Interpretation of Special Core Analysis using Finite Volume Graph Networks and Operator Learning

Roman Manasipov[1,*], *Bettina* Jenei[1]
[1]TU Clausthal, Institute of Subsurface Energy Systems, Reservoir Engineering Department, 38678 Clausthal-Zellerfeld, Germany

Abstract. Interpreting core flooding experiments to extract relative permeability and capillary pressure functions remains a fundamental challenge in Special Core Analysis (SCAL). Existing analytical and semi-analytical methods are restricted by simplifying assumptions or applicability conditions that limit the range of interpretable data. Numerical history-matching approaches offer greater generality but are computationally demanding and lack a direct functional mapping between observations and rock-fluid properties. This study presents a neural network-based framework for SCAL interpretation that addresses these limitations. The approach combines two complementary architectures: a Finite Volume Graph Network (FVGN), which encodes multiphase flow equations within a graph-based message-passing structure that preserves local conservation properties, and a Deep Operator Network (DeepONet), which learns parametric mappings from input functions to solution fields, providing efficient surrogates for repeated forward evaluations during inversion. We focus on recovering relative permeability curves from observed pressure drops and produced phase volumes, assuming capillary pressure is independently measured. The relative permeabilities are parameterized using B-splines, which naturally accommodate monotonicity and convexity constraints through conditions on the spline coefficients. These physical constraints are incorporated directly into the training objective. By leveraging the differentiability of neural network surrogates, the framework enables gradient-based inversion that significantly accelerates interpretation compared to conventional simulation-based workflows. This computational efficiency makes the approach particularly attractive for systematic sensitivity and uncertainty analyses, offering a practical tool for more comprehensive SCAL interpretation.

1 Introduction

Traditionally, Special Core Analysis (SCAL) has relied on analytical or empirical models to analyze and interpret experimental data and estimate the relative permeability and capillary pressure functions. For a more accurate estimation, numerical modelling is also commonly applied. However, with advancements in machine learning techniques, there is an emerging trend in utilizing neural networks for modelling physical systems. Primarily, physics-informed neural networks (PINN) and operator learning networks such as Deep Operator Networks (DeepONet) [1], Fourier Neural Operators (FNO) [2], and Graph Neural Networks [3] are among those that are used directly for solving Ordinary Differential Equations (ODE) and Partial Differential Equations (PDE). More specialized Lagrangian Neural Networks [4] have been applied for modelling Lagrangian systems on generalized coordinates. Another architecture commonly helpful in modelling physical systems is the Residual Neural Network (ResNet) [5], which is useful for working with time series and, in general, for time integration. U-Net [6] networks are effective for capturing multiscale phenomena and are commonly utilized in diffusion models, which are well known for their ability to produce high-quality images.

The main advantage of these methods is their ability to model complex physical systems by honouring physical laws, while simultaneously representing the solution as a differentiable function of its input parameters. This allows sensitivity studies or inverse modelling to be performed efficiently without the need to run simulations each time a new evaluation needs to be made. The inverse problem can be solved using optimization with gradient-based methods, which can significantly reduce the number of iterations required to converge to the desired solution.

The present study employs a physics-informed neural network method combined with DeepONet (PI-DeepONet) to solve inverse problems for interpreting relative permeabilities. The representation of relative permeabilities is achieved using B-splines, which offer a smooth functional form and are well suited for the DeepONet approach. To improve the quality of the physical solution, the training process is enhanced by the data loss using simulation results from DuMuX [7], which is not necessary for physics-informed machine learning models, but it is generally improves the accuracy of the trained model and speeds up the training process.

The PI-DeepONet network on the basis of Finite Volume Graph Network (FVGN) with Weak-Entropy Physics Informed Loss (WE-PINNs) presents a physics-based, differentiable solution that accelerates history matching compared to conventional simulation-based approaches and enhances the accuracy of interpretations relative to analytical solutions with restrictive assumptions. This makes it an excellent tool for conducting comprehensive sensitivity analysis of the final interpretations.

* Corresponding author: roman.manasipov@gmail.com

2 Methodology

Special core analysis aims to interpret the relative permeability and capillary functions from the measured pressure drop across the core and the produced volumes of individual phases over time. Additionally, when X-ray or gamma-ray scans are available, the in situ saturation monitoring (ISSM) can be performed [8]. In general, the better the monitoring performed in the experiment, the better and easier the inverse modelling can be performed with less uncertainty. In this sense, performing microfluidic experiments, given all the differences, can be beneficial for the continuous monitoring of the flooding timeline using a high-resolution camera. Interpretation of the relative permeability and capillary functions from observed data requires solving an inverse problem, in which neural networks can fit naturally. The formulated inverse problem can be solved numerically or using neural networks.

In the following section, the mathematical formulation of the coreflooding problem for the unsteady-state experiment is described. This is followed by a description of the relative permeability and capillary pressure functions using B-splines. Subsequently, the neural network architecture used in the inverse modelling is described.

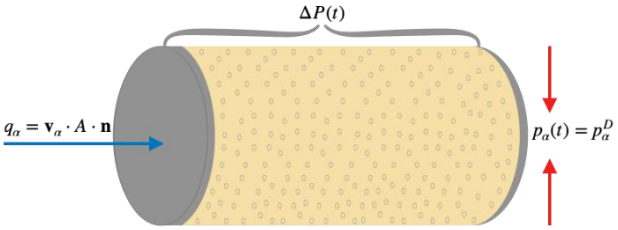

Fig. 1. Core flood modeling setup.

2.1 Mathematical model of unsteady state experiment

An unsteady state coreflooding experiment for an oil-water system can be modelled under the assumption of an incompressible immiscible flow. The following system of equations describes it:

$$\phi \frac{\partial \rho_\alpha S_\alpha}{\partial t} - \nabla \cdot \left(\rho_\alpha \frac{k_{r\alpha}(S_\alpha)}{\mu_\alpha} \boldsymbol{K} \cdot (\nabla p_\alpha - \rho_\alpha \boldsymbol{g}) \right) - \rho_\alpha q_\alpha = 0 \quad (1)$$

$$p_c(S_w) = p_w - p_n \quad (2)$$

$$\mathbf{v}_\alpha = -\frac{k_{r\alpha}(S_\alpha)}{\mu_\alpha} \boldsymbol{K} \cdot (\nabla p_\alpha - \rho_\alpha g) \quad (3)$$

$$\mathbf{v}_\alpha A \cdot \boldsymbol{n} = q_\alpha^{inlet} \quad (4)$$

$$p_\alpha = p_D^{outlet} \quad (5)$$

, where ϕ, k are the porosity and absolute permeability of the core sample; $\alpha = \{w, n\}$ is wetting and non-wetting phase respectively; ρ_α, μ_α, S_α, p_α are the phase density, dynamic viscosity, saturation and pressure; q_α are the source/sink terms; kr_α are the relative permeabilities of associated phases; v_α Darcy's velocity of individual phases.

