**hp-VARIATIONAL PHYSICS-INFORMED NEURAL NETWORKS FOR NONLINEAR TWO-PHASE TRANSPORT IN POROUS MEDIA**

**Mingyuan Yang* & John T. Foster**

Department of Petroleum and Geosystems Engineering, The University of Texas at Austin, Austin, Texas, USA, E-mail: mingyuanyang@utexas.edu

Original Manuscript Submitted: 2/8/2021; Final Draft Received: 6/2/2021

Neural networks (NN) have gained a lot attention recently in solving a wide range of computational physical problems. In this paper, we focus on solving a dynamic fluid-flow in a subsurface problem with the hp-variational physics-informed neural networks (hp-VPINNs) approach. The problem is governed by a nonlinear first-order hyperbolic partial differential equation (PDE) with initial and boundary conditions. The idea is to train a neural network representing the solution such that the underlying physical laws are honored while the constraints are satisfied. By employing the approach of hp-VPINNs, the forward problem is solved without any additional labeled data in the interior of the domain. It works for a case with the nonconvex flux functions in the PDE, where the solution contains shocks and mixed waves. In addition, we performed hp-refinement analysis on the problem and show that p refinement is suitable as it resolves the discontinuity in the solution. Finally, we investigated the inverse two-phase transport problem and solved for the nonlinear constitutive relation. With using sparse measurements as prior knowledge, the nonlinear constitutive relation was calculated and a solution over the entire computational domain was obtained.

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; Tchelepi and Fuks, 2020; Goswami et al., 2020; Haghighat et al., 2020; He et al., 2020; Kharazmi et al., 2020; Raissi, 2018; Tartakovsky et al., 2018; Yang et al., 2018; Zhu et al., 2019). More specifically, physics-informed neural networks (PINNs) have been used to target problems where engineering conservation equations and constitutive closure relationships are known while labeled data are sparse (Berg and Nyström, 2018; He et al., 2020; Raissi, 2018). The neural networks are defined with several hidden layers followed by the nonlinear activation functions so that they are capable of approximating the complex nonlinear solutions. PINNs provide an alternative approach to classical techniques such as finite element methods (FEMs) in solving partial-differential equations with the universal approximation properties of neural networks (Kharazmi et al., 2020; Raissi, 2018). In addition, it has been shown that the deep learning approach is also capable of integrating data or measurements as prior information so that forward and inverse problems can be solved via the same network architecture (He et al., 2020).

Deep learning has been used recently to model two-phase flow in subsurface porous media which is described by a nonlinear first-order hyperbolic PDE (Fraces et al., 2020; Tchelepi and
Fuks, 2020). This problem is known to be challenging for numerical methods as a steep saturation front (i.e., discontinuity) develops in the solution when the flux function is nonconvex (Tchelepi and Fuks, 2020). In Tchelepi and Fuks (2020), the authors used a naive PINN to solve this problem and showed that the method is not suitable. Only when an artificial diffusion term was added to the original conservation equation, could the neural networks solution approximate the true solution (Tchelepi and Fuks, 2020). We were inspired by this paper and decided to tackle the problem with a different formulation of PINNs.

In this work, we use the same two-phase flow problem as a model setting, but employ a variation of PINNs. In the framework of PINNs, it is crucial to design a formulation using the outputs of neural networks to represent residuals of the desired PDEs. With a proper formulation of loss functions, solving differential equations becomes an optimization problem. Inspired by methods of the variational/weighted residuals (Finlayson and Scriven, 1966), many loss function formulations have been proposed and applied in PINNs. In Sirignano and Spiliopoulos (2018), the authors developed the deep Galerkin method (DGM) which is based on least-squared methods. In Raissi (2018), the naive PINNs were developed to solve PDEs by enforcing the satisfaction of strong-form residuals at collocation points. In Kharazmi et al. (2019) the authors developed variational physics-informed neural networks (VPINNs), which are based on Galerkin methods, by using global polynomials as test functions. Later, $hp$ variational physics-informed neural networks ($hp$-VPINNs) was developed by using local polynomials as test functions in formulating the loss function Kharazmi et al. (2020). Both PINNs and $hp$-VPINNs can be categorized into Petrov-Galerkin methods. Under this classification, the test functions for PINNs are Dirac delta functions and the test functions for $hp$-VPINNs are piecewise polynomial functions (Kharazmi et al., 2020). More importantly, the $hp$-VPINN was developed to address the local steep solution and provides a better accuracy compared to naive PINNs (Kharazmi et al., 2020).

