheng and Hu, Xiaolong and Wang, Yanling and Wang, Lijun (2023). Reconstructing the Unsaturated Flow Equation From Sparse and Noisy Data: Leveraging the Synergy of Group Sparsity and Physics-Informed Deep Learning. *Water Resources Research*, 59(5), e2022WR034122.
- Sonntag, Matthias and Munz, Claus-Dieter (2014). Shock capturing for discontinuous Galerkin methods using finite volume subcells. *Finite Volumes for Complex Applications VII-Elliptic, Parabolic and Hyperbolic Problems: FVCA 7, Berlin, June 2014*. Springer, 945–953.
- Soulaïmani, Azzeddine and Feng, Zhengkun and Ali, Amin Ben Haj (2005). Solution techniques for multi-physics problems with application to computational nonlinear aeroelasticity. *Nonlinear Analysis: Theory, Methods & Applications*, 63(5-7), e1585–e1595.
- Srivastava, Rajesh and Yeh, T-C Jim (1992). A three-dimensional numerical model for water flow and transport of chemically reactive solute through porous media under variably saturated conditions. *Advances in water resources*, 15(5), 275–287.
- Stevens, David and Power, Henry (2011). A scalable and implicit meshless RBF method for the 3D unsteady nonlinear Richards equation with single and multi-zone domains. *International journal for numerical methods in engineering*, 85(2), 135–163.
- Stuyt, LCPM and Dierickx, W (2006). Design and performance of materials for subsurface drainage systems in agriculture. *Agricultural Water Management*, 86(1-2), 50–59.
- Suh, Hyoung Suk and Song, Jun Young and Kim, Yejin and Yu, Xiong and Choo, Jinhyun (2024). Data-driven discovery of interpretable water retention models for deformable porous media. *Acta Geotechnica*, 19(6), 3821–3835.
- Sukumar, Natarajan and Srivastava, Ankit (2022). Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389, 114333.

- Taccari, Maria Luisa and Wang, He and Goswami, Somdatta and De Florio, Mario and Nuttall, Jonathan and Chen, Xiaohui and Jimack, Peter K (2024). Developing a cost-effective emulator for groundwater flow modeling using deep neural operators. *Journal of Hydrology*, 630, 130551.
- Tancik, Matthew and Srinivasan, Pratul and Mildenhall, Ben and Fridovich-Keil, Sara and Raghavan, Nithin and Singhal, Utkarsh and Ramamoorthi, Ravi and Barron, Jonathan and Ng, Ren (2020). Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., vol. 33, 7537–7547.
- Tang, Hesheng and Liao, Yangyang and Yang, Hu and Xie, Liyu (2022). A transfer learning-physics informed neural network (TL-PINN) for vortex-induced vibration. *Ocean Engineering*, 266, 113101.
- Tarbiyati, Homayoon and Nemati Saray, Behzad (2023). Weight initialization algorithm for physics-informed neural networks using finite differences. *Engineering with Computers*, 1–17.
- Tartakovsky, Alexandre M and Marrero, C Ortiz and Perdikaris, Paris and Tartakovsky, Guzel D and Barajas-Solano, David (2020). Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems. *Water Resources Research*, 56(5), e2019WR026731.
- Toutlini, Nour-eddine and Beljadid, Abdelaziz and Soulaïmani, Azzeddine (2024). Analysis of second-order temporal schemes for modeling flow-solute transport in unsaturated porous media. *arXiv preprint arXiv:2404.03603*.
- Toutlini, Nour-Eddine and Beljadid, Abdelaziz and Soulaïmani, Azzeddine (2025a). A semi-implicit second-order temporal scheme for solving the pressure head-based form of Richards' and advection-dispersion equations. *Computers & Mathematics with Applications*, 187, 106–131.
- Toutlini, Nour-Eddine and Beljadid, Abdelaziz and Soulaïmani, Azzeddine (2025b). A semi-implicit second-order temporal scheme for solving the pressure head-based form of Richards' and advection-dispersion equations. *Computers & Mathematics with Applications*, 187, 106–131.
- Tracy, Fred T (2006). Clean two-and three-dimensional analytical solutions of Richards' equation for testing numerical solvers. *Water Resources Research*, 42(8).
- Vallejo-Gomez, David and Osorio, Marisol and Hincapie, Carlos A (2023). Smart irrigation systems in agriculture: A systematic review. *Agronomy*, 13(2), 342.
- Van Cleve, K and Yarie, J (1986). Interaction of temperature, moisture, and soil chemistry in controlling nutrient cycling and ecosystem development in the taiga of Alaska. *Forest ecosystems in the Alaskan taiga: a synthesis of structure and function*, Springer. 160–189.

