

Water Resources Research

RESEARCH ARTICLE

10.1029/2020WR029479

Key Points:

- Physics-informed neural network (PINN) method is proposed for forward and backward advection-dispersion equations
- The physics-informed neural network (PINN) method has several advantages over some grid-based discretization methods for high Péclet number problems
- The physics-informed neural network (PINN) method is accurate for the considered backward advection-dispersion equations (ADEs) that otherwise must be treated as computationally expensive inverse problems

Correspondence to:

Q. He and A. M. Tartakovsky,
qizhi.he@pnnl.gov;
amt1998@illinois.edu


Citation:

He, Q., & Tartakovsky, A. M. (2021). Physics-informed neural network method for forward and backward advection-dispersion equations. *Water Resources Research*, 57, e2020WR029479. <https://doi.org/10.1029/2020WR029479>

Received 16 DEC 2020

Accepted 19 JUN 2021

Physics-Informed Neural Network Method for Forward and Backward Advection-Dispersion Equations

QiZhi He¹  and Alexandre M. Tartakovsky^{1,2} 

¹Physical and Computational Sciences Directorate, Pacific Northwest National Laboratory, Richland, WA, USA,

²Department of Civil and Environmental Engineering, University of Illinois Urbana-Champaign, Urbana, IL, USA

Abstract We propose a discretization-free approach based on the physics-informed neural network (PINN) method for solving the coupled advection-dispersion equation (ADE) and Darcy flow equation with space-dependent hydraulic conductivity $K(\mathbf{x})$. In this approach, $K(\mathbf{x})$, hydraulic head, and concentration fields are approximated with deep neural networks (DNNs). We assume that $K(\mathbf{x})$ is given by its values on a grid, and we use these values to train the K DNN. The head and concentration DNNs are trained by minimizing the residuals of the flow equation and ADE and using the initial and boundary conditions as additional constraints. The PINN method is applied to one- and two-dimensional forward ADEs, where its performance for various Péclet numbers (Pe) is compared with the analytical and numerical solutions. We find that the PINN method is accurate with errors of less than 1% and outperforms some conventional discretization-based methods for large Pe . Next, we demonstrate that the PINN method remains accurate for the backward ADEs, with the relative errors in most cases staying under 5% compared to the reference concentration field. Finally, we show that when available, the concentration measurements can be easily incorporated in the PINN method and significantly improve (by more than 50% in the considered cases) the accuracy of the PINN solution of the backward ADE.

1. Introduction

Advection-dispersion equations (ADEs) are an important class of partial differential equations (PDEs) that are used to describe transport phenomena in the fields of hydrology (Ingham & Ma, 2005) and hydrogeology (Patil & Chore, 2014).

We consider both forward and backward ADEs. In the former case, the initial conditions at time t_0 as well as boundary conditions are specified, and the solutions of forward ADEs are found for later times. This is a well-posed problem with well-established numerical methods. Nevertheless, there are some challenges in numerically solving forward ADEs, mainly associated with the advection-dominated problems. In backward ADEs, the concentration is known at later (terminal) times and solutions are sought for earlier times. Backward ADEs arise in the source identification problems (Neupauer & Wilson, 1999; Wilson & Liu, 1994) and could lead to numerically unstable grid-based solutions that require some form of regularization (Xiong et al., 2006) or should be solved as an inverse problem that is computationally more expensive because it requires solving forward problems multiple times (Atmadja & Bagtzoglou, 2001).

Numerical discretization-based methods, including the finite element (FE) and finite difference (FD) methods, are commonly used for solving the Darcy flow equation and ADE. Discretization-based methods approximate the PDE solution with its values at a set of grid points distributed over the spatiotemporal domain. The discrete solution is obtained by discretizing the time and spatial derivatives of state variables. It is worth noting that the space-dependent parameters such as hydraulic conductivity are usually given not as a continuous field but as a set of values at the grid points.

The combination of an advection (first-order) term and a dispersion (second-order) term in ADEs present several challenges for numerical methods. For example, for advection-dominated transport (i.e., Péclet number $Pe \gg 1$), the numerical solutions can develop oscillations (over- or undershoot) or numerical dispersion (Huyakorn, 2012; Pinder & Gray, 2013). These two numerical issues are closely related, and a numerical scheme developed to reduce numerical dispersion generally causes oscillation, whereas the suppression of oscillation comes at the cost of increased numerical dispersion (Wang & Lacroix, 1997).

Errors in numerical methods for ADEs can be reduced by using a smaller grid size, but this results in a higher computational cost. Several methods have been developed to reduce errors for a given grid size (Ewing & Wang, 2001), including the upwind methods (Brooks & Hughes, 1982; Heinrich et al., 1977; Noorishad & Mehran, 1982). Based on the “optimal” upwind concept, the streamline upwind Petrov-Galerkin (SUPG) (Brooks & Hughes, 1982; Hughes et al., 1986), the Galerkin least squares (Hughes et al., 1989), and the unusual stabilized FE (Franca & Farhat, 1995) methods have been proposed to increase the stability of the standard polynomial FE methods by consistently adding diffusive terms to the variational forms of ADEs. Higher-order schemes have also been used to improve the accuracy of FD methods for ADEs (Cecchi & Pirozzi, 2005; Ding & Zhang, 2009; Noye & Tan, 1988; Rigal, 1994).