In the numerical results section, we present the solution to the nonlinear two-phase flow problem in porous media with the $hp$-VPINN approach. It shows that the problem can be solved to have a good match with the analytical solution by constructing the loss function in a weak-form residual fashion. Compared to the results in Tchelepi and Fuks (2020), the solution can be improved through a systematic refinement without adding an arbitrary diffusion term. In addition, the inverse problem is solved to find the nonlinear constitutive relations with the framework of $hp$-VPINNs by training two neural networks at the same time, one for the PDE variable and the other one for the unknown constitutive relation. The formulation of loss function is of a similar structure for both forward and inverse problems, which helps avoid the use of a separate optimization loop which is used in conventional approaches (Raissi, 2018; Tartakovsky et al., 2018). It has shown great success to solve for unknown coefficients/relations within the framework of PINNs (Mitusch et al., 2021; Raissi, 2018; Yang et al., 2021). We show the capability of our model in handling both clean and noisy measurements.

2. NONLINEAR TWO-PHASE TRANSPORT MODEL

In this work, we consider the classical Buckley-Leverett model which describes two incompressible immiscible fluids flowing through one-dimensional porous media (Fanchi, 2005; Tchelepi and Fuks, 2020). A nonwetting phase, e.g., oil ($o$), is displaced by a wetting phase, e.g., water ($w$), in the media whose absolute permeability and porosity are $k$ and $\phi$, we are interested in the dynamic water saturation distribution. Gravity and capillary effects are neglected here. In addition, the fluid is pumped into the media with a constant rate $q$, and no sources or sinks exist.
in the media. Therefore, total mass conservation and the mass conservation of the water can be written as:
\[
\nabla \cdot u = 0, \\
\Phi \frac{\partial S_w}{\partial t} + \nabla \cdot (f_w u_t) = 0,
\]
where \( S_w \) is the saturation of water, \( u_t = u_w + u_o \) is the total flux, and \( u_i \) is the flux of individual phases \((i = w, o)\) which are calculated with Darcy’s law in this work. \( f_w \) is the fractional flow of water which is defined as:
\[
\begin{align*}
    f_w &= \frac{u_w}{u_w + u_o}, \\
    u_i &= \frac{k_{ri}(S_w)}{\mu_i} \nabla p,
\end{align*}
\]
where \( k_{ri} \) and \( \mu_i \) are the relative permeability and viscosity of each phase. \( u_t \) is constant throughout the media since the fluids are incompressible. Similar to the work by Tchelepi and Fuks (2020), we perform the dimensionless transformation on the two variable \( x \) and \( t \), with introducing the following terms:
\[
\begin{align*}
    t_D &= \frac{u_t}{\Phi L}, \\
    x_D &= \frac{x}{L}.
\end{align*}
\]
In addition, we apply the chain rule on the fractional flow term. The one-dimensional problem becomes:
\[
\frac{\partial S_w}{\partial t} + \frac{df_w}{dS_w} \frac{\partial S_w}{\partial x} = 0, \quad x, t \in [0, 1] \times [0, T_D],
\]
where \( T_D > 0 \). Here we assume the irreducible water saturation to be 0.0, so that the initial and boundary conditions can be stated as:
\[
\begin{align*}
    S_w(t = 0) &= 0.0, \\
    S_w(x = 0) &= 1.0.
\end{align*}
\]
The selection of relative permeability functions and viscosity ratio \( M (\mu_w/\mu_o) \) determines the shape of the \( f_w \) function. In this work, we use the following setting as shown in Fig. 1:
\[
\begin{align*}
    k_{rw} &= S_w^2, \\
    k_{ro} &= (1 - S_w)^2, \\
    M &= 10.0.
\end{align*}
\]
This setting would bring a nonconvex \( f_w \) function and a discontinuity in the \( S_w \) solution.
The exact solution to the problem can be obtained in a semianalytical fashion. To solve for the water saturation \( S_0 \) of a point \( x = x_0 \) at \( t = t_n \) in the unswept region, a nonlinear equation needs to be solved.
FIG. 1: The relative permeabilities and water fractional flow as functions of the water saturation