The boundary conditions for the unsteady state experiment can be simply set using the Neumann condition or source term on the inlet by specifying the injection rate of the displacing phase, and by setting the Dirichlet condition on the outlet by specifying the backend pressure (Fig. 1). It is common for modelling the coreflooding experiments to define an inlet and outlet blocks [9] with zero capillary pressure and linear relative permeabilities, which in actual experiment represent mixing sections typically with 'spyder web' groove pattern to promote flow mixing and reduce capillary end effect [8].

To improve the numerical stability of the solution and more importantly in the context of this work, to promote scalability of the system for the machine learning training process the dimensionless form of the above system of equations is considered. This ensures that the trained model can be applied to experiments performed on cores of different lengths, porosities, and absolute permeabilities. The mentioned parameters can still be viewed as input features, but the scaling factors are applied directly to the solution to bring the observations and solution to the same scale.

The independent spatial variable x is scaled with respect to the length of the core and temporal variable t can be seen as pore-volumes injected over the course of the experiment.

$$x_D = \frac{x}{L}, t_D = \frac{\int_0^t q_t(t)dt}{\phi AL} \quad (6)$$

Using the chain rule, partial derivatives can be expressed as follows:

$$\frac{\partial}{\partial t} = \frac{\partial}{\partial t_D}\frac{\partial t_D}{\partial t} = \frac{q_t(t)}{\phi AL}\frac{\partial}{\partial t_D} \quad (7)$$

$$\frac{\partial}{\partial x} = \frac{\partial}{\partial x_D}\frac{\partial x_D}{\partial x} = \frac{1}{L}\frac{\partial}{\partial x_D} \quad (8)$$

Substituting the above derivates into equation (1) and accounting for the incompressibility assumption, we get:

$$\frac{\phi q_t(t)}{\phi AL}\frac{\partial S_\alpha}{\partial t_D} - \frac{1}{L}\frac{\partial}{\partial x_D}\left(\frac{k_{r\alpha}(S_\alpha)}{\mu_\alpha} k \left(\frac{1}{L}\frac{\partial p_\alpha}{\partial x_D} + \rho_\alpha g \sin\beta\right)\right) - q_\alpha = 0 \quad (9)$$

, where β is a dip angle of the core orientation measured counter-clockwise from the horizontal direction. This allows to model coreflooding in case of inclined or vertically oriented core as one dimensional problem.

The equations can be further simplified by introducing dimensionless pressure, viscosity, density, and volumetric rate.

$$P_\alpha = \frac{kA}{q_t(t)\mu_d L} p_\alpha, q_\alpha^D = \frac{AL}{q(t)} q_\alpha, \quad (10)$$

$$\mu_\alpha^D = \frac{\mu_\alpha}{\mu_d}, \mu_d = \frac{1}{2}(\mu_w + \mu_n) \quad (11)$$

$$\frac{q_t(t)}{AL}\frac{\partial S_\alpha}{\partial t_D} - \frac{1}{L}\frac{\partial}{\partial x_{\mathrm{D}}}\left(\frac{k_{r\alpha}(S_\alpha)}{\mu_\alpha^D \mu_d} k\left(\frac{q_t(t)\mu_d L}{kAL}\frac{\partial P_\alpha}{\partial x_{\mathrm{D}}} + \rho_\alpha g \sin\beta\right)\right) - \frac{q(t)}{AL} q_\alpha^D = 0 \quad (12)$$

$$\frac{\partial S_\alpha}{\partial t_D} - \frac{\partial}{\partial x_{\mathrm{D}}}\left(\frac{k_{r\alpha}(S_\alpha)}{\mu_\alpha^D}\left(\frac{\partial P_\alpha}{\partial x_{\mathrm{D}}} + \frac{Ak\rho_\alpha g}{q_t(t)\mu_d}\sin\beta\right)\right) - q_\alpha^D = 0 \quad (13)$$

, where μ_d is an average viscosity of wetting and non-wetting phases.

By introducing dimensionless density and gravity number, as the ration of gravity and viscosity force, the equation can be further simplified:

$$\rho_\alpha^D = \frac{\rho_\alpha}{\Delta\rho}, \Delta\rho = \rho_w - \rho_n \quad (14)$$

$$G = \frac{\text{gravity force}}{\text{viscous force}} = \frac{\Delta\rho g L \sin\beta}{\frac{q_t(t)\mu_d L}{kA}} = \frac{Ak\Delta\rho g \sin\beta}{q_t(t)\mu_d} \quad (15)$$

$$\frac{\partial S_\alpha}{\partial t_D} - \frac{\partial}{\partial x_{\mathrm{D}}}\left(\frac{k_{r\alpha}(S_\alpha)}{\mu_\alpha^D}\left(\frac{\partial P_\alpha}{\partial x_{\mathrm{D}}} + \rho_\alpha^D G\right)\right) - q_\alpha^D = 0 \quad (16)$$

When solving two phase incompressible immiscible flow various formulations with respect to primary variables can be applied, e.g. pressure of one phase and saturation of another phase, global pressure formulation, additionally fractional flow form can be used [10]. Here we chose the non-wetting phase pressure and wetting phase saturation formulation p_n, S_w. The system of equations can then be written as:

$$\frac{\partial S_w}{\partial t_D} - \frac{\partial}{\partial x_{\mathrm{D}}}\left(\frac{k_{rw}(S_w)}{\mu_w^D}\left(\frac{\partial P_n}{\partial x_{\mathrm{D}}} - \frac{\partial P_c(S_w)}{\partial x_{\mathrm{D}}} + \rho_w^D G\right)\right) - q_w^D = 0 \quad (17)$$

$$-\frac{\partial S_w}{\partial t_D} - \frac{\partial}{\partial x_{\mathrm{D}}}\left(\frac{k_{rn}(S_w)}{\mu_n^D}\left(\frac{\partial P_n}{\partial x_{\mathrm{D}}} + \rho_n^D G\right)\right) - q_n^D = 0 \quad (18)$$

$$S_w = 1 - S_n \quad (19)$$

$$P_c(S_w) = \frac{kA}{q_t(t)\mu_d L} p_c(S_w) \quad (20)$$

In case if Leverett-J function is supplied the equation (17) can be written as follows:

$$\frac{\partial S_w}{\partial t_D} - \frac{\partial}{\partial x_{\mathrm{D}}}\left(\frac{k_{rw}(S_w)}{\mu_w^D}\left(\frac{\partial P_n}{\partial x_{\mathrm{D}}} - \varepsilon\frac{\partial J(S_w)}{\partial x_{\mathrm{D}}} + \rho_w^D G\right)\right) - q_w^D = 0 \quad (21)$$

$$-\frac{\partial S_w}{\partial t_D} - \frac{\partial}{\partial x_{\mathrm{D}}}\left(\frac{k_{rn}(S_w)}{\mu_n^D}\left(\frac{\partial P_n}{\partial x_{\mathrm{D}}} + \rho_n^D G\right)\right) - q_n^D = 0 \quad (22)$$

$$p_c(S_w) = \frac{\sigma\cos\theta}{\sqrt{k/\phi}} J(S_w) \quad (23)$$

$$\varepsilon = \frac{\text{capillary force}}{\text{viscous force}} = \frac{\frac{\sigma\cos\theta}{\sqrt{k/\phi}}}{\frac{q_t(t)\mu_d L}{kA}} = \frac{A\sqrt{k\phi}\sigma\cos\theta}{q_t(t)\mu_d L} \quad (24)$$

, where ε is a capillary-viscous ratio.