- Van Duyn, CJ and Peletier, LA (1982). Nonstationary filtration in partially saturated porous media. *Archive for Rational Mechanics and Analysis*, 78, 173–198.
- Van Genuchten, M Th (1980). A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil science society of America journal*, 44(5), 892–898.
- Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N and Kaiser, Łukasz and Polosukhin, Illia (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- Vemuri, Sai Karthikeya and Büchner, Tim and Denzler, Joachim (2024). Estimating Soil Hydraulic Parameters for Unsaturated Flow Using Physics-Informed Neural Networks. *International Conference on Computational Science*. Springer, 338–351.
- Vyas, Nikhil and Morwani, Depen and Zhao, Rosie and Kwun, Mujin and Shapira, Itai and Brandfonbrener, David and Janson, Lucas and Kakade, Sham (2024). Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*.
- Wan, Heyang and Qi, Hongwei and Shang, Songhao (2023). Estimating soil water and salt contents from field measurements with time domain reflectometry using machine learning algorithms. *Agricultural Water Management*, 285, 108364.
- Wang, Chunying and Boithias, Laurie and Ning, Zigong and Han, Yuping and Sauvage, Sabine and Sánchez-Pérez, José-Miguel and Kuramochi, Kanta and Hatano, Ryusuke (2017). Comparison of Langmuir and Freundlich adsorption equations within the SWAT-K model for assessing potassium environmental losses at basin scale. *Agricultural Water Management*, 180, 205–211.
- Wang, Sifan and Bhartari, Ananyae Kumar and Li, Bowen and Perdikaris, Paris (2025). Gradient Alignment in Physics-informed Neural Networks: A Second-Order Optimization Perspective. *arXiv preprint arXiv:2502.00604*.
- Wang, Sifan and Li, Bowen and Chen, Yuhan and Perdikaris, Paris (2024a). PirateNets: Physics-informed Deep Learning with Residual Adaptive Networks. *arXiv preprint arXiv:2402.00326*.
- Wang, Sifan and Sankaran, Shyam and Perdikaris, Paris (2022a). Respecting causality is all you need for training physics-informed neural networks.
- Wang, Sifan and Sankaran, Shyam and Perdikaris, Paris (2024b). Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421, 116813.
- Wang, Sifan and Sankaran, Shyam and Wang, Hanwen and Perdikaris, Paris (2023a). An expert’s guide to training physics-informed neural networks. *arXiv preprint arXiv:2308.08468*.
- Wang, Sifan and Teng, Yujun and Perdikaris, Paris (2021a). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5), A3055–A3081.

- Wang, Sifan and Wang, Hanwen and Perdikaris, Paris (2021b). Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science advances*, 7(40), eabi8605.
- Wang, Sifan and Wang, Hanwen and Perdikaris, Paris (2022b). Improved architectures and training algorithms for deep operator networks. *Journal of Scientific Computing*, 92(2), 35.
- Wang, Sifan and Wang, Hanwen and Seidman, Jacob H. and Perdikaris, Paris (2022c). Random Weight Factorization Improves the Training of Continuous Neural Representations. ArXiv:2210.01274 [cs].
- Wang, Sifan and Yu, Xinling and Perdikaris, Paris (2022d). When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449, 110768.
- Wang, Yanling and Shi, Liangsheng and Hu, Xiaolong and Song, Wenxiang and Wang, Lijun (2023b). Multiphysics-Informed Neural Networks for Coupled Soil Hydrothermal Modeling. *Water Resources Research*, 59(1), e2022WR031960.
- Wight, Colby L and Zhao, Jia (2020). Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542*.
- Wu, Chenxi and Zhu, Min and Tan, Qinyang and Kartha, Yadhu and Lu, Lu (2023). A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403, 115671.
- Xiong, Yunwu (2014). Flow of water in porous media with saturation overshoot: A review. *Journal of Hydrology*, 510, 353–362.
- Xu, Chen and Cao, Ba Trung and Yuan, Yong and Meschke, Günther (2023a). Transfer learning based physics-informed neural networks for solving inverse problems in engineering structures under different loading scenarios. *Computer Methods in Applied Mechanics and Engineering*, 405, 115852.
- Xu, Qingsong and Shi, Yilei and Bamber, Jonathan and Tuo, Ye and Ludwig, Ralf and Zhu, Xiao Xiang (2023b). Physics-aware machine learning revolutionizes scientific paradigm for machine learning and process-based hydrology. *arXiv preprint arXiv:2310.05227*.
- Yan, Xia and Lin, Jingqi and Wang, Sheng and Zhang, Zhao and Liu, Piyang and Sun, Shuyu and Yao, Jun and Zhang, Kai (2024). Physics-informed neural network simulation of two-phase flow in heterogeneous and fractured porous media. *Advances in Water Resources*, 189, 104731.
- Yang, Liu and Meng, Xuhui and Karniadakis, George Em (2021). B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics*, 425, 109913.