\[ x_0 = f'_w(S_{wf})f'_w(S_0)t_n, \]

where \( f'_w \) is the derivative of the \( f_w \) function with respect to \( S_w \); \( S_{wf} \) is the water saturation at the frontal, which can be calculated by solving the following equation:

\[ f_w(S_{wf}) - S_{wf}f'_w(S_{wf}) = 0. \]

The complete derivation can be found in the book by Fanchi (2005).

3. hp-VARIATIONAL PHYSICS-INFORMED NEURAL NETWORKS

In the framework of PINNs, we embed the conservation equations, constitutive laws, and physical constraints into the loss function. The loss function is minimized by using optimizers in an iterative fashion, and each iteration is called an epoch. During the training process, the weights and bias of the NN keep being updated. After a tolerance is reached, the NN which satisfies the physical constraints can serve as the solution that we desire.

VPINNs is a variation of PINNs, in which the loss function is constructed by using a variational residual form Kharazmi et al. (2020). It extends the capability of PINNs by using flexible test functions, and enables systematic local refinements, e.g., \( h \) and \( p \) refinement. We briefly explain the construction of \( hp \)-VPINNs in the rest of this section.

Here we consider a general problem govern by PDEs and constraints for demonstration,

\[ \mathcal{L}u(x, t) = f(x, t), \quad x, t \in \Omega \times (0, T], \quad (1) \]
\[ u(x, t) = h(x, t), \quad x, t \in \partial \Omega_h \times (0, T], \quad (2) \]
\[ \frac{\partial u(x, t)}{\partial x} = g(x, t), \quad x, t \in \partial \Omega_g \times (0, T], \quad (3) \]
\[ u(x, 0) = l(x), \quad x \in \Omega, \quad (4) \]

where \( \Omega \) is the domain bounded by the boundaries \( \partial \Omega \), \( u(x, t) \) is the solution to the PDEs on \( \Omega \times (0, T] \), and \( \mathcal{L} \) is an operator that is applied on \( u(x, t) \). \( \Omega_h \) is part of the boundary where the
essential boundary condition \( h(x, t) \) is applied, and \( \Omega_g \) is the part of boundary where the natural boundary condition \( g(x, t) \) is applied. \( l(x) \) is the initial condition of the PDEs.

Then, we construct a fully connected neural network \( u_{NN} \) which comprises \( m \) layers with \( n \) neurons in each layer followed by an activation function \( \sigma \). The neural networks can be expressed as:

\[
\begin{align*}
  u_{NN}(x, t; W, b) &= T^m \circ T^{m-1} \circ \cdots \circ T^2 \circ T^1(x, t), \\
  T^j(\cdot) &= \sigma^j(W^j \times \cdot + b^j), \quad j = 1, 2, \cdots, n
\end{align*}
\]

where \( W^j \) and \( b^j \) are weights and bias of layer \( j \); \( (\cdot) \) is the output from the previous layer. A schematic of fully connected neural networks is shown in Fig. 2.