2.2 Relative permeability representation

In this study, B-splines were used to represent the relative permeabilities and capillary pressure functions for several reasons. First, B-splines are smooth functions with smooth derivatives once the degree is higher than one; hence, they are well-suited for gradient optimization. Second, it is easy to impose monotonicity and convexity constraints on splines when needed. And last, but not least, as DeepONet structure expects inputs on fixed grid, the B-splines knots naturally fit for such requirement and allow to train model for various relative permeability and capillary pressure functions without losing on coverage of possible curves. The spline of the degree p (order $p+1$) defined on the knot vector $\boldsymbol{t}$ can be written as follows:

$$S_{p,\boldsymbol{t}}(x) = \sum_{i=0}^{m-p-1} C_i \cdot B_{i,p}(x) \quad (25)$$

, where C_i are the control points of B-spline $C_0, \dots, C_{m-p-1}$; $m+1$ is the number of B-spline knots $t_0, \dots, t_m$; p is the degrees of B-spline polynomials.

By definition B-spline of order p+1 is a polynomial of degree p. As can be seen the spline of order p+1 with m+1 knots has $m-p$ control points. B-spline basis functions can be recursively defined using the Cox de Boor formula, where the first-order B-spline is a piecewise constant function. In the given formulas the convention '0/0=0' is applied:

$$B_{i,0}(t) = \begin{cases} 1, & \text{if } t_j \le t \le t_{j+1} \\ 0, & \text{otherwise} \end{cases} \quad (26)$$

$$B_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} B_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(t) \quad (27)$$

, where $0 \le t_i \le 1$ are the B-spline knots defined for $i = 0 \dots m$, where $m+1$ is a number of control points of corresponding B-spline.

Due to the way the recursion formulas are defined it can be seen that for each interval $[t_k, t_{k+1})$ additionally p knots should be defined before and after interval ends, hence overall

there are $m-2p-1$ internal knots with $p+1$ extra knots from each side, in total $m+1$ knots $t_0, \dots, t_m$. Therefore, if there are n control points, the number of required knots is $n+p+1$, among which only $n-p+1$ are unique. When knot is repeated multiple times it is defined as the multiplicity of the knot. Therefore, the first and last knots have $p+1$ multiplicity [11].

$$t = \left(\underbrace{0, \cdots, 0}_{p+1}, t_{p+1}, \cdots, t_{m-p-1}, \underbrace{1, \cdots, 1}_{p+1} \right) \tag{28}$$

It is important to mention that knots generally do not represent the coordinates of the control points (Fig. 2). To obtain the coordinates of the control polygon (t_i^*, C_i), the so called Greville abscissas should be calculated [11]:

$$t_i^* = \frac{1}{p} \sum_{k=1}^{p} t_{i+k} \tag{29}$$

The monotonicity condition can be derived from the hodograph formula (first derivative) of the B-spline [12]:

$$C_{i-1} \leq C_i, \quad i = 1, \dots, m-p-1 \tag{30}$$

The convexity condition can also be expressed using only the condition on the control points [12]:

$$C_i \leq \frac{1}{2}(C_{i-1} + C_{i+1}), i = 1, \dots, m-p-2 \tag{31}$$

Both monotonicity and convexity conditions are useful when one wants to impose constraints on the shape of capillary pressure and relative permeability. It can be applied during inverse modelling and the generation of training samples of relative permeabilities.

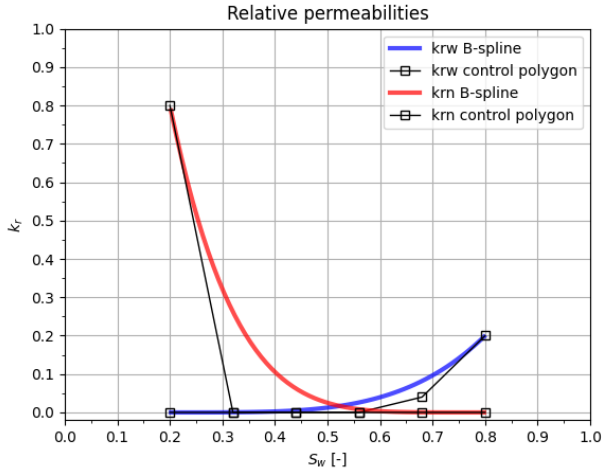

Fig. 2. Relative permeabilities using B-spline approximation.

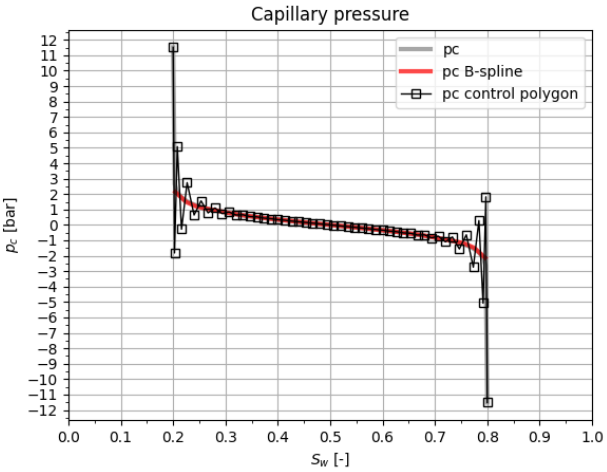

Fig. 3. Capillary pressure using B-spline approximation.

To further simplify and have better control over the optimization and training process, the relative permeabilities are determined on the effective saturation domain [0,1], and the endpoints are introduced to map B-spline range to [0,1] interval:

$$k_{r\alpha}(S_w) = kr_\alpha^{max} \cdot S_{p,t}\big(S_e(S_w)\big) \tag{32}$$

$$p_c(S_w) = S_{p,t}\big(S_e(S_w)\big) \tag{33}$$

$$S_e(S_w) = \frac{S_w - S_{wr}}{1 - S_{wr} - S_{nr}} \tag{34}$$

, where $\alpha = \{w, n\}$ are the wetting and non-wetting phase correspondingly; S_{wr}, S_{nr} are the residual saturation of wetting and non-wetting phases, kr_w^{max}, kr_n^{max} are the end point of wetting phase and non-wetting phase relative permeabilities.

In the current definition, when relative permeability endpoints are used as scaling parameters, the first and last control points of relative permeabilities are fixed and equal to 0 and 1, respectively.

