USING PHYSICS-INFORMED NEURAL NETWORKS TO SOLVE FOR PERMEABILITY FIELD UNDER TWO-PHASE FLOW IN HETEROGENEOUS POROUS MEDIA

Mingyuan Yang & John T. Foster*

Department of Petroleum and Geosystems Engineering, The University of Texas at Austin, Austin, Texas 78712, USA

*Address all correspondence to: Mingyuan Yang, Department of Petroleum and Geosystems Engineering, The University of Texas at Austin, Austin, Texas 78712, USA; Tel.: +1 512 201 5615, E-mail: mingyuanyang@utexas.edu

Original Manuscript Submitted: 11/18/2022; Final Draft Received: 12/8/2022

Physics-informed neural networks (PINNs) have recently been applied to a wide range of computational physical problems. In this paper, we use PINNs to solve an inverse two-phase flow problem in heterogeneous porous media where only sparse direct and indirect measurements are available. The forward two-phase flow problem is governed by a coupled system of partial differential equations (PDEs) with initial and boundary conditions. As for inverse problems, the solutions are assumed to be known at scattered locations but some coefficients or variable functions in the PDEs are missing or incomplete. The idea is to train multiple neural networks representing the solutions and the unknown variable function at the same time such that both the underlying physical laws and the measurements can be honored. The numerical results show that our proposed method is able to recover the incomplete permeability field in different scenarios. Moreover, we show that the method can be used to forecast the future dynamics with the same format of loss function formulation. In addition, we employ a neural network structure inspired by the deep operator networks (DeepONets) to represent the solutions which can potentially shorten the time of the training process.

KEY WORDS: *physics-informed neural networks, two phase flow, inverse problem, dynamics prediction*

1. INTRODUCTION

Deep learning techniques have recently been developed to solve various computational science and engineering problems (Berg and Nyström, 2018; Fraces et al., 2020; Goswami et al., 2020; Haghighat et al., 2020; He et al., 2020; Kharazmi et al., 2020; Raissi, 2018; Tartakovsky et al., 2018; Tchelepi and Fuks, 2020; Yang et al., 2018; Zhu et al., 2019). More specifically, physics-informed neural networks (PINNs) have been used to solve engineering problems where the governing partial differential equations (PDEs) are known and some of the solutions can be acquired (Berg and Nyström, 2018; He et al., 2020; Raissi, 2018). The neural networks are defined with several hidden layers followed by user designed nonlinear activation functions such that they are capable of approximating the complex solutions. PINNs provide an alternative approach to classical numerical techniques like finite-difference methods (FDMs) in solving PDEs

2689–3967/23/$35.00 © 2023 by Begell House, Inc. www.begellhouse.com

given the universal approximation properties of neural networks (Kharazmi et al., 2020; Raissi, 2018). In addition, the PINNs approach is also capable of integrating data or measurements as prior information seamlessly so that forward and inverse problems can be solved via the same neural network architecture (He et al., 2020).

Deep learning has been used recently to model flow and transport in subsurface porous media (Song and Tartakovsky, 2022; Tchelepi and Fuks, 2020; Wang et al., 2021a, 2022; Wang and Lin, 2020; Yang and Foster, 2021; Zhou et al., 2022). In the paper by Song and Tartakovsky (2022), the authors proposed to use deep convolutional NN as a surrogate model to predict dynamic behaviors. The training was performed on multifidelity data and the trained model was used for uncertainty quantification which would be very computationally expensive on physical models directly. In the paper by Zhou et al. (2022), the authors proposed to use a combination of convolutional adversial autoencoder and a dense convolutional encoder-decoder to construct a surrogate model which then was used for inversion. The idea is also to replace the expensive physical model with a more affordable surrogate model; in the meantime, the accuracy is retained. In Wang and Lin (2020), the authors used the deep convolutional auto-encoder to build an efficient surrogate which predicts the transport of the water saturation in multiphase subsurface transport. Their method requires the water saturation data from the whole domain for the entire duration of simulation to train the surrogate. Once the model is trained, it can reduce the time of prediction by orders of magnitude with respect to a traditional numerical solver. In Wang et al. (2021a), the authors used the theory-guided auto-encoder to learn the dynamics of a scaler transport in one-phase flow. The training is performed in a data-free fashion which is similar to the idea of PINNs in that the physical laws are embedded into the loss function so that no input-output pairs are needed as in traditional machine learning applications. The trained model can also be used to solve inverse problems, but the accuracy of the model depends on the sampling for training. In Yang and Foster (2021) and Tchelepi and Fuks (2020), the authors used PINNs directly to solve the forward and inverse two-phase flow problems but the applications are limited to one-dimensional cases.

In this work, we use PINNs to solve the inverse two-phase flow problems in porous media in two dimensions. The goal is to solve for the heterogeneous permeability field with sparse measurements of water saturation, pressure, and permeability. This is analogous to the history matching in petroleum engineering reservoir simulation in that the parameters of forward simulators are “tuned” to fit actual hydrocarbon production data from a reservoir. We define multiple neural networks to represent each variable and use them to formulate the residuals of the balance laws. Also, given measurements can be satisfied by constructing additional loss terms with a mean squared error norm. In the Results section, we present different cases for inversion to show the flexibility of the approach. Interestingly, this approach can also be used to solve hybrid problems such that the inversion and forecast can be achieved at the same time. We also propose to use a neural network structure similar to DeepONets (Wang and Perdikaris, 2021; Wang et al., 2021b) for time-dependent variables. The architecture is composed of two separate fully connected networks which process spatial and temporal information. This can potentially reduce the training time by using fewer evaluation points to formulate the residuals when compared with finite-difference methods.

2. PHYSICS-INFORMED NEURAL NETWORKS FOR INVERSE PROBLEMS

In the framework of PINNs, we embed the conservation equations, constitutive laws, and other physical constraints into the loss function. Any available measurements and data can also be

enforced to satisfy additional loss term with a desired norm (e.g., the mean squared error). The loss function is then minimized by using optimizers in an iterative fashion and the weights and bias of the neural network (NN) are updated during the training process. After a tolerance is reached, the NN can serve as the field solutions or the variable function of interest which satisfies both the physics laws and any available measurements.

A schematic of a PINN using fully connected networks is shown in Fig. 1. The neural network u_{NN} in Fig. 1 is used to represent the solution u. In this work, and without loss of generalization, only fully connected neural networks are considered which can be expressed as

$$u_{NN}(x; W, b) = T^m \circ T^{m-1} \circ \cdots \circ T^2 \circ T^1(x), \tag{1}$$

$$T^j(\cdot) = \sigma^j(W^j \times \cdot + b^j), \quad j = 1, 2, \ldots, n, \tag{2}$$

where W^j and b^j are the weights and bias of layer j; $(\cdot)$ is the output from the previous layer. In this work, multiple neural networks are defined for different variables and the outputs are used to construct the loss function in Fig. 1. More details of the loss function formulation are presented in Section 3.

In this work, we use neural networks with the DeepONets structure to represent the time dependent variables. A schematic is shown in Fig. 2. We define two fully connected neural networks to embed the spatial and temporal information and each produces multiple outputs.

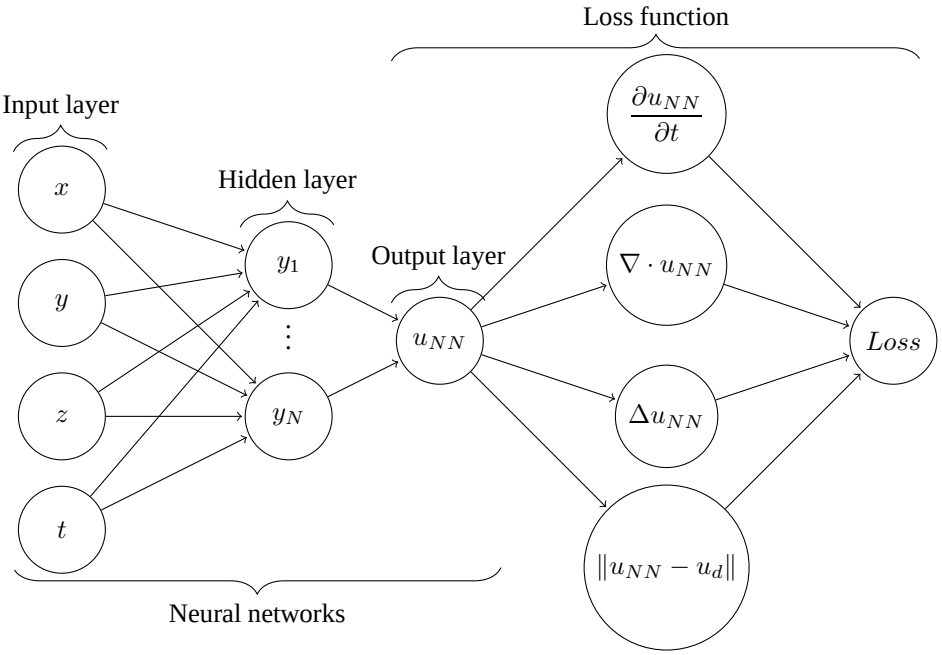


FIG. 1: A fully connected neural network schematic with four, N, and one neuron in the input, hidden, and output layers, respectively. The loss function is constructed with operations applied on the output of the neural networks u_{NN}.