The strong form residuals of Eqs. (1)–(4) can be defined as

\[
\begin{align*}
  r_d(u_{NN}) &= L u_{NN} - f, \quad x, t \in \Omega \times (0, T], \quad (5) \\
  r_e(u_{NN}) &= u_{NN} - h, \quad x, t \in \partial \Omega_h \times (0, T], \quad (6) \\
  r_n(u_{NN}) &= \frac{\partial u_{NN}}{\partial x} - g, \quad x, t \in \partial \Omega_g \times (0, T], \quad (7) \\
  r_l(u_{NN}) &= u_{NN} - l, \quad x \in \Omega. \quad (8)
\end{align*}
\]

Ideally, we would like to obtain a neural networks solution \( u_{NN} \) which makes the above residuals be zeros throughout the entire domain. In the framework of \( hp \)-VPINNs, we drive the residuals to zeros in the following approach. First, we construct the weak form residual of the domain by multiplying the strong form residual of the domain by a set of test functions \( v_j \), and then integrate the resulting products over the domain. Each integration also needs to be zero.

\[
\begin{align*}
  R^j_d(u_{NN}) &= \int_{\Omega \times [0, T]} (L u_{NN} - f) v_j dx dt = 0, \quad (9)
\end{align*}
\]

where \( j = 1, 2, \cdots, N_t \) and \( N_t \) is the number of test functions used. If we choose the \( v \) to be local functions which are only defined on nonoverlapping subdomains,
\[
v_j = \begin{cases} 
v_j & \text{over subdomain } j \\
0 & \text{over subdomain } i, i \neq j
\end{cases}
\]

the overall loss function can be defined as

\[
L = w_d \sum_{i=1}^{N_s} \sum_{j=1}^{N_s} \| R_{ij}^d \| + \frac{w_e}{N_c} \sum_{i=1}^{N_c^e} \| r_e \| + \frac{w_n}{N_c} \sum_{i=1}^{N_c^n} \| r_n \| + \frac{w_t}{N_c} \sum_{i=1}^{N_c^t} \| r_t \|,
\]

where \( \| \cdot \| \) is the \( L_2 \) norm, \( N_s \) is the number of subdomains, and \( N_c^e \) and \( N_c^n \) are the respective number of collocation points on the part of the boundaries with essential boundary conditions and natural boundary conditions. \( N_c^i \) is the number of collocation points for the initial conditions. \( w_d, w_e, w_n, \) and \( w_t \) are respective weights for each residual term.

The \( L \) operation on \( u_{NN} \) with respect to \( x \) and \( t \) can be achieved by using autodifferentiation or numerical differentiation methods. In the case of deep networks, it would be very hard to calculate the integration in the residual term analytically, so we employ the Gauss quadrature rules to approximate the integration. The number of subdomains \((h \text{ refinement})\) and the number of test functions used in each subdomain \((p \text{ refinement})\) may vary; this is where the \( hp \) comes from in the name of the method. \( u_{NN}(x, t) \) converges to the true solution \( u(x, t) \) as we train the neural networks by minimizing the loss function.

4. NUMERICAL RESULTS

4.1 Model Setup

In this section, we present the results of solving the one-dimensional Buckley-Leverett problem with \( hp \)-VPINNs. The test functions we employ here are Legendre polynomials which are shown in Fig. 3.

Here we use the backward Euler method to resolve the first-order difference in time (implicit method) and only define the variational residuals for the spatial dimension. We discretize the time dimension into \( N_{elt} \) pieces, so that the whole domain \([0, 1] \times [0, T]\) is structurally divided into \( N_{elx} N_{elt} \) pieces, and each test function \( v_i(x) \) is mapped onto the subdomain \([x_{elx}, x_{elx+1}]_j\), where \( e_x = 1, 2, \cdots, N_{elx}, j = 1, 2, \cdots, N_{elt}; N_{elx} \) is the number of subdomains in the spatial dimension. Now the residual term in Eq. (9) can be written as

\[
R_d^i = \sum_{e_x=1}^{N_{elx}} \sum_{e_t=1}^{N_{elt}} R_{e_x e_t}^i, \quad i = 1, 2, \cdots, N_p,
\]