2.2 Physics informed machine learning

The main characteristic of physics-informed neural networks is the ability of the solution to satisfy physical constraints. This is done through the extension of the loss function with ordinary or partial differential equations describing the model of interest. Given that neural network functions are differentiable, which is how the back-propagation step is performed during training, the partial derivatives defined in the loss function can be calculated directly. The observation data loss can be either ignored in PINN or only a small sample of data is present, which is a second characteristic of PINN that allows the model to be trained without observation data or with much less observation data than traditional neural network approaches. Typically, adding observation data improves the quality of the trained model and speeds up convergence. Observations are usually obtained from simulation results or analytical solutions, if they exist, and

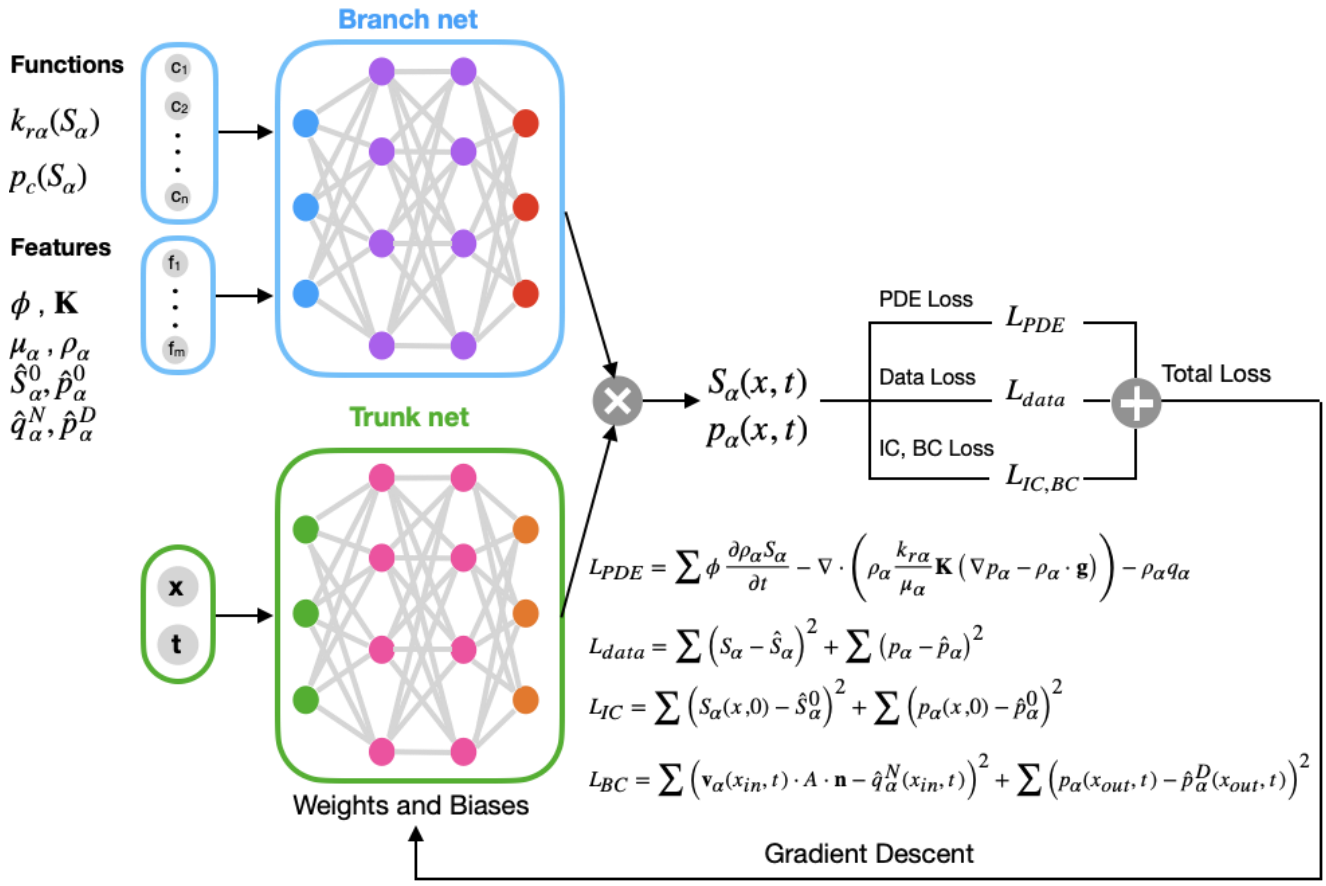

Fig. 4. Physics informed DeepONet architecture for modelling two phase core flooding experiments.

sampled at locations of interest. The second component of the machine learning model used in this study is DeepONet [1,13], which is a special architecture that allows to train operators defined on the function space. The combination of PINN and DeepONet is known as PI-DeepONet [14].

2.2.1 Forward modelling

The loss function of the PINN should contain PDE loss and optional data loss. The loss function can be extended with the initial and boundary conditions, that is, the so-called soft conditions. Alternatively, these conditions can be enforced directly, so-called hard conditions. We use hyperparameter flags, which activate one of the mentioned conditions. Therefore, all the components of the loss function can be defined as follows:

$$L = \lambda_{data} L_{data} + \lambda_{PDE} L_{PDE} + b_{IC,soft} \lambda_{IC} L_{IC} + b_{BC,soft} \lambda_{BC} L_{BC} \quad (35)$$

$$L_{PDE} = \frac{1}{N} \sum_{i=1}^{N} \phi \frac{\partial \rho_\alpha S_\alpha}{\partial t} - \nabla \cdot \left(\rho_\alpha \frac{k_{r\alpha}(S_\alpha)}{\mu_\alpha} \boldsymbol{K} \cdot (\nabla p_\alpha - \rho_\alpha \boldsymbol{g}) \right) - q_\alpha \quad (36)$$

$$L_{data} = \frac{1}{N} \sum_{i=1}^{N} \left(S_\alpha(x_i,t) - \hat{S}_\alpha(x_i,t) \right)^2 + \frac{1}{N} \sum_{i=1}^{N} \left(p_\alpha(x_i,t) - \hat{p}_\alpha(x_i,t) \right)^2 \quad (37)$$

$$L_{IC} = \frac{1}{N} \sum_{i=1}^{N} \left(S_\alpha(x_i,0) - \hat{S}_\alpha(x_i,0) \right)^2 + \frac{1}{N} \sum_{i=1}^{N} \left(p_\alpha(x_i,0) - \hat{p}_\alpha(x_i,0) \right)^2 \quad (38)$$

$$L_{BC} = \frac{1}{2} \sum_{i=0}^{2} \left(\mathbf{v}_\alpha(x_{in},t) \cdot A \cdot \boldsymbol{n} - \mathrm{q}_\alpha^N(x_{in},t) \right)^2 + \frac{1}{2} \sum_{i=1}^{2} \left(p_\alpha(x_{out},t) - \hat{p}_\alpha^D(x_{out},0) \right)^2 \quad (39)$$

, where λ indices the weights for each loss function contribution; $b_{IC,soft}$, $b_{BC,soft}$ are the hyperparameter flags indication whether soft or hard boundary and initial conditions are applied.

Internally, instead of the general form (36) the dimensionless form of the PDE loss is used, as defined in equations (21)-(22).