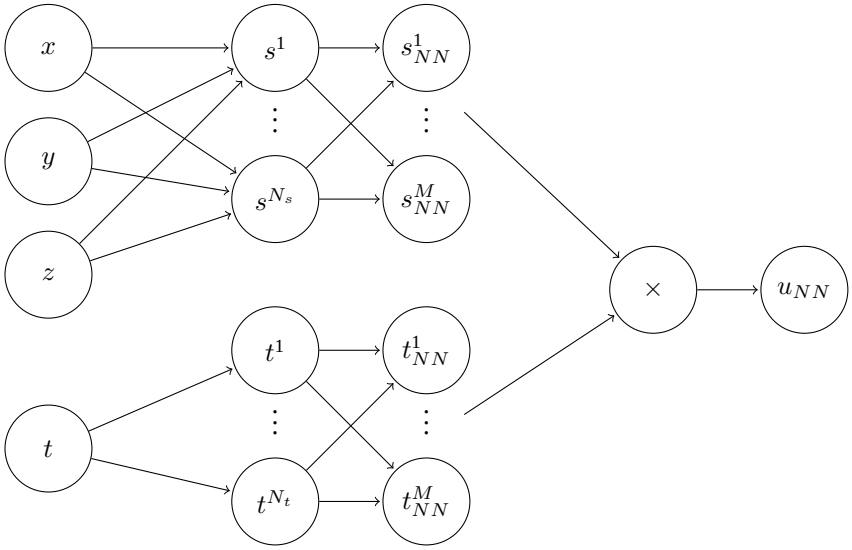


FIG. 2: A schematic of neural networks with the DeepONets structure

Then the final output is obtained by computing the product,

$$u_{NN}(x, y, z, t) = \sum_{i=1}^{M} s^i_{NN}(x, y, z) \cdot t^i_{NN}(t), \tag{3}$$

where M is the number of outputs of the space and time neural networks s_{NN} and t_{NN}. This proposed neural network does not lose the generality in that it can be used to evaluate the value at an arbitrary point in the domain. However, it can provide some benefits certain formulations of the physical law residuals when compared to the conventional fully connected networks in Fig. 1. For example, if the finite different method is used to discretize PDEs, then we will need $(N_x \times N_y \times N_z \times N_t)$ points to formulate the residual and the backpropagation will update the weights and bias for each point. With the proposed neural networks, the space and time evaluation can be separated so that only $(N_x \times N_y \times N_z + N_t)$ times forward evaluations are needed to generate the points to formulate the residual which can greatly reduce the training time. For more discussion of DeepONets please refer to the papers by Wang et al. (2021b) and Wang and Perdikaris (2021).

3. TWO-PHASE FLOW MODEL WITH PINNS

3.1 Incompressible Two-Phase Flow Model

In this work, gravity effects and capillary pressure are neglected; hence the flow equations with Darcy's law can be written as

$$u = -\lambda(S_w)k\nabla p, \tag{4}$$

$$\nabla \cdot u = r, \tag{5}$$

where u is the total flow velocity, k is the absolute permeability, p is the pressure, r is the total source/sink, and $\lambda(S_w)$ is the total mobility which is a function of the water saturation (S_w). u

and $\lambda(S_w)$ can be calculated as follows:

$$u = u_w + u_o, \tag{6}$$

$$u_i = -\frac{k_{ri}}{\mu_i} k \nabla p, \quad i = w, o, \tag{7}$$

$$\lambda(S_w) = \frac{k_{rw}(S_w)}{\mu_w} + \frac{k_{ro}(S_w)}{\mu_o}, \tag{8}$$

where the subscript w and o represent the water and oil phase, u_w and u_o are the respective flow velocities, μ_w and μ_o are the respective viscosities which are considered as constant here and throughout this work, k_{rw} and k_{ro} are the respective relative permeabilities which are functions of S_w. The equation of S_w transport can be written as

$$\frac{\partial S_w}{\partial t} + u \cdot \nabla f_w = r_w, \tag{9}$$

where r_w is the source/sink of water and f_w is the water fractional flow function which can be calculated as a function of S_w:

$$f_w = \frac{k_{rw}/\mu_w}{k_{rw}/\mu_w + k_{ro}/\mu_o}. \tag{10}$$

In this work, all variables are considered to be dimensionless with normalization and the constitutive relations for the two-phase flow model are the same for all cases in Section 4. The ratio μ_w/μ_o is constant as 5.0. k_{rw} and k_{ro} are defined as

$$k_{rw} = S_w^2, \tag{11}$$

$$k_{ro} = (1 - S_w)^2, \tag{12}$$

where S_w is a scalar function between 0 and 1. The functions of k_{rw}, k_{ro}, and f_w are shown in Fig. 3.

3.2 PINNs with a Finite Difference Residual Formulation

Three neural networks $\mathcal{N}_s$, $\mathcal{N}_p$, and $\mathcal{N}_k$ are defined for S_w, p, and k to construct the overall loss function for training. Note that u can be calculated with S_w and p; hence we do not need a

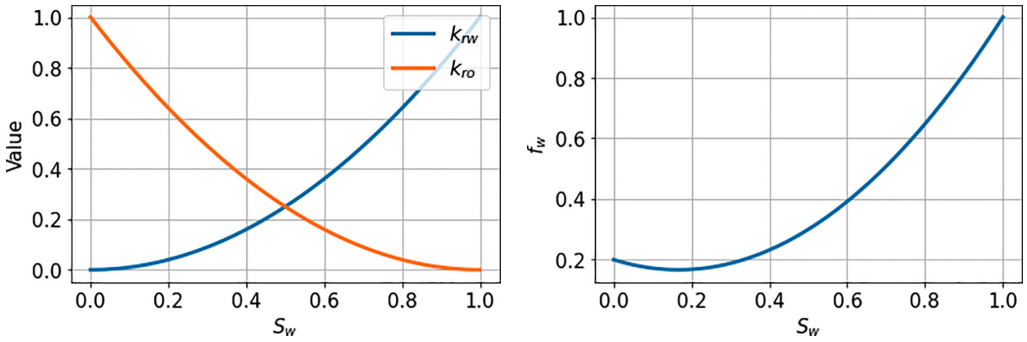


FIG. 3: Relative permeability functions k_{rw} and k_{ro}, and the water fractional flow function f_w that are used in this work

separate neural network for u. The three neural networks can be evaluated on structured space-time grids,

$$S_w{}^{i,j,q} = \mathcal{N}_s(i\Delta x, j\Delta y, q\Delta t), \tag{13}$$

$$p^{i,j,q} = \mathcal{N}_p(i\Delta x, j\Delta y, q\Delta t), \tag{14}$$

$$k^{i,j} = \mathcal{N}_k(i\Delta x, j\Delta y), \tag{15}$$

where $i = 1, 2, ..., N_x$; $j = 1, 2, ..., N_y$; and $q = 1, 2, ..., N_t$. N_x, N_y, and N_t are the number of grid points in each dimension. Δx, Δy, and Δt are the grid size in each dimension. In this work, we assume to have no source/sink terms for both phases. The residual of (5) can be calculated as

$$\lambda^{i,j,q} = \lambda(S_w{}^{i,j,q}), \tag{16}$$

$$c_x = \frac{\lambda^{i+1,j,q}k^{i+1,j} + \lambda^{i,j,q}k^{i,j}}{2}, \tag{17}$$