- Ye, Qiang and Huang, Zijie and Zheng, Qiang and Zeng, Lingzao (2025). Predicting water movement in unsaturated soil using physics-informed deep operator networks. *Advances in Water Resources*, 105001.
- Younes, Anis and Hoteit, Hussein and Helmig, Rainer and Fahs, Marwan (2022). A robust upwind mixed hybrid finite element method for transport in variably saturated porous media. *Hydrology and Earth System Sciences*, 26(20), 5227–5239.
- Zha, Yuanyuan and Yang, Jinzhong and Shi, Liangsheng and Song, Xuehang (2013). Simulating one-dimensional unsaturated flow in heterogeneous soils with water content-based Richards equation. *Vadose Zone Journal*, 12(2), 1–13.
- Zha, Yuanyuan and Yang, Jinzhong and Zeng, Jicai and Tso, Chak-Hau M. and Zeng, Wenzhi and Shi, Liangsheng (2019). Review of numerical solution of Richardson–Richards equation for variably saturated flow in soils. *WIREs Water*, 6(5).
- Zhan, Yang and Guo, Zhilin and Yan, Bicheng and Chen, Kewei and Chang, Zhenbo and Babovic, Vladan and Zheng, Chunmiao (2024). Physics-informed identification of PDEs with LASSO regression, examples of groundwater-related equations. *Journal of Hydrology*, 638, 131504.
- Zhang, Ruiyang and Liu, Yang and Sun, Hao (2020). Physics-informed multi-LSTM networks for metamodeling of nonlinear structures. *Computer Methods in Applied Mechanics and Engineering*, 369, 113226.
- Zhang, Xiaoping and Zhu, Yan and Wang, Jing and Ju, Lili and Qian, Yingzhi and Ye, Ming and Yang, Jinzhong (2022). GW-PINN: A deep learning algorithm for solving groundwater flow equations. *Advances in Water Resources*, 165, 104243.
- Zhao Zhang and Xia Yan and Piyang Liu and Kai Zhang and Renmin Han and Sheng Wang (2023). A physics-informed convolutional neural network for the simulation and prediction of two-phase Darcy flows in heterogeneous porous media. *Journal of Computational Physics*, 477, 111919.
- Zhou, Zhao and Ansems, Nienke and Torfs, Paul (2015). A global assessment of nitrate contamination in groundwater. *International Groundwater Resources Assessment Center. Internship report*, 4.
- Zhu, Min and Zhang, Handi and Jiao, Anran and Karniadakis, George Em and Lu, Lu (2023). Reliable extrapolation of deep neural operators informed by physics or sparse observations. *Computer Methods in Applied Mechanics and Engineering*, 412, 116064.
- Zhuang, Fuzhen and Qi, Zhiyuan and Duan, Keyu and Xi, Dongbo and Zhu, Yongchun and Zhu, Hengshu and Xiong, Hui and He, Qing (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1), 43–76.
- Šimůnek, J and Van Genuchten, M Th (1996). Estimating unsaturated soil hydraulic properties from tension disc infiltrometer data by numerical inversion. *Water Resources Research*, 32(9), 2683–2696.

Šimůnek, Jirka and van Genuchten, Martinus Th (2008). Modeling nonequilibrium flow and transport processes using HYDRUS. *Vadose zone journal*, 7(2), 782–797.

Šimůnek, Jirka and Van Genuchten, M Th and Sejna, M (2005). The HYDRUS-1D software package for simulating the one-dimensional movement of water, heat, and multiple solutes in variably-saturated media. *University of California-Riverside Research Reports*, 3, 1–240.

Šimůnek, Jiri and Van Genuchten, Martinus Th and Sejna, Miroslav (2016). Recent developments and applications of the HYDRUS computer software packages. *Vadose Zone Journal*, 15(7), vzj2016–04.

Šimůnek, JIRKA and Van Genuchten, M Th and Sejna, M and others (2006). The HYDRUS software package for simulating two-and three-dimensional movement of water, heat, and multiple solutes in variably-saturated media. *Technical manual, version, 1*, 241.