In this work, we obtain solutions of ADEs and the Darcy equation using the physics-informed neural network method (PINN) (Mao et al., 2020; Raissi et al., 2019). Recently, the PINN method was applied for estimating hydraulic conductivity $K(\mathbf{x})$, steady-state hydraulic head $h(\mathbf{x})$, and concentration $u(\mathbf{x})$ fields using sparse measurements of these fields (He et al., 2020). In the PINN method for parameter and state estimation, the deep neural networks (DNNs) \hat{k} , \hat{h} , and \hat{u} are used to approximate the K , h , and u fields, respectively. These DNNs are trained using K , h , and u measurements constrained by the steady-state ADE and the Darcy flow equation. In many applications, system states (including hydraulic head and concentration) change over time and can be easily observed as time series at fixed locations. The assimilation of time-varying data in the PINN framework requires constraining the DNN training with time-dependent governing equations. In this paper, we demonstrate that the PINN method for training \hat{k} , \hat{h} , and \hat{u} DNNs given a known $K(\mathbf{x})$ field and constrained by the Darcy equation and time-dependent ADE with known initial and boundary conditions is equivalent to solving the forward Darcy and ADE equations. We also show that when the concentration is known at terminal time T , the PINN method approximately recovers the backward solution of the ADE for time less than T . Finally, we show that the measurements of u (when available) can be easily incorporated in the PINN method and significantly improve the accuracy of the PINN solution. The accuracy of the PINN method is investigated via comparison with analytical and numerical solutions.

The PINN method approximates the solution of a PDE with a DNN. In this, the PINN method relies on the approximating property of DNNs—given a sufficient size, DNNs can represent any continuous and bounded functions (Cybenko, 1989; Hornik et al., 1989). Different from discretization-based methods, time and space derivatives in the PINN method are evaluated using automatic differentiation of the DNNs (Baydin et al., 2015). Then, the DNNs’ coefficients are computed by minimizing the loss function that is the sum of the residuals of both the PDEs and initial and boundary conditions. The PINN method has been used to solve various PDEs, including the steady-state diffusion equations with space- and state-dependent diffusion coefficients (Tartakovsky et al., 2020), continuity and momentum conservation equations describing the flow of Newtonian (Mao et al., 2020; Raissi et al., 2020; Sun et al., 2020) and non-Newtonian fluids (Kissas et al., 2020; Reyes et al., 2020), advection-diffusion equations (Cai et al., 2021; Dwivedi & Srinivasan, 2020; Khodayi-Mehr & Zavanos, 2019; Pang et al., 2019; Yadav et al., 2016), and equations of solid mechanics (Rao et al., 2020; Samaniego et al., 2020). The PINN method has also been used to develop surrogate models where a DNN-based solution is obtained as a function of parameters describing fluid properties and initial and boundary conditions (Sun et al., 2020). As far as we know, this is the first study where the PINN method is used for solving an ADE with the velocity variations and the anisotropic dispersion tensor.

The main idea in the PINN method of using an artificial neural network as an approximation function to solve a PDE dates back to the 1990s, when shallow feed-forward neural networks were used to solve initial-boundary value problems (Lagaris et al., 1997, 2000; Lee & Kang, 1990). The new interest in using artificial neural networks for solving PDEs is mostly attributed to advances in automatic differentiation (Baydin et al., 2015), optimization methods (Byrd et al., 1995; Goodfellow et al., 2016; Kingma & Ba, 2014), and specialized hardware (such as a graphic processing unit) that significantly simplified implementation of the DNN-based methods and enabled the training of large DNNs, which might be necessary to approximate the solutions of PDEs.

Our paper is organized as follows. Section 2 presents the PINN method for ADEs. In Sections 3 and 4, we present the PINN solutions of forward and backward ADEs, respectively. Section 5 describes how data can be assimilated in the PINN method to increase the accuracy of ADE solutions. Finally, the conclusions are provided in Section 6.

2. Problem Formulation

2.1. Advection-Dispersion Equation

We assume that transport in porous media is described by the ADE:

$$\begin{cases} u_t + \nabla \cdot (-D \nabla u + \mathbf{v}u) = s, & (\mathbf{x}, t) \in \Omega \times (0, T) \\ u = g_D, & (\mathbf{x}, t) \in \partial\Omega_D \times (0, T) \\ -D \nabla u \cdot \mathbf{n} = g_N, & (\mathbf{x}, t) \in \partial\Omega_N \times (0, T) \\ u(\mathbf{x}, t = 0) = u_0, & (\mathbf{x}, t) \in \Omega \end{cases} \quad (1)$$