\[
R_{e_x e_t}^i = \int_{x_{elx}}^{x_{elx+1}} \left( \frac{\partial u_{NN}}{\partial t} + \frac{df_w}{ds_w} \frac{\partial u_{NN}}{\partial x} \right) v_i(x) dx, \quad i = 1, 2, \cdots, N_p,
\]

where \( N_p \) is the number of test functions, \( u_{NN}(x, t) \) is the approximation of the solution \( S_w(x, t) \), and \( \Delta t \) is the time step which is \( 1/N_{elt} \) in this case. The overall loss function becomes:

\[
L = w_d \sum_{i=1}^{N_{elx}} \sum_{j=1}^{N_{elt}} \sum_{k=1}^{N_p} \| R_{ijk}^d \| + \frac{w_e}{N_c} \sum_{i=1}^{N_c^e} \| r_e \| + \frac{w_n}{N_c} \sum_{i=1}^{N_c^n} \| r_n \| + \frac{w_t}{N_c} \sum_{i=1}^{N_c^t} \| r_t \|.
\]
Neural Networks for Two-Phase Transport in Porous Media

(a) Five polynomials used in each of the two subdomains

(b) \( p \) refinement: ten polynomials used in each of the two subdomains

(c) \( h \) refinement: five polynomials used in each of the five subdomains

FIG. 3: \( hp \) refinement illustration with Legendre polynomials

Here \( w_d, w_e, \) and \( w_i \) are assigned to be 1.0. \( u_{NN} \) is defined as a fully connected neural network with five hidden layers and 20 neurons in each hidden layer. This structure of neural networks is used throughout this work. The hyperbolic tangent function (tanh) is used as the activation function, and we apply the Sigmoid function on the output to project the solution back to the range of \([0, 1]\). We use the LBFGS optimizer with a history size of 100. We experimented with two optimizers in our study, Adam and LBFGS. It has been reported that LBFGS has a superior performance with a better convergence rate (He et al., 2020; Tartakovsky et al., 2018), so that we stick with LBFGS in the analysis of the forward problem. It should be noted that the LBFGS optimizer is sensitive to the initialization of the neural networks, so that some of the initializations may lead to the failure of converging to the solution. The Xavier normal initialization strategy is adopted throughout this study and it would return neural networks with different weights for different random seeds. We did our analysis of the model consistency and \( hp \) refinements with the working cases under different initializations. The simulation is performed up to \( T = 1.0 \) for all the cases in this study. The \( hp \)-VPINNs method in this work is implemented with PyTorch (Paszke et al., 2017) on Kaggle NVidia K80 GPUs.
4.2 Analysis of the \(hp\)-VPINNs Solution

In this section, we solve the problem by using five subdomains with 40 test functions. In addition, 80 quadrature points are used in each subdomain to calculate the integration. The time dimension is divided into 1000 pieces, so that the time step is \(1 \times 10^{-3}\). Two hundred evenly spaced collocation points are used for the initial condition and boundary condition, respectively. This step size of time is used for all the cases in this work. The training was performed for 200 epochs and the overall loss decreased to \(5 \times 10^{-4}\).

The simulation results are presented in Fig. 4. From the \(S_w\) solution color map, we can see a clear discontinuity which represents the shock of the water frontal. The blue region toward the bottom right is unswept and the red region toward the top left is swept by the injected water. The \(S_w\) solutions at \(t = 0.1, 0.25, \) and \(0.4\) are chosen, and we compare them to the exact solutions. The solutions match very well and the shocks are captured without adding any extra second-order diffusion terms.