$$c_y = \frac{\lambda^{i,j+1,q}k^{i,j+1} + \lambda^{i,j,q}k^{i,j}}{2}, \tag{18}$$

where c_x and c_y are the average coefficients[†] between grid points in the x and y dimensions. Then we have

$$r_x{}^{i,j,q} = \frac{c_x{}^{i+1,j,q}(p^{i+2,j,q} - p^{i+1,j,q}) - c_x{}^{i,j,q}(p^{i+1,j,q} - p^{i,j,q})}{\Delta x^2}, \tag{19}$$

$$r_y{}^{i,j,q} = \frac{c_y{}^{i,j+1,q}(p^{i,j+2,q} - p^{i,j+1,q}) - c_y{}^{i,j,q}(p^{i,j+1,q} - p^{i,j,q})}{\Delta y^2}, \tag{20}$$

$$r_c{}^{i,j,q} = r_x{}^{i,j,q} + r_y{}^{i,j,q}, \tag{21}$$

$$R_c = \frac{1}{N_c}\sum {r_c}^2, \tag{22}$$

where N_c is the total number of r_c which is $(N_x - 2) \times (N_y - 2) \times N_t$. For the transport equation (9), we use the backward Euler method to discretize the time dimension and formulate the residual using in its implicit form

$$f_w{}^{i,j,q} = f_w(S_w{}^{i,j,q}), \tag{23}$$

$$\begin{aligned} r_s{}^{i,j,q} = {} & \frac{S_w{}^{i,j,q} - S_w{}^{i,j,q-1}}{\Delta t} - \lambda^{i,j,q}k^{i,j}\left(\frac{p^{i+1,j,q} - p^{i-1,j,q}}{2\Delta x}\right) \\ & \times \left(\frac{f_w{}^{i+1,j,q} - f_w{}^{i-1,j,q}}{2\Delta x}\right) - \lambda^{i,j,q}k^{i,j}\left(\frac{p^{i,j+1,q} - p^{i,j-1,q}}{2\Delta y}\right) \\ & \times \left(\frac{f_w{}^{i,j+1,q} - f_w{}^{i,j-1,q}}{2\Delta y}\right), \end{aligned} \tag{24}$$

$$R_s = \frac{1}{N_s}\sum {r_s}^2, \tag{25}$$

[†]There is no loss of generality in using the algebraic mean here; we could alternatively use the harmonic mean as is often done in petroleum reservoir simulation.

where N_s is the total number of r_s which is $(N_x - 2) \times (N_y - 2) \times (N_t - 1)$. To make the problem well defined, we also need to include the residuals of the boundary conditions (Dirchlet, Neumann or mixed) and the initial condition in the overall loss function. They can be calculated in a similar fashion as R_c and R_s so that we omit the details here. We refer to the general boundary condition and initial condition as R_b and R_i.

Finally, we calculate the mean squared error (MSE) of S_w, p, and k from the neural networks with respect to any available data. This is the key part of the inversion and it guarantees that the resulting neural networks honor the data where available. In this work, we assume to have sparse measurements of ${S_w}^d$, p^d, and k^d, so that the MSE can be calculated as

$$R_{ms} = \frac{1}{N_{ms}} \sum_{i=0}^{N_{ms}} \left(\mathcal{N}_s \left(x_i, y_i, q_i \right) - {S_w}_i^d \right)^2, \tag{26}$$

$$R_{mp} = \frac{1}{N_{mp}} \sum_{i=0}^{N_{mp}} \left(\mathcal{N}_p \left(x_i, y_i, q_i \right) - p_i^d \right)^2, \tag{27}$$

$$R_{mk} = \frac{1}{N_{mk}} \sum_{i=0}^{N_{mk}} \left(\mathcal{N}_k \left(x_i, y_i \right) - k_i^d \right)^2, \tag{28}$$

where N_{ms}, N_{mp}, and N_{mk} are the number of respective measurements. Note that these three loss terms have the same format as that of the Dirchlet boundary condition residuals. Therefore, the PINNs framework can be used to solve the forward and inverse problems with the same structure which makes it possible to integrate various kinds of data in training. The overall loss function L can then be defined as

$$L = w_1(R_c + R_s) + w_2(R_b + R_i + R_{ms} + R_{mp} + R_{mk}), \tag{29}$$

where w_1 and w_2 are the user defined weights (i.e., hyperparameters) for different loss terms.

4. NUMERICAL RESULTS

In this section, we present our numerical simulation results of two schemes. In the first scheme (Case 1 to Case 4), water is injected from the left side of a unit square domain which is saturated with oil at the beginning and fluids flow out through the right side. The injection and producing pressure are fixed. The top and bottom sides are assumed to be impermeable. Therefore, the boundary conditions and initial conditions are

$$\begin{aligned} S_w(x = 0) &= 1.0, \\ S_w(t = 0) &= 0.0, \\ p(x = 0) &= 1.0, \\ p(x = 1) &= 0.0, \\ u(y = 0) &= 0.0, \\ u(y = 1) &= 0.0. \end{aligned}$$

The S_w profiles at three different time points are shown in Fig. 4. These serve as the synthetic S_w data for the first four cases.

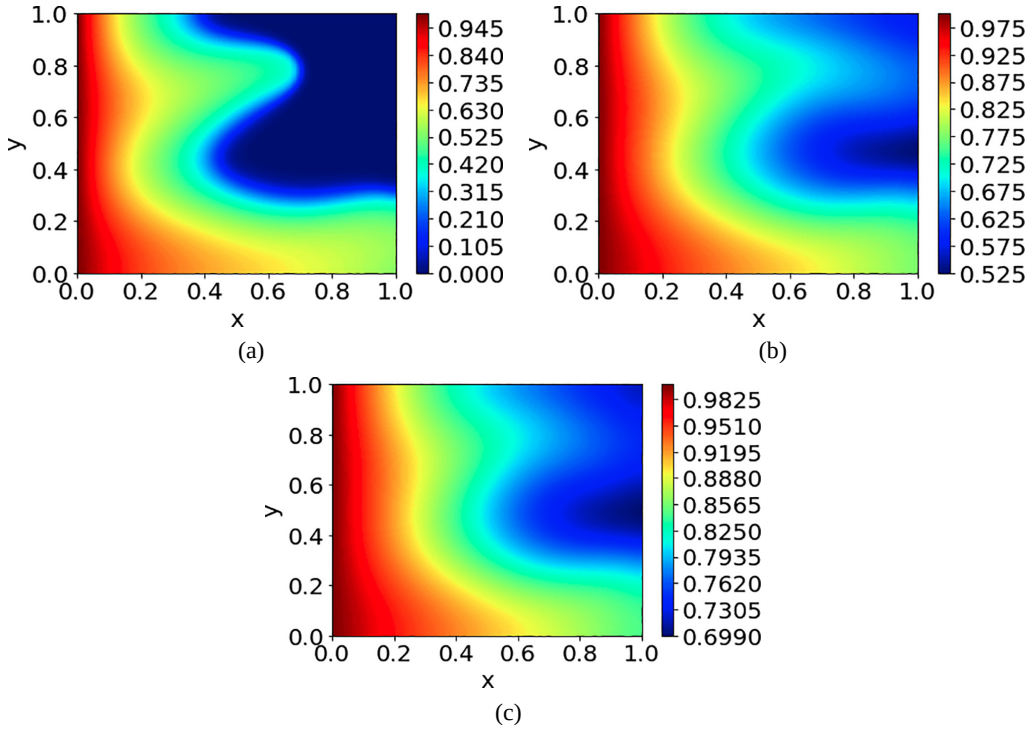


FIG. 4: S_w profiles at different times: (a) S_w at $t = 2$; (b) S_w at $t = 4$; and (c) S_w at $t = 6$