As previously mentioned, Dirichlet boundary conditions can also be enforced using a penalty approach [13].

$$\mathcal{G}(v)(x,t) = g(x), x \in \Gamma_D \quad (40)$$

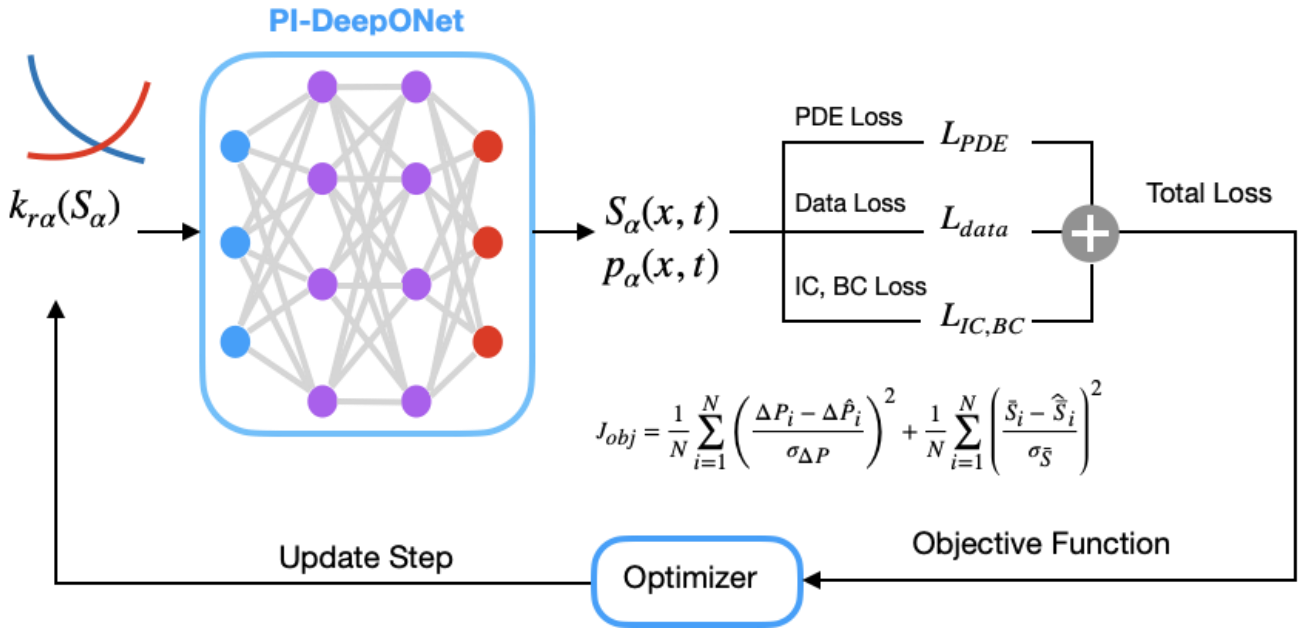

Fig. 5. Inverse modeling using Physics Informed DeepONet.

$$\mathcal{G}(v)(x,t) = g(x) + \chi_D(x)\mathcal{N}(v)(x,t) \tag{41}$$

$$\chi_D(x) = \begin{cases} 0, & \text{if } x \in \Gamma_D \\ > 0, & \text{otherwise} \end{cases} \tag{42}$$

As our coreflooding problem is defined in [0,1] domain and Dirichlet condition are applied only on the outlet, while on the inlet the Neumann conditions are applied through the source term, we can specify Dirichlet boundary conditions as follows.

$$\mathcal{G}(v)(x=1,t) = g_1^D \tag{43}$$

$$\mathcal{G}(v)(x,t) = g_1^D + (1-x)\mathcal{N}(v)(x,t) \tag{44}$$

A similar penalty approach can be applied to enforce the initial conditions.

$$\mathcal{G}(v)(t=0) = g_0 \tag{45}$$

$$\mathcal{G}(v)(x,t) = e^{-t}g_0 + (1-e^{-t})\mathcal{N}(v)(x,t) \tag{46}$$

Finally, when both the initial and boundary conditions are applied, the solution is as follows:

$$\mathcal{G}(v)(t=0) = g_0 \tag{47}$$

$$\mathcal{G}(v)(x=1,t) = g_1^D \tag{48}$$

$$\mathcal{G}(v)(x,t) = e^{-t}(1-x)g_0 + (1-e^{-t})g_1^D + (1-e^{-t})(1-x)\mathcal{N}(v)(x,t) \tag{49}$$

DeepONet [1,13–15] is a special architecture of the neural network that is based on the universal approximation theorem for operator, where two separate networks are built, namely the branch and trunk net (Fig. 4). The branch network is responsible for featurizing the input functions, such as relative permeability and capillary pressure, defined on fixed nodes, in our case, knots of the B-spline. It can also be extended with other features, such as the characteristics of the core, fluids, boundary, and initial conditions. The trunk network, on the other hand, takes the spatial and temporal coordinates as input. Therefore, the DeepONet network can be expressed as follows:

$$G(v)(x,t) = \sum_{i=1}^{p} b_k(v)t_k(x,t) + b_0 \tag{50}$$

, where b_0 is the bias term; $b_1, \dots, b_p$ are the p outputs of the branch network, and $t_1, \dots, t_p$ are the p output of the trunk network; v input functions; x, t spatial and temporal coordinates.

In our implementation we adopt a branch input as canonical curve formulation rather than the conventional sensor-based DeepONet branch [15]. The relative permeability curves are represented by monotone clamped B-splines on the effective saturation, and the branch input is the concatenation of the B-spline control polygons together with the rock-fluid characteristics, [S_{wc}, S_{or}, k_{rw}^{end}, k_{ro}^{end}, μ_w, μ_n, ϕ, K]. The B-Splines control polygon is used as the canonical, low-dimensional parameterisation of the input function that the branch consumes directly. The control polygons are produced once by an offline B-spline fit and are also the natural parameter for inversion, since gradient-based kr recovery from saturation observations optimises c_w and c_o directly via autograd through the frozen forward operator (see Section 2.2.3). The trunk network retains its conventional role of taking the spatial and temporal coordinates (x, t) as input. To capture the sharp saturation fronts characteristic of two-phase displacement, the trunk decoder in this work is implemented as a finite-volume graph network (FVGN) trained against a weak-entropy physics-informed (WE-PINN) loss, as described in Section 2.2.2.

It is worth mentioning that DeepONet is not the only operator learning network, and it is not necessary to have

branch and trunk nets to learn operators. However, it was shown that such an architecture has advantages compared to other approaches, primarily generalization across different input domains, flexibility in the architecture of the trunk and branch net, and accuracy due to universal approximation theorem for operators [15].

2.2.2 FVGN and WE-PINNs

The expressiveness of the trunk network in DeepONet controls how sharply the operator can resolve local features of the solution. For incompressible immiscible two-phase displacement in a porous media, the dominant local feature is the saturation shock that forms as soon as the wetting phase is injected: the non-convex fractional-flow function $f_w(S_w)$ produces a moving saturation shock. A pointwise trunk decoder typically smears this shock into a diffuse front, lets its position drift over time, and produces non-physical expansion shocks unless an entropy selection criterion is imposed. Capturing the saturation shock at the correct speed and amplitude is therefore the central challenge for the trunk-side decoder, addressed in the rest of this section by the finite-volume graph-network (FVGN) trunk and the weak-entropy physics-informed (WE-PINN) loss.