![Color map of the \(S_w\) solution in the \(x-t\) domain at the end of the training](image1)

![\(S_w\) distribution at \(t = 0.1\)](image2)

![\(S_w\) distribution at \(t = 0.25\)](image3)

![\(S_w\) distribution at \(t = 0.4\)](image4)

**FIG. 4:** The \(S_w\) solution and the comparison to the exact solutions at three different time points
In order to show the quantitative mismatch to the analytical solution, we define the mismatch to be:

\[ M = \sum_{i=1}^{N} \left[ (S_{ei}^c - S_{ci}^c)^2 |_{t=0.1} + (S_{ei}^c - S_{ci}^c)^2 |_{t=0.25} + (S_{ei}^c - S_{ci}^c)^2 |_{t=0.4} \right], \]

where \( N = 800 \); 800 points are equally spaced between 0 and 1 in \( x \) dimension where \( S_w \) is evaluated. \( S_{ei}^c \) and \( S_{ci}^c \) are the exact solution of \( S_w \) and our calculated solution. This expression provides us with a quantitative sense of the overall mismatch at three time slices \( t = 0.1, 0.25, \) and 0.4 which are shown in Fig. 4. We ran our simulations ten times with different initialization and the results are listed in Table 1. This shows the consistency of our prediction model.

4.3 hp Refinement Study

In this section, we present the results using a different number of subdomains (\( h \) refinement) and test functions (\( p \) refinement). The structure of the neural networks and the hyperparameters remains the same as in the last example. As for the \( p \) refinement case, we use only one subdomain with 5, 20, and 80 test functions.

The convergences of loss function for three cases are shown in Fig. 5, and the comparison to the exact solutions are shown in Fig. 6. It should be noted that the loss function value does not necessarily mean the accuracy of the model as the loss function formulation is different when using different test functions, so that it only indicates its individual convergence. We can see that the approximation becomes closer to the exact solutions as we use more test functions (higher order polynomials). For this simple one-dimensional problem, one subdomain with 80 test functions is sufficient to provide an accurate solution. In Table 2, we show the averaged mismatch and elapsed time of ten simulations with different initializations.

As for the \( h \) refinement case, we use only one test function (the piecewise constant function) on ten, 100 and 200 subdomains. The convergences of loss function for three cases are shown in Fig. 7 and the comparisons to the exact solutions are shown in Fig. 8.

<table>
<thead>
<tr>
<th>No.</th>
<th>Loss</th>
<th>Time, s</th>
<th>Mismatch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.96 \times 10^{-4}</td>
<td>194</td>
<td>13.7</td>
</tr>
<tr>
<td>2</td>
<td>4.79 \times 10^{-4}</td>
<td>187</td>
<td>13.7</td>
</tr>
<tr>
<td>3</td>
<td>4.83 \times 10^{-4}</td>
<td>203</td>
<td>13.8</td>
</tr>
<tr>
<td>4</td>
<td>4.92 \times 10^{-4}</td>
<td>204</td>
<td>13.7</td>
</tr>
<tr>
<td>5</td>
<td>5.02 \times 10^{-4}</td>
<td>144</td>
<td>13.9</td>
</tr>
<tr>
<td>6</td>
<td>4.91 \times 10^{-4}</td>
<td>141</td>
<td>13.7</td>
</tr>
<tr>
<td>7</td>
<td>5.16 \times 10^{-4}</td>
<td>173</td>
<td>13.8</td>
</tr>
<tr>
<td>8</td>
<td>5.16 \times 10^{-4}</td>
<td>175</td>
<td>13.8</td>
</tr>
<tr>
<td>9</td>
<td>4.72 \times 10^{-4}</td>
<td>152</td>
<td>13.8</td>
</tr>
<tr>
<td>10</td>
<td>4.79 \times 10^{-4}</td>
<td>187</td>
<td>13.9</td>
</tr>
</tbody>
</table>

TABLE 1: Loss, elapsed time and mismatch after 200 epochs for ten simulations. The first simulation in the table produces the results in Fig. 4.
Again, we observe that as we use more subdomains, the solutions become closer to the exact solutions. However, we can still see a mismatch near the water frontal for the case with 200 subdomains. The reason could be that the discontinuity cannot be properly captured locally without using higher-order polynomials. Therefore, it would be necessary to employ a large number of test functions if we knew that discontinuities reside in the solution. In Kharazmi et al. (2020), the authors show that $hp$ refinement can be performed locally, which helps avoid the unnecessary computational expense. In Table 3, we show the averaged mismatch and elapsed time of ten simulations with different initializations.