In the second scheme (Case 5), we assume to have multiple injection and producing wells in the domain. All the wells are operated at fixed pressure, so that they can be seen as pointwise Dirchlet boundary conditions of p. All four sides are assumed to be impermeable. The boundary condition and initial condition can be described as follows:

$$
\begin{aligned}
S_w({x_i}^{inj}, {y_i}^{inj}) &= 1.0, \quad i = 1, 2, ..., N_{inj}, \\
S_w(t = 0) &= 0.0, \\
u(x = 0) &= 0.0, \\
u(x = 1) &= 0.0, \\
u(y = 0) &= 0.0, \\
u(y = 1) &= 0.0, \\
p({x_i}^{inj}, {y_i}^{inj}) &= 1.0, \quad i = 1, 2, ...N_{inj}, \\
p({x_i}^{pro}, {y_i}^{pro}) &= 0.0, \quad i = 1, 2, ...N_{pro},
\end{aligned}
$$

where N_{inj} and N_{pro} are the number of injection and producing wells. The S_w profiles at three different times are shown in Fig. 5. This will serve as the synthetic S_w data for Case 5. In this work, we use synthetic data for training and they come from the finite-element simulations with FEniCS (Alnæs et al., 2015) using a highly refined mesh. Note that we added a small diffusion term of S_w in Eq. (8) when running the FEniCS simulations to make the solution stable. Nevertheless, we can add the same diffusion in the PINNs formulation so that it does

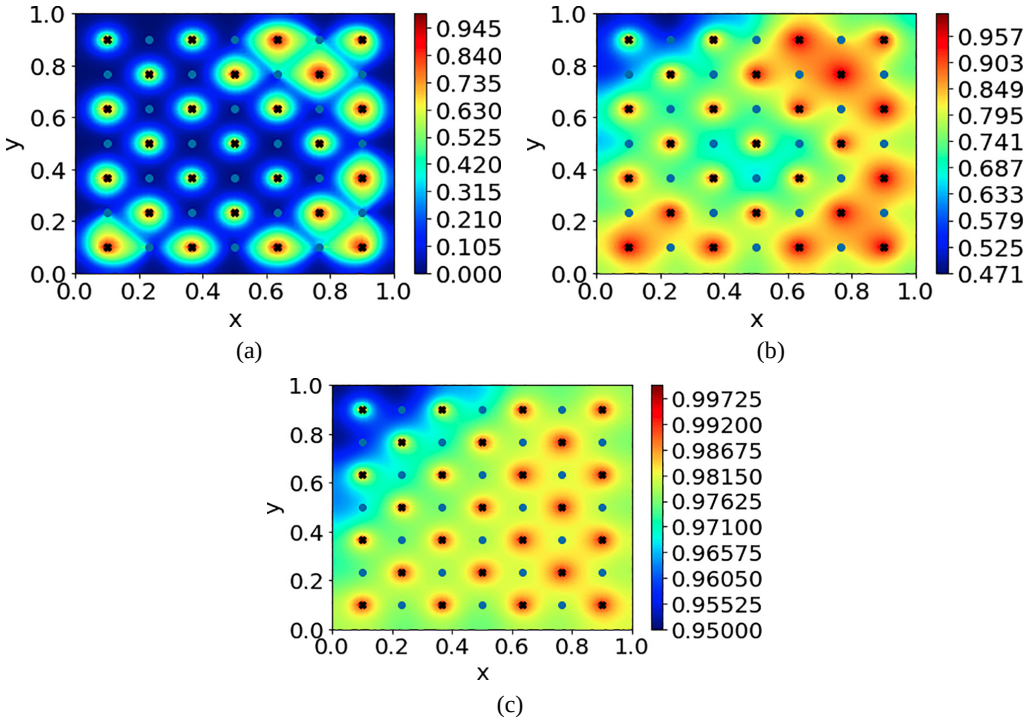


FIG. 5: S_w profiles at different times: (a) S_w at $t = 0.2$; (b) S_w at $t = 0.6$; and (c) S_w at $t = 1.0$

not degrade the generality of our approach. It is common to see the diffusion effects in the nature due to the capillary pressure. In this work, all the PINNs applications are implemented with PyTorch (Paszke et al., 2017). An LBFGS optimizer is used to train the neural networks and the hyperbolic tangent function ($\tanh$) is used as the activation function. The space and time neural networks for S_w and p have the same structure which consists of six hidden layers and 40 neurons per layer (40 neurons in the output layer). The neural networks for k has two hidden layers and 40 neurons per layer (one neuron in the output layer). We use the same neural networks for all the cases throughout this study.

4.1 Case 1: Measurements on a Regular Grid

In this case, we assume to have direct measurements of k on a 7×7 regular grid. Also, we assume that measurements of S_w and p are available at the same locations during the simulation. The results are shown in Figs. 6–8. It is shown that we are able to obtain a k field that is in agreement with the true k field well at the end of training. Also, we verified the outputs of the neural networks for S_w and p and it shows that the measurements are also in agreement.

4.2 Case 2: Measurements at Random Locations

In this case, we assume to have direct measurements of k on 49 random locations. Again, we assume that measurements of S_w and p are available at the same locations for the duration of the

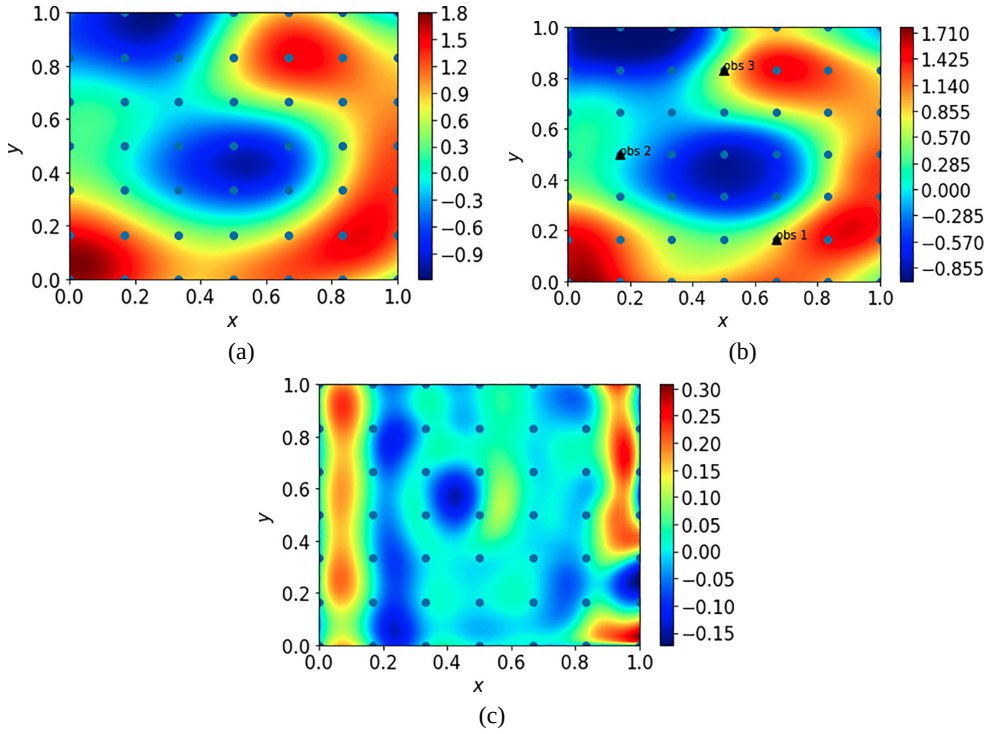


FIG. 6: Comparison of the inverted k field to the true k field. The blue dots indicate the locations of measurements, the black triangles are the three observation locations where we verify the match of S_w and p data. (a) True k field in logarithm, (b) inverted k field in logarithm, and (c) absolute error of k.

simulation. The results are shown in Figs. 9–11. As in the previous case, we are able to obtain a k field that matches the true k field well at the end of training.

Figure 9 shows that the inverted k field agrees with the true k field well at the end of training. In the meantime, the p and S_w predictions match the measurements at locations that are not on regular grids.