Shock capturing is a well-known difficulty for neural PDE solvers [16], addressed in the literature by artificial-viscosity smoothing, architectural fixes, and weak-form losses [17,18]; De Ryck & Mishra in particular gives the theoretical foundation for why a strong-form L_2 residual fails at the shock, but both that work and Chaumet & Giesselmann rely on a dual-norm min-max formulation that requires an extra optimisation over a parametric family of test functions. In contrast, Oubarka et al [19], cast the same weak-entropy admissibility condition as a space-time flux balance on randomly sampled control volumes, which is far cheaper to evaluate and is the formulation we follow here.

Graph neural networks (GNNs) operate on data structured as a graph of nodes and edges, propagating information through learned message-passing steps in which each node updates its state from the features of its incident edges and neighbouring nodes; this gives them a natural inductive bias for problems on unstructured meshes and for discretisation schemes whose update at a point depends only on a local stencil. Pfaff et al. [20]applied this idea to mesh-based PDE simulation in the MeshGraphNets framework, and Li et al. [21] (FVGN) extended it to a finite-volume formulation in which each cell is a node, each face is an edge, and the learned messages on edges play the role of inter-cell fluxes. Finite volumes are the natural choice here for two reasons. First, the scheme conserves wetting-phase volume by construction: every face flux added to one cell is subtracted from its neighbour, so the total water in the core is preserved no matter how the fluxes are computed. Second, conservative schemes admit weak solutions and propagate shocks at the correct speed, which is exactly the property the trunk decoder needs. Following this idea, we discretise the core as a one-dimensional graph of finite-volume cells and advance the wetting-phase saturation one step at a time as $S_i^{n+1} = S_i^n - \frac{\Delta t}{\phi V_i}\left(F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}\right)$, where $F_{i\pm\frac{1}{2}}$ are the wetting-phase volumetric fluxes through the right and left faces of cell i. The role of the GNN is therefore not to predict the saturation directly but to learn the local stencil that produces these face fluxes from the local saturations, cell geometry, and the relative permeabilities supplied by the DeepONet branch.

On top of the standard PINN losses introduced in Section 2.2.1 (PDE residual, data, initial and boundary conditions), the WE-PINN formulation adds two further terms that target the saturation shock directly. The first is an entropy loss that enforces the Kruzhkov entropy inequality cell by cell: it penalises only entropy creation and leaves the natural entropy dissipation across the shock untouched, which is what selects the physical shock and forbids non-physical expansion shocks. The second is a total-variation (TVD) loss that penalises any increase in the total variation of the saturation profile from one step to the next, which damps the spurious oscillations that conservative schemes can produce around steep fronts. Together with the conservation built into the FVGN update, these two losses give the trunk decoder the structural bias it needs to place and propagate the saturation shock at the correct speed.

2.2.3 Inverse modelling

The inverse modelling in the current work is achieved through gradient optimization in the same framework as neural network training because of its convenience and potential for further advances (Fig. 5). The optimization parameters are the control points of the relative permeability functions $C_{i,\alpha}$ and relative permeability endpoints $k_{r\alpha}^{max}$. The objective function is defined as:

$$J_{obj} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\Delta p_i - \Delta\widehat{p_i}}{\sigma_{\Delta \mathrm{p}}}\right)^2 + \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\Delta S_{avg,i} - \Delta\widehat{S_{avg,i}}}{\sigma_{S_{avg}}}\right)^2 \tag{51}$$

, where $\Delta\,\widehat{p_i}$ and $\widehat{S_{avg,i}}$ are the differential pressure and average wetting phase saturation measurements inside the core; $\sigma_{\Delta p}$ and $\sigma_{S_{avg}}$ are the standard deviations of corresponding measurements.

The initial guess for the optimization is formed using the analytical approach JBN [22] for the unsteady state experiment, based on the Buckley-Leverett solution [23]. The JBN approach calculates the relative permeability functions at the outlet phase after a breakthrough occurs.

$$S_w(L,t) = S_{wr} + \frac{1}{AL\phi}\left(t\frac{dQ_w(t)}{dt} - Q_w(t)\right) \tag{52}$$

$$k_{r\alpha}(S_w) = \mathrm{v}_\alpha L\frac{\mu_\alpha}{k}\left(\Delta\hat{\mathrm{p}}(t) - t\frac{d\Delta\hat{\mathrm{p}}(t)}{dt}\right)^{-1} \tag{53}$$

$$\mathrm{v}_\alpha = \frac{1}{A}\frac{dQ_\alpha}{dt} \tag{54}$$

, where $\alpha = \{w, n\}$ is wetting and non-wetting phase correspondingly; Q_w, Q_n are the production of expelled phased.

To complete the shape of the curves the Corey approximation is used [24].

$$k_{r\alpha}(S_\alpha) = k_{r\alpha}^{max}\left(\frac{S_\alpha - S_{r\alpha}}{1 - S_{wr} - S_{nr}}\right)^{c_\alpha} \quad (55)$$

, where c_α are the Corey exponents.

After optimization using PI-DeepONet, a simulation run using DuMuX simulator [25] is performed, for accuracy and assurance reasons.

3 Results and Discussion

To produce supervised data for the machine learning (ML) model, we decided to use the DuMuX simulator [7,25], which is well suited for the modelling of multiphase multicomponent flow in porous media, capable of modelling pore network flow and solving Navier-Stokes equations using a phase field approach for capturing the interfaces of several phases. This opens great opportunities to adapt interpretation routines to various fluid systems and experimental setups. In addition, DuMuX has already been used to develop the SCORES simulator [9], which is freely available for performing simulation runs, but is unfortunately not provided with the corresponding source code. We plan to fix this by releasing open-source code for coreflooding experiments using DuMuX implemented during this study. Additionally, the corresponding open-source Python package [26], will include the interpretation methods presented in this paper. It is worth mentioning that, as we believe in open science, there are other great open-source tools available for interpreting of SCAL experiments [27].

3.1 DuMuX vs. SCORES Benchmark

Based on a very thorough and extensive benchmark study of four simulators, SCORES, SENDRA, PORLAB, and CYDAREX [28,29], an unsteady state experiment (Case-3) was used (Table 1) to validate the simulation model. As shown in Fig. 6 and Fig. 7, the results for the differential pressure and average saturation in the core are practically identical. This is not surprising because both results are based on the DuMuX simulator; however, it is worth mentioning that the results from SCORES were published based on at least and eight years older version of DuMuX. The injection schedule setup is defined in Table 1, and the core sample and fluid properties are listed in Table 2. The capillary pressure is shown in Fig. 3. The relative permeabilities have endpoints $kr_w^{max} = 0.5$ and $kr_n^{max} = 0.5$, and Corey exponents $c_w = 3$ and $c_n = 3$ corresponsingly.

Table 1. Injection schedule.

Time (hour)	Q water (cm3/hour)
10	1
13	10
16	70
21	200
26	300

Table 2. Core sample and fluid properties.