4.4 Inverse Problem with Measurements

4.4.1 Cases with Exact Measurements

The solution to the one-dimensional Buckley-Leverett problem is determined by the constitutive relation of water fractional flow function $[f_w(S_w)]$. Here we define the inverse problem as solving for $f_w(S_w)$ with measurements of water saturation ($S_w$) in the $x$–$t$ domain.

![Graph](image_url)

**FIG. 6.**
Neural Networks for Two-Phase Transport in Porous Media

FIG. 6: Comparison to the exact solutions at three different time points with the number of test function $N_t = 5, 20, \text{ and } 80$

Volume 2, Issue 2, 2021
TABLE 2: Averaged mismatch and elapsed time after 200 epochs with different number of test functions

<table>
<thead>
<tr>
<th>No. of Test Functions</th>
<th>Averaged Mismatch</th>
<th>Averaged Time, s</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>25.1</td>
<td>83</td>
</tr>
<tr>
<td>20</td>
<td>14.3</td>
<td>85</td>
</tr>
<tr>
<td>80</td>
<td>13.8</td>
<td>109</td>
</tr>
</tbody>
</table>

FIG. 7: Loss function value during training of the model with three different numbers of subdomains. The \( x \) axis denotes the epoch number of the LBFGS optimization method.

We generated our synthetic data with the forward model described in Section 4.2. The measurements are taken by calculating the \( S_w \) at selected scatter points with coordinates of \( (x_m, t_m) \), where \( m = 1, 2, \ldots, N_m \) and \( N_m \) is the total number of the selected points. We consider two scenarios of scatter points sampling; they are shown in Fig. 9.

We define a fully connected neural network \( f_{NN} \) with one input and one output to approximate the \( f_w \) function. \( f_{NN} \) comprises three hidden layers and ten neurons in each layer. Again, we use tanh as the activation function, and apply the Sigmoid function to the output of \( f_{NN} \). The structure of neural networks for \( S_w \) remains the same as in the forward problem section. Now the loss function is constructed with two neural networks, \( u_{NN} \) and \( f_{NN} \). Hence, the residual term in Eq. (10) becomes

\[
R_{e,e_i} = \int_{x_e}^{x_{e+1}} \left( \frac{\partial u_{NN}}{\partial t} + \frac{d f_{NN}}{d S_w} \frac{\partial u_{NN}}{\partial x} \right) v_i(x) dx, \quad i = 1, 2, \ldots, N_p.
\]

In addition to the change of the domain residual term, there are two new terms in the loss function of this inverse problem. They can be written as follows:

\[
R_{data} = \frac{1}{N_m} \sum_{i=1}^{N_m} \|u_{NN}(x_m, t_m) - S_m\|,
\]

\[
R_c = \|f_{NN}(0)\| + \|f_{NN}(1) - 1\|.
\]
FIG. 8.
TABLE 3: Averaged mismatch and elapsed time after 200 epochs with different number of subdomains

<table>
<thead>
<tr>
<th>No. of Subdomains</th>
<th>Averaged Mismatch</th>
<th>Averaged Time, s</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>15.8</td>
<td>82</td>
</tr>
<tr>
<td>100</td>
<td>14.4</td>
<td>203</td>
</tr>
<tr>
<td>200</td>
<td>14.1</td>
<td>379</td>
</tr>
</tbody>
</table>

FIG. 8: The comparison to the exact solutions at three different time points with the number of subdomain $N_s = 10, 100, \text{ and } 200$

FIG. 9: Two scenarios of scatter points sampling

(a) 20 random points in the $x$–$t$ domain
(b) 20 evenly spaced points at $x = 1$