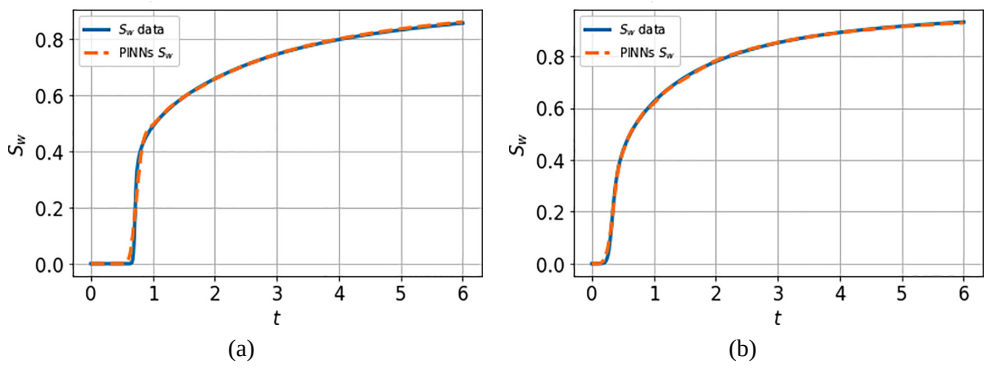
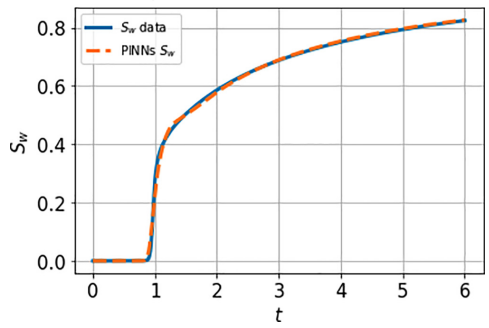
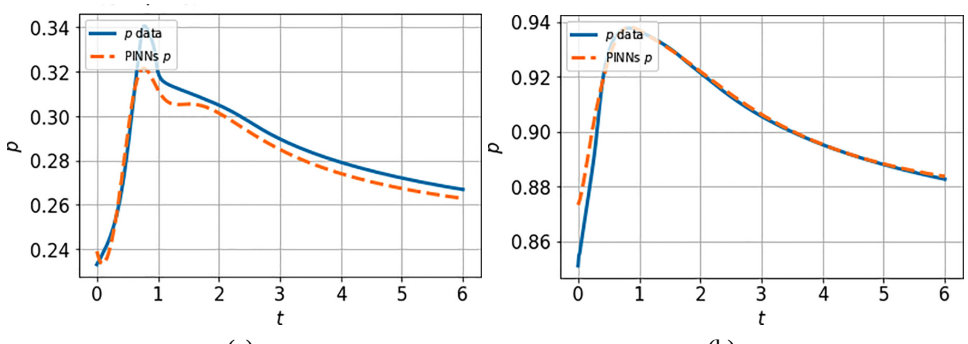


FIG. 7.

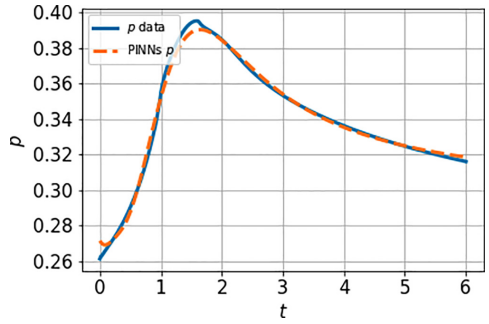


(c)

FIG. 7: Verification of S_w match at three observation locations. (a) S_w comparison at obs 1, (b) S_w comparison at obs 2, and (c) S_w comparison at obs 3.



(a) (b)



(c)

FIG. 8: Verification of p match at three observation locations. (a) p comparison at obs 1, (b) p comparison at obs 2, and (c) p comparison at obs 3.

4.3 Case 3: Using Fewer Direct Measurements of k

In this case, we assume to have fewer direct measurements of k than in the last two cases. The measurements of S_w and p remain the same as in the last case. The results are shown in Figs. 12–14. It is demonstrated that we are still able to obtain a k field in agreement with the true data with only ten measurements of k. Again, the measurements of S_w and p are honored closely as well.

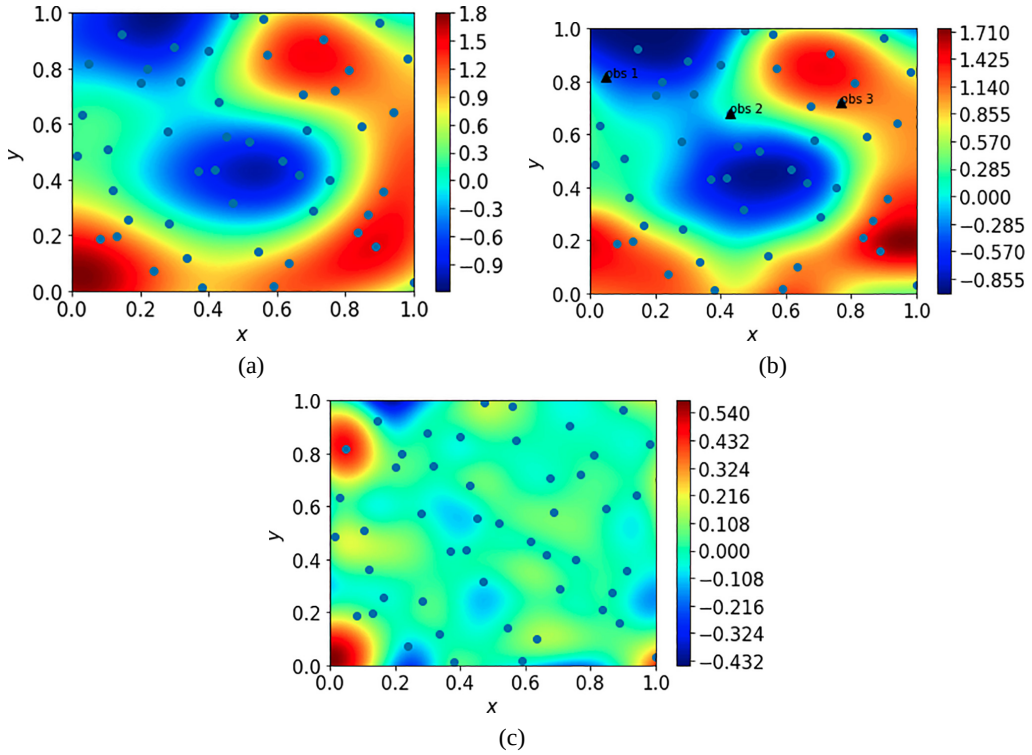


FIG. 9: Comparison of the inverted k field to the true k field. The blue dots indicate the locations of measurements, the black triangles are the three observation locations where we verify the match of S_w and p data. (a) True k field in logarithm, (b) inverted k field in logarithm, and (c) absolute error of k.

4.4 Case 4: Forecasting the Dynamics

In this case, we assume to have measurements of S_w and p for a shorter period of time and solve a hybrid problem. For time from 0 to 1.2, it is the same inversion problem as the last few cases, and

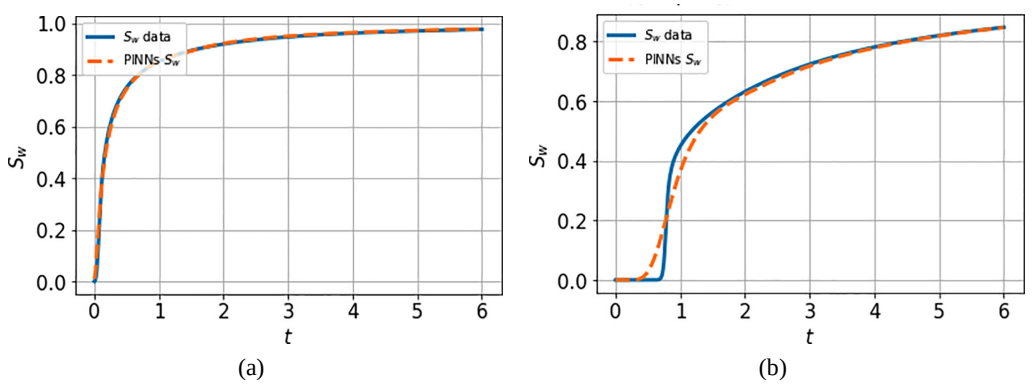
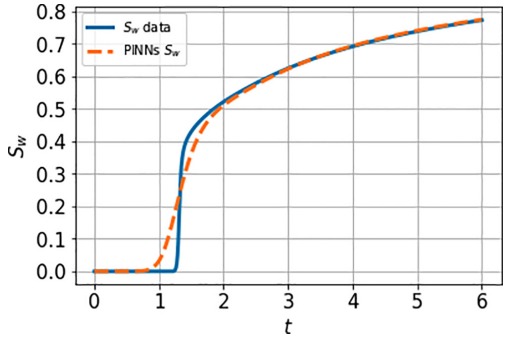
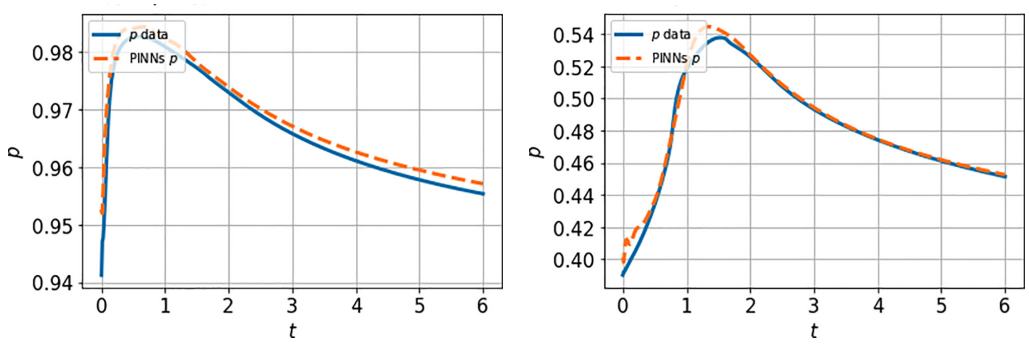