Property	Value
core length [cm]	8
core diameter [cm]	4
core permeability [mD]	100
water viscosity [cP]	1
water density [g/cm^3]	1
oil viscosity [cP]	5
oil density [g/cm^3]	0.8
initial water saturation	0.2
final water saturation	0.8

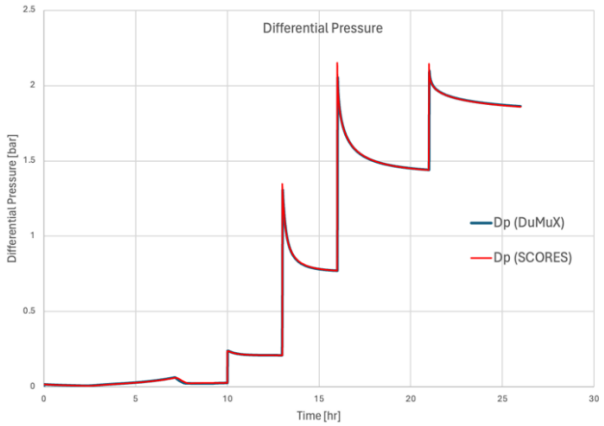

Fig. 6. Differential pressure. Unsteady state simulation benchmark DuMuX vs. SCORES (DuMuX) (Case-3).

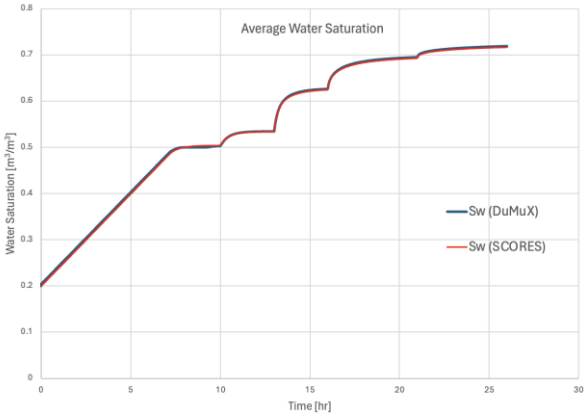

Fig. 7. Average water saturation in the core. Unsteady state simulation benchmark DuMuX vs. SCORES (DuMuX) (Case-3).

3.2 Training and Validation

The model is implemented in PyTorch and trained on an 80-cell mesh for 4000 epochs split into four progressive-rollout phases (800, 1000, 1000 and 1200 epochs at rollout horizons of 1, 2, 4 and 8 steps), so the operator is gradually asked to forecast further ahead before each phase boundary. Optimisation uses Adam with learning rate 1e-3, weight decay 1e-5, gradient clipping at 1.0, and a cosine learning-rate schedule within each phase. The B-spline relative permeabilities are pre-fit to the analytical Corey curves in a 1500-step offline pass and then frozen for the rest of training: only the FVGN flux-correction network is updated during the

rollout. The data loss is computed against reference saturation and pressure trajectories produced by DuMuX [7,25] for the same injection schedule, core, and fluid properties, with random-start sampling from the DuMuX rollout on 80% of steps to make the operator robust to off-trajectory initial conditions. The best checkpoint is selected by the validation L_2 error against DuMuX at t = 0.1, 0.2 and 0.3. The training curves are shown in and the resulting fits on the relative permeabilities, and saturation rollouts are shown in Fig. 9 and Fig. 10. Validation against the DuMuX reference. Top-left: predicted wetting-phase saturation in the (x, t) plane, showing the characteristic Buckley–Leverett triangular profile with a sharp shock propagating to the right at the Welge velocity. Top-right: DuMuX reference in the same colour scale; visually indistinguishable from the prediction, with the shock position matching cell-by-cell. Bottom-left: B-spline relative permeabilities kr_w and kr_o overlaid on the analytical Corey curves, which coincide to plotting precision. Bottom-right: time slices through both rollouts at t = 0.1, 0.2, 0.3 and 0.4 (prediction solid, DuMuX dashed), with all four shock fronts overlapping closely and no visible lag or smearing..

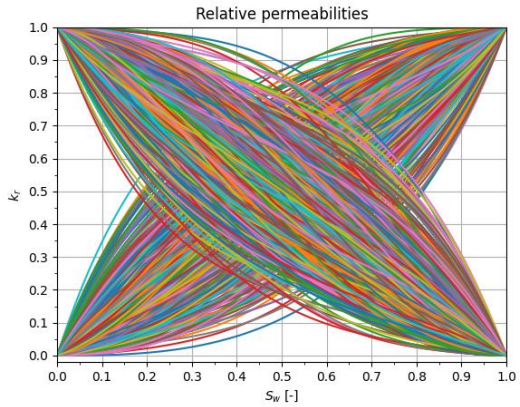

Fig. 8. Training samples of relative permeabilities.

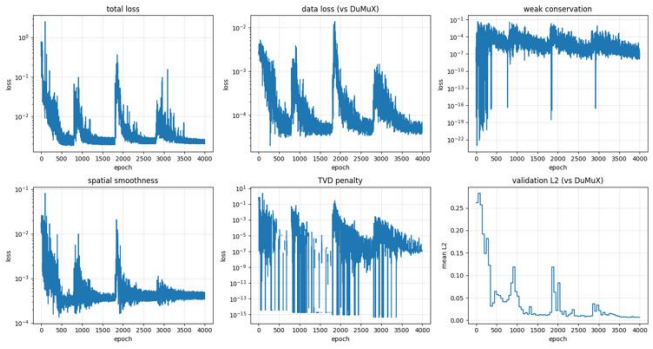

Fig. 9. Training loss curves over 4000 epochs. The total loss, data loss against the DuMuX reference, weak-conservation residual, spatial-smoothness term, TVD penalty, and validation L2 are plotted against epoch. Phase boundaries at epochs 800, 1800 and 2800 mark the rollout-horizon doublings ($1 \rightarrow 2 \rightarrow 4 \rightarrow 8$ steps); short transient spikes at each boundary are absorbed within roughly 200 epochs as the operator adapts to the longer forecast horizon. The data loss is the dominant term and converges to about 5e-5; the weak-conservation residual stays at machine precision throughout, confirming that wetting-phase volume is conserved by construction by the FVGN.

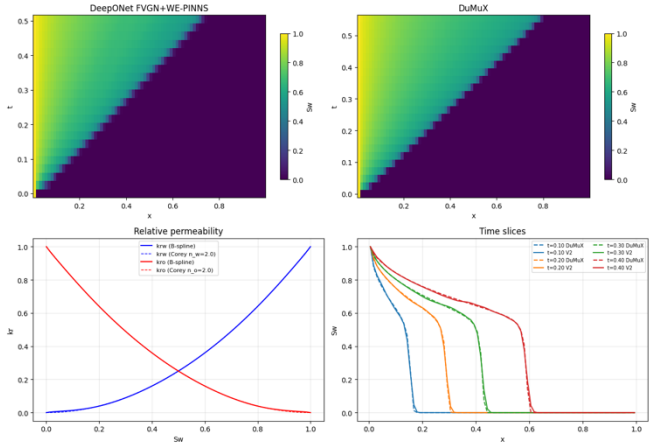

Fig. 10. Validation against the DuMuX reference. Top-left: predicted wetting-phase saturation in the (x, t) plane, showing the characteristic Buckley–Leverett triangular profile with a sharp shock propagating to the right at the Welge velocity. Top-right: DuMuX reference in the same colour scale; visually indistinguishable from the prediction, with the shock position matching cell-by-cell. Bottom-left: B-spline relative permeabilities kr_w and kr_o overlaid on the analytical Corey curves, which coincide to plotting precision. Bottom-right: time slices through both rollouts at t = 0.1, 0.2, 0.3 and 0.4 (prediction solid, DuMuX dashed), with all four shock fronts overlapping closely and no visible lag or smearing.