The first one denotes the mismatch of the solution to the measurements at the sampled points and the second one denotes the constraints of the $f_w$ function. Therefore, the overall loss function can be defined as

$$L_b = w_1 L(u_{NN}, f_{NN}) + w_2 R_{\text{data}} + w_3 R_c,$$

where $L$ is the loss function of the forward problem defined in Eq. (11). $w_1$, $w_2$, and $w_3$ are respective weights which are assigned to be 1 in this study. The loss function $L_b$ is minimized by
training the two neural networks together. In the framework of \( hp \)-VPINNs, the data and prior knowledge can be integrated into the loss function in the same way as boundary conditions and initial conditions, so the inverse problem can be solved with a minor modification of the model structure of the forward problem. In this section, the Adam optimizer is used with the learning rate of \( 10^{-3} \).

![Graphs and diagrams showcasing the approximation of the two scenarios and the pointwise error.](image)

**FIG. 10:** \( f_w \) approximation, \( S_w \) approximation, and the pointwise \( S_w \) error of the two scenarios at the end of training
As seen in Fig. 10, an accurate approximation of $f_w$ is obtained after 10,000 epochs of training. In addition, an accurate approximation of $S_w$ in the whole domain is also obtained as a byproduct with the provided sparse measurements. The $S_w$ solution obtained in Section 4.2 is used as the reference solution here for comparison. The second scenario can be seen as $S_w$ measurements being taken at the outlet of a 1D core. Hence, this inverse framework provides a way to infer the $f_w$ curve with data from the lab.

### 4.4.2 Cases with Noisy Measurements

The same $S_w$ measurements are used as in the last section; however, we further assume that those measurements are not exact and the error at each point obeys a distribution. We would like to solve for the constitutive relation between $f_w$ and $S_w$ corresponding to the measurements with uncertainties. For both sampling scenarios, we investigated two error distributions as follows:

1. $\epsilon \sim \mathcal{N}(0, 0.02^2)$,
2. $\epsilon \sim \mathcal{N}(0, 0.1^2)$.

We used the bootstrapping approach to generate multiple realizations of the measurements $S'_w$ where the errors are drawn from the distributions, e.g., $S'_w = S_w + \epsilon$. Then we repeated the workflow in the inverse part to obtain multiple $f_w$ functions with $S'_w$. In the end, we calculated the mean of those functions and that is our $f_w$ estimation given the noisy data. A schematic of the error distributions is shown in Fig. 11.

The $f_w$ approximations of two sampling scenarios with noisy measurements are shown in Fig. 12. For all cases, we obtained 100 realizations of $f_w$ and the mean predicted $f_w$ matches the true $f_w$ fairly well. It is expected that measurements with large noise result in a wide bandwidth of solutions, but the average of all predictions can still give us a reasonable $f_w$ approximation. This approach is not limited to the normal distribution of error used in this section, in which the prior knowledge of the error can be integrated into the workflow to generate noisy measurements.

![Fig. 11: A schematic of the two error distributions](image-url)
5. CONCLUSION AND FUTURE WORK

We investigated the application of the $hp$-VPINNs method to the nonlinear two-phase transport in one-dimensional porous media. The solution to the problem can be obtained without using any additional labeled data other than the initial and boundary conditions. Our study shows that the $p$ refinement is suitable for this problem as there is a discontinuity in the solution. We also investigated calculating the constitutive relation with sparse available measurements and showed that the framework of PINNs can be used for the forward and inverse problems by applying a minor modification on the loss function. Furthermore, we have shown that the inverse workflow can handle noisy data and produce good estimations of the unknown constitutive relation of interest.

In the future, we would like to extend this approach to solve the two-phase transport problem in higher spatial dimensions through heterogeneous porous media, which is commonly seen in real-life subsurface conditions. In addition, it is also of great interest to us to explore the capability of this approach in solving the inverse problem to find the underlying physics.
REFERENCES


*Journal of Machine Learning for Modeling and Computing*