FIG. 10.



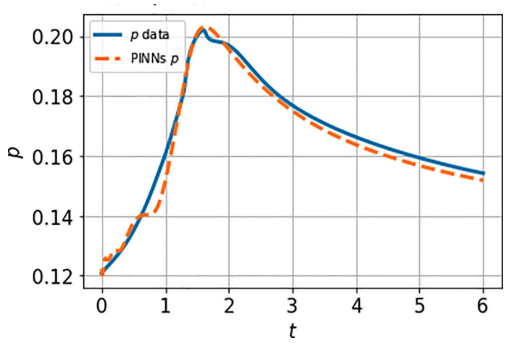
(c)

FIG. 10: Verification of S_w match at three observation locations. (a) S_w comparison at obs 1, (b) S_w comparison at obs 2, and (c) S_w comparison at obs 3.



(a)

(b)



(c)

FIG. 11: Verification of p match at three observation locations. (a) p comparison at obs 1, (b) p comparison at obs 2, and (c) p comparison at obs 3.

for time from 1.2 to 6, it is actually a forward problem as no S_w and p measurements are available except for boundary conditions. With the framework of PINNs, we are able to achieve these two goals at the same time. It is also worth noting that the forward simulation with FEniCS using a CPU takes about 3200 sec., while the whole training takes only about 1340 sec. using a GPU.

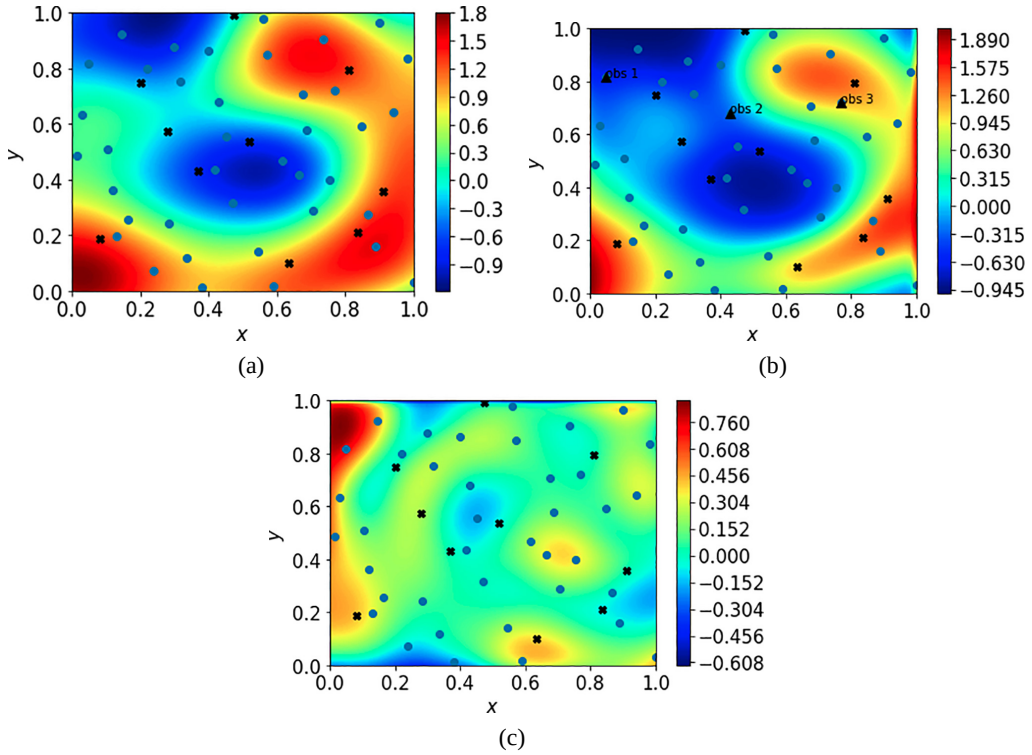


FIG. 12: Comparison of the inverted k field to the true k field. Measurements of S_w and p are located on all the markers (blue dots and black crosses); measurements of k are located on black crosses (10 in total); black triangles are the three observation locations where we verify the match of S_w and p data. (a) True k field in logarithm, (b) inverted k field in logarithm, and (c) absolute error of k.

The results are shown in Figs. 15–17. Again, the inverted k field matches the true k field well. Moreover, the dynamics of S_w and p can be forecasted well even for locations where the measurements are all prior to water breakthrough (e.g., observation 3).

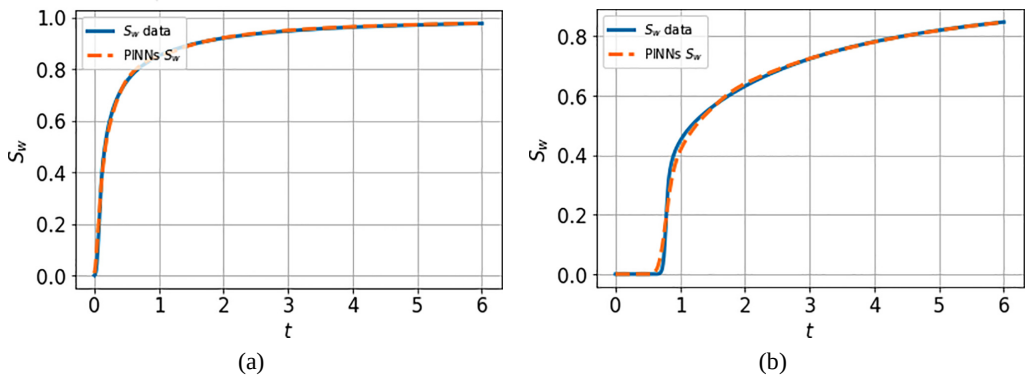
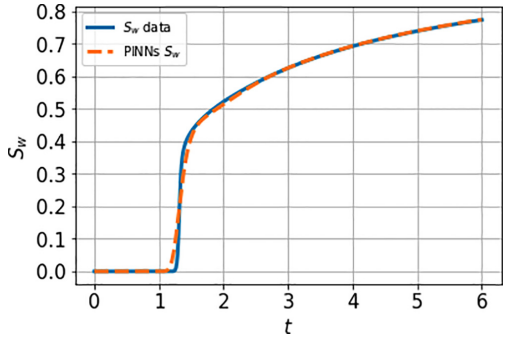
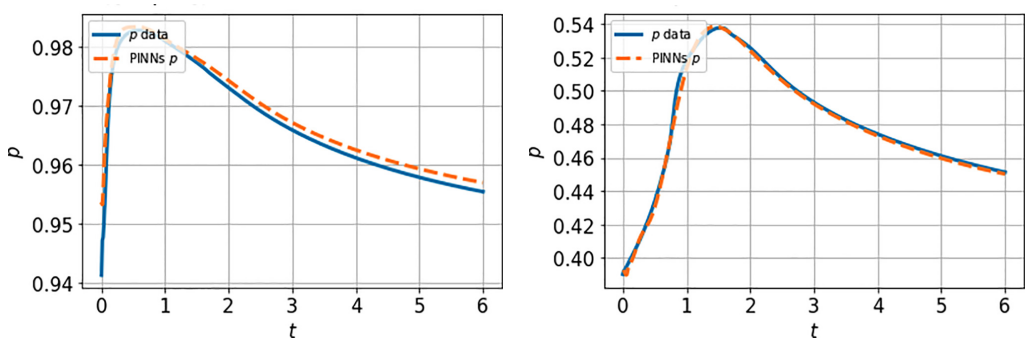


FIG. 13.