4 Conclusions

This study introduces the use of physics-informed neural networks (PINN) combined with the operator-learning neural network architecture DeepONet for interpreting special core analysis experiments, specifically focusing on estimating relative permeabilities. The PI-DeepONet approach combines the advantages of existing analytical, semi-analytical, and numerical methods while offering new possibilities. The neural network is trained to satisfy both observation data and multiphase flow equations, with relative permeability functions represented using B-splines. A Finite-Volume Graph Network combined with Weak-Entropy loss functions allows the operator to capture sharp saturation fronts while performing physics-informed machine learning.

Validation against the DuMuX reference on Corey relative permeabilities with non-trivial residual saturations shows that the FVGN + WE-PINN operator places and propagates the saturation shock at the reference's accuracy while preserving the physical bounds exactly. Together with the gradient-based inverse loop described in Section 2.2.3, this means a calibrated PI-DeepONet can serve as a fast, differentiable surrogate for the forward simulator inside an interpretation workflow: relative permeability functions can be recovered from saturation observations through standard backpropagation, opening a route to physics-consistent, on-the-fly relative permeability estimation for unsteady-state coreflooding experiments.

Acknowledgments

The authors thank the open-source community and academia for continuing to provide access to state-of-the-art research and the most advanced technology.

References

1. L. Lu, P. Jin, and G. E. Karniadakis, "DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators," (2019).
2. Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier Neural Operator for Parametric Partial Differential Equations," ICLR 2021 - 9th International Conference on Learning Representations (2020).
3. A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to Simulate Complex Physics with Graph Networks," 37th International Conference on Machine Learning, ICML 2020 8428–8437 (2020).
4. M. Cranmer, S. Greydanus, S. Hoyer, G. Research, P. Battaglia, D. Spergel, and S. Ho, "Lagrangian Neural Networks," (2020).
5. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition **2016-December**, 770–778 (2015).
6. W. Weng and X. Zhu, "U-Net: Convolutional Networks for Biomedical Image Segmentation," IEEE Access **9**, 16591–16603 (2015).
7. T. Koch, D. Gläser, K. Weishaupt, S. Ackermann, M. Beck, B. Becker, S. Burbulla, H. Class, E. Coltman, S. Emmert, T. Fetzer, C. Grüninger, K. Heck, J. Hommel, T. Kurz, M. Lipp, F. Mohammadi, S. Scherrer, M. Schneider, G. Seitz, L. Stadler, M. Utz, F. Weinhardt, and B. Flemisch, "DuMux 3 – an open-source simulator for solving flow and transport problems in porous media with a focus on model coupling," Computers & Mathematics with Applications **81**, 423–443 (2021).
8. C. McPhee, J. Reed, and I. Zubizarreta, "Best Practice in Coring and Core Analysis," Developments in Petroleum Science **64**, 1–15 (2015).
9. J. G. Maas, "Open source simulator DuMuX available for SCAL data interpretation," SCA2011-08 (2011).
10. R.Helmig, *Multiphase Flow and Transport Processes in the Subsurface*, A Contribution to the Modeling of Hydrosystems (Springer-Verlag, 1997).
11. L. Piegl and W. Tiller, "The NURBS Book," (1997).
12. L. Schumaker, "Spline functions: Basic theory, third edition," Spline Functions: Basic Theory, Third Edition 1–582 (2007).
13. L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators," Nature Machine Intelligence 2021 3:3 **3**(3), 218–229 (2021).
14. S. Wang, H. Wang, and P. Perdikaris, "Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets," Sci. Adv. **7**(40), (2021).
15. L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, G. E. Karniadakis, L. Lu, X. Meng, and S. Cai, "ScienceDirect A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data," Comput. Methods Appl. Mech. Eng. **393**, 114778 (2022).
16. J. Abbasi, A. D. Jagtap, B. Moseley, A. Hiorth, and P. Ø. Andersen, "Challenges and advancements in modeling shock fronts with physics-informed neural networks: A review and benchmarking study," Neurocomputing **657**, 131440 (2025).
17. A. Chaumet and J. Giesselmann, "Improving Weak PINNs for Hyperbolic Conservation Laws: Dual Norm Computation, Boundary Conditions and Systems," SMAI Journal of Computational Mathematics **10**, 373–401 (2024).
18. T. De Ryck and S. Mishra, "Numerical analysis of physics-informed neural networks and related models in physics-informed machine learning," Acta Numerica **33**, 633–713 (2024).
19. I. Oubarka, I. Kissami, M. Boubekeur, F. Benkhaldoun, A. Madrane, and Z. Saadi, "Weak and entropy physics-informed neural networks for conservation laws," (2026).
20. T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, "Learning Mesh-Based Simulation with Graph Networks," ICLR 2021 - 9th International Conference on Learning Representations (2020).
21. T. Li, S. Zou, X. Chang, L. Zhang, and X. Deng, "Finite Volume Graph Network(FVGN): Predicting unsteady incompressible fluid dynamics with finite volume informed neural network," Physics of Fluids **36**(4), (2024).
22. E. F. Johnson, D. P. Bossler, and V. O. N. Bossler, "Calculation of Relative Permeability from Displacement Experiments," Transactions of the AIME **216**(01), 370–372 (1959).
23. B. S. E Buckley, M. C. Leverett, and M. Aime, "Mechanism of Fluid Displacement in Sands," Transactions of the AIME **146**(01), 107–116 (1942).
24. A. T. Corey, "Mechanics of immiscible fluids in porous media," 252 (1994).
25. B. Flemisch, M. Darcis, K. Erbertseder, B. Faigle, A. Lauser, K. Mosthaf, S. Müthing, P. Nuske, A. Tatomir, M. Wolff, and R. Helmig, "DuMux: DUNE for multi-{phase,component,scale,physics,...} flow and transport in porous media," Adv. Water Resour. **34**(9), 1102–1112 (2011).
26. R. Manasipov, "coreflow," https://pypi.org/project/coreflow/ (2024).
27. S. Berg, H. Dijk, E. Unsal, R. Hofmann, B. Zhao, and V. Raju Ahuja, "Simultaneous determination of relative permeability and capillary pressure from an unsteady-state core flooding experiment," Comput. Geotech. **168**, 106091 (2024).
28. R. Lenormand, K. Lorentzen, J. G. Maas, and D. Ruth, "Comparison of four numerical simulators for SCAL experiments," SCA2016-006 (2016).
29. R. Lenormand, K. Lorentzen, J. G. Maas, and D. Ruth, "Comparison of four numerical simulators for SCAL experiments," Petrophysics **58**(1), 48–56 (2017).