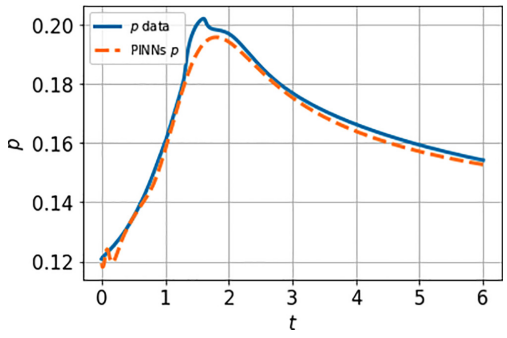


(c)

FIG. 13: Verification of S_w match at three observation locations. (a) S_w comparison at obs 1, (b) S_w comparison at obs 2, and (c) S_w comparison at obs 3.



(a) (b)



(c)

FIG. 14: Verification of p match at three observation locations. (a) p comparison at obs 1, (b) p comparison at obs 2, and (c) p comparison at obs 3.

4.5 Case 5: Multiple Injection and Producer Wells

In this case, we apply the same algorithm to the multiple-well setting which is more commonly seen in the petroleum engineering reservoir problems. The results are shown in Fig. 18 and we are able to obtain a close k field again at the end of training.

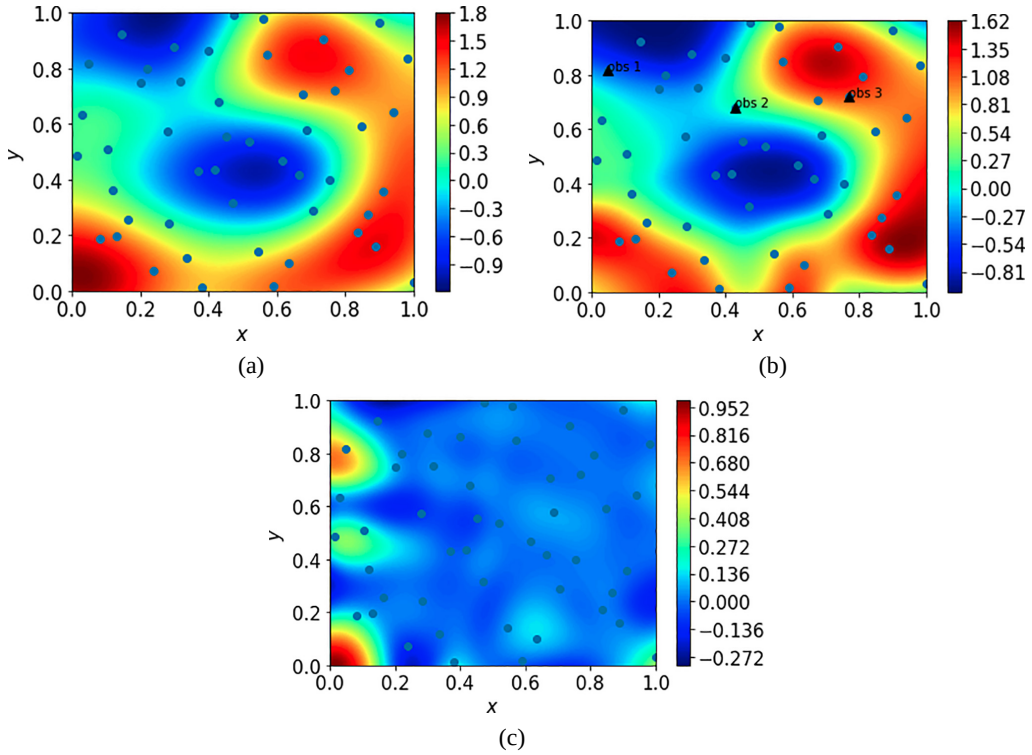


FIG. 15: Comparison of the inverted k field to the true k field. The blue dots indicate the locations of measurements; the black triangles are the three observation locations where we verify the match of S_w and p data. (a) True k field in logarithm, (b) inverted k field in logarithm, and (c) absolute error of k.

5. CONCLUSION AND FUTURE WORK

In this work, we used PINNs to solve the inverse two-phase flow problem in heterogeneous media. The results demonstrate that the permeability field k in the entire domain can be obtained

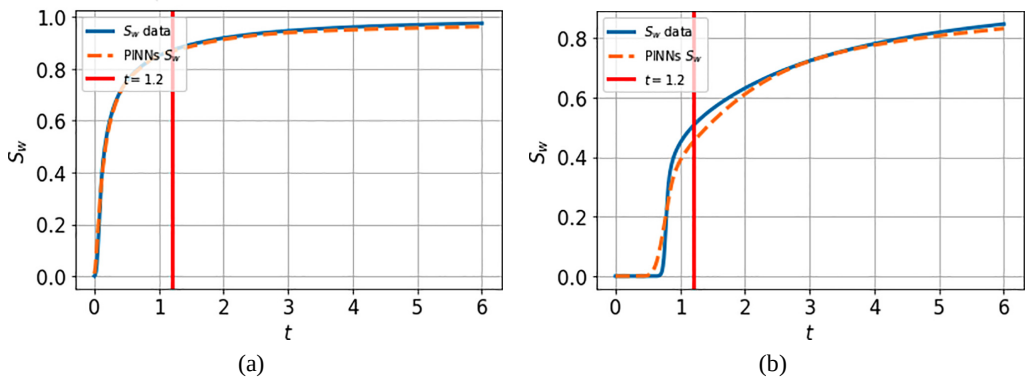
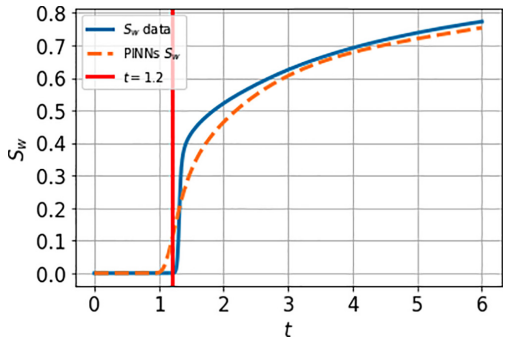
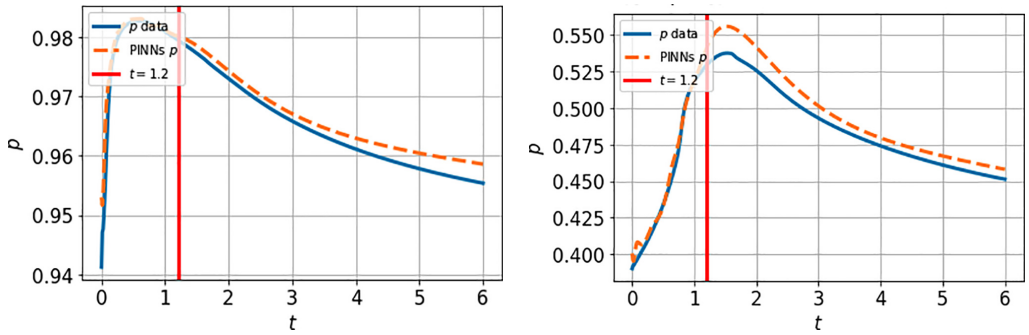


FIG. 16.

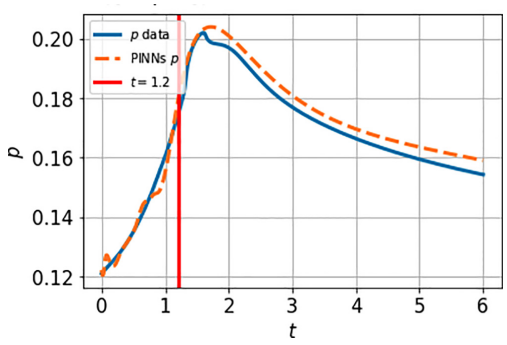


(c)

FIG. 16: Verification of S_w match at three observation locations. (a) S_w comparison at obs 1, (b) S_w comparison at obs 2, and (c) S_w comparison at obs 3.



(a) (b)



(c)

FIG. 17: Verification of p match at three observation locations. (a) p comparison at obs 1, (b) p comparison at obs 2, and (c) p comparison at obs 3.

using sparse measurements of S_w, p, and k. We also tested the method in different scenarios and it shows that the inversion can be achieved with sparse measurements. In addition, we are able to forecast the dynamics using the same framework so that the inversion and prediction can be done at the same time.

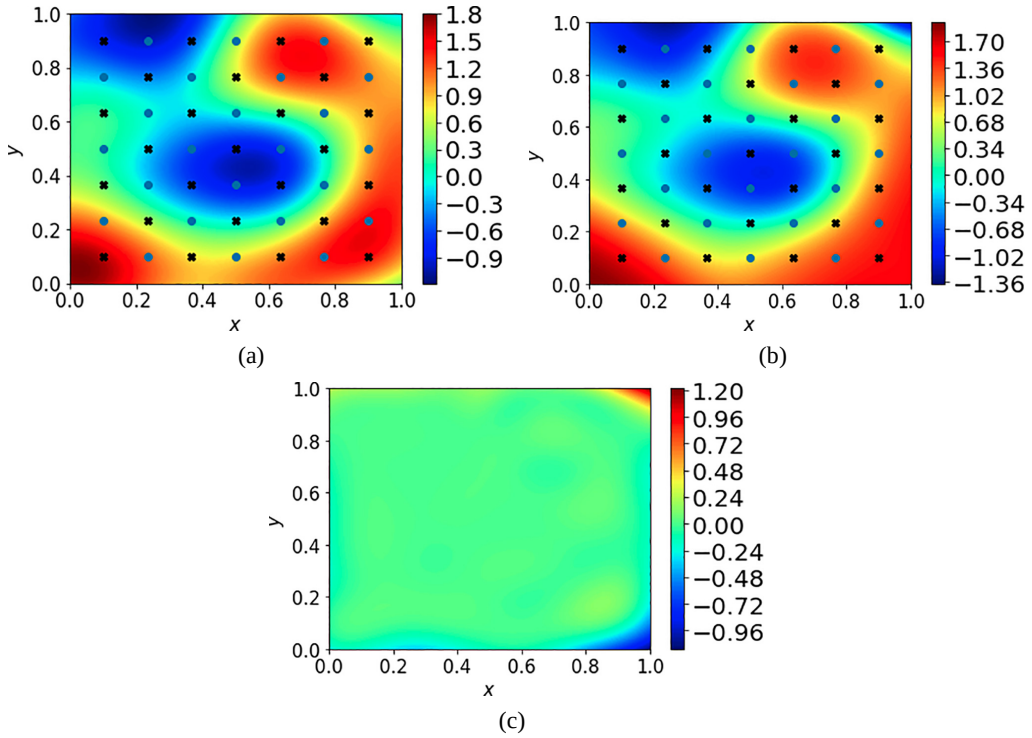


FIG. 18: Comparison of the inverted k field to the true k field. The blue dots the injection wells; the black crosses are the producer wells. (a) True k field in logarithm, (b) inverted k field in logarithm, and (c) absolute error of k.

In the future, we would like to extend the current work and apply the method on measurements with uncertainties, which are ubiquitous in petroleum engineering applications, and is another important aspect of the inverse problems. We would also like to explore different neural network architectures than those proposed in this work with the goal of further reducing the training time of PINNs.

REFERENCES

Alnæs, M., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M.E., and Wells, G.N., The FEniCS Project Version 1.5, *Arch. Numer. Software*, 2015. DOI: 10.11588/ans.2015.100.20553

Berg, J. and Nyström, K., A Unified Deep Artificial Neural Network Approach to Partial Differential Equations in Complex Geometries, *Neurocomput.*, vol. **317**, pp. 28–41, 2018.

Fraces, C.G., Papaioannou, A., and Tchelepi, H., Physics Informed Deep Learning for Transport in Porous Media. Buckley Leverett Problem, arXiv: 2001.05172, 2020.

Goswami, S., Anitescu, C., Chakraborty, S., and Rabczuk, T., Transfer Learning Enhanced Physics Informed Neural Network for Phase-Field Modeling of Fracture, *Theor. Appl. Fracture Mech.*, vol. **106**, p. 102447, 2020.

Haghighat, E., Raissi, M., Moure, A., Gomez, H., and Juanes, R., A Deep Learning Framework for Solution and Discovery in Solid Mechanics, arXiv: 2003.02751, 2020.

He, Q., Barajas-Solano, D., Tartakovsky, G., and Tartakovsky, A.M., Physics-Informed Neural Networks for Multiphysics Data Assimilation with Application to Subsurface Transport, *Adv. Water Res*., vol. **141**, no. 2, p. 103610, 2020.

Kharazmi, E., Zhang, Z., and Karniadakis, G.E., hp-VPINNs: Variational Physics-Informed Neural Networks with Domain Decomposition, arXiv: 2003.05385, 2020.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A., Automatic Differentiation in PyTorch, in *31st Conf. on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, December 4–9, 2017.

Raissi, M., Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations, *J. Mach. Learn. Res*., vol. **19**, no. 1, pp. 932–955, 2018.

Song, D.H. and Tartakovsky, D.M., Transfer Learning on Multifidelity Data, *J. Mach. Learn. Model. Comput*., vol. **3**, no. 1, pp. 31–47, 2022.

Tartakovsky, A.M., Marrero, C.O., Perdikaris, P., Tartakovsky, G.D., and Barajas-Solano, D., Learning Parameters and Constitutive Relationship with Physics Informed Deep Neural Networks, arXiv: 1808.03398, 2018.

Tchelepi, H.A. and Fuks, O., Limitations of Physics Informed Machine Learning for Nonlinear Two-Phase Transport in Porous Media, *J. Machine Learn. Model. Comput*., vol. **1**, no. 1, pp. 19–37, 2020.

Wang, N., Chang, H., and Zhang, D., Theory-Guided Auto-Encoder for Surrogate Construction and Inverse Modeling, *Comput. Methods Appl. Mech. Eng*., vol. **385**, p. 114037, 2021a.

Wang, N., Chang, H., Zhang, D., Xue, L., and Chen, Y., Efficient Well Placement Optimization Based on Theory-Guided Convolutional Neural Network, *J. Petrol. Sci. Eng*., vol. **208**, p. 109545, 2022.

Wang, S. and Perdikaris, P., Long-Time Integration of Parametric Evolution Equations with Physics-Informed Deeponets, arXiv: 2106.05384, 2021.

Wang, S., Wang, H., and Perdikaris, P., Learning the Solution Operator of Parametric Partial Differential Equations with Physics-Informed DeepONets, *Sci. Adv*., vol. **7**, no. 40, p. eabi8605, 2021b.

Wang, Y. and Lin, G., Efficient Deep Learning Techniques for Multiphase Flow Simulation in Heterogeneous Porousc Media, *J. Comput. Phys*., vol. **401**, p. 108968, 2020.

Yang, L., Zhang, D., and Karniadakis, G.E., Physics-Informed Generative Adversarial Networks for Stochastic Differential Equations, arXiv: 1811.02033, 2018.

Yang, M. and Foster, J.T., *hp*-Variational Physics-Informed Neural Networks for Nonlinear Two-Phase Transport in Porous Media, *J. Mach. Learn. Model. Comput*., vol. **2**, no. 2, pp. 15–32, 2021.

Zhou, Z., Zabaras, N., and Tartakovsky, D.M., Deep Learning for Simultaneous Inference of Hydraulic and Transport Properties, *Water Res. Res*., vol. **58**, no. 10, p. e2021WR031438, 2022.

Zhu, Y., Zabaras, N., Koutsourelakis, P.S., and Perdikaris, P., Physics-Constrained Deep Learning for High-Dimensional Surrogate Modeling and Uncertainty Quantification without Labeled Data, *J. Comput. Phys*., vol. **394**, pp. 56–81, 2019.