where $\Omega \in \mathbb{R}^d$ is the spatial domain with the boundary $\partial\Omega$, d is the number of spatial dimensions, t is time, $u(\mathbf{x}, t)$ is the concentration, $\partial\Omega_D$ and $\partial\Omega_N$ are the Dirichlet and Neumann boundaries, respectively, $\mathbf{v}(\mathbf{x})$ is the average pore velocity (LT^{-1}), s is the source term ($ML^{-3}T^{-1}$), $g_D(\mathbf{x}, t)$ and $g_N(\mathbf{x}, t)$ are the prescribed concentration and mass flux at the Dirichlet and Neumann boundary conditions (BCs), respectively, and $u_0(\mathbf{x})$ is the initial condition (IC). The dispersion coefficient $D [L^2T^{-1}]$ is given as

$$D = D_w \tau \mathbf{I} + \alpha \|\mathbf{v}\|_2, \quad (2)$$

where D_w is the diffusion coefficient, τ is the tortuosity of the medium, \mathbf{I} is the identity tensor, and α is the dispersivity tensor with the principal components α_L and α_T . In Sections 3.1–3.3, we study a special case of Equation 1, where \mathbf{v} is known and constant in space and time and $D = \kappa \mathbf{I}$ with a scalar coefficient κ . In Section 3.4, we consider Equation 1, with \mathbf{v} given by the solution of the Darcy equation with space-varying conductivity. In this test, we assume that the flow is steady and the solution is diluted, such that the velocity \mathbf{v} is independent of u and t .

2.2. PINN Approximation of the ADE Solutions

Below, we formulate the PINN method for the ADE Equation (1). More discussions on the PINN method can be found in (He et al., 2020; Lu et al., 2019; Mao et al., 2020; Raissi et al., 2019, 2020; Tartakovsky et al., 2020). In the PINN method, the solution $u(\mathbf{z})$ is approximated with a DNN as

$$u(\mathbf{z}) \approx \hat{u}(\mathbf{z}, \theta) = y_{n_l+1} \left(y_{n_l} \left(\dots \left(y_2(\mathbf{z}) \right) \right) \right), \quad (3)$$

where \hat{u} is a DNN approximation of u , $\mathbf{z} = [x_1, \dots, x_d, t]$ is the space-time coordinate vector, and

$$\begin{aligned} y_2(\mathbf{z}) &= \sigma(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1) \\ y_{i+1}(y_i) &= \sigma(\mathbf{W}_i y_i + \mathbf{b}_i), i = 1, \dots, n_l - 1 \\ y_{n_l+1}(y_{n_l}) &= \mathbf{W}_{n_l} y_{n_l} + \mathbf{b}_{n_l}. \end{aligned} \quad (4)$$

Here, n_l denotes the number of hidden layers, σ is the predefined activation function, and \mathbf{W}_i and \mathbf{b}_i are the weights and biases of the DNN, respectively, that are assembled in the vector θ .

Substituting $\hat{u}(\mathbf{z}, \theta)$ into Equation 1 yields the residual DNN:

$$r_f(\mathbf{z}, \theta) = \hat{u}_t(\mathbf{z}, \theta) + \nabla \cdot (-D \nabla \hat{u}(\mathbf{z}, \theta) + \mathbf{v} \hat{u}(\mathbf{z}, \theta)) - s, \quad \mathbf{z} \in \Omega \times (0, T). \quad (5)$$

When \mathbf{v} is nonuniform, it is also approximated with a DNN that is trained separately from $\hat{u}(\mathbf{z}, \theta)$ because of the assumption that $\mathbf{v}(\mathbf{x})$ is independent of $u(\mathbf{x}, t)$, as described in Section 3.4.

Similarly, the residual DNNs corresponding to the BCs and IC are obtained by substituting $\hat{u}(\mathbf{z}, \theta)$ into the boundary conditions for Equation 1 as

$$\begin{aligned} r_{BC_D}(\mathbf{z}, \theta) &= \hat{u}(\mathbf{z}, \theta) - g_D(\mathbf{z}), \quad \mathbf{z} \in \partial\Omega_D \times (0, T) \\ r_{BC_N}(\mathbf{z}, \theta) &= \mathbf{n} \cdot \nabla \hat{u}(\mathbf{z}, \theta) - g_N(\mathbf{z}), \quad \mathbf{z} \in \partial\Omega_N \times (0, T) \end{aligned} \quad (6)$$

and

$$r_{IC}(\mathbf{x}, \theta) = \hat{u}(\mathbf{x}, t = 0) - u_0(\mathbf{x}), \quad \mathbf{x} \in \Omega. \quad (7)$$

The residual networks in Equations 5–7, can be easily evaluated by applying automatic differentiation (Baydin et al., 2015) to the DNN $\hat{u}(z; \theta)$. Next, we define the loss function

$$J(\theta) = w_f J_f(\theta) + w_{BC} J_{BC}(\theta) + w_{IC} J_{IC}(\theta), \quad (8)$$

where

$$\begin{aligned} J_f(\theta) &= \frac{1}{N_f} \sum_{i=1}^{N_f} r_f^2(z_f^i, \theta) \\ J_{BC}(\theta) &= \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} r_{BC}^2(z_{BC}^i, \theta) \\ J_{IC}(\theta) &= \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} r_{IC}^2(x_{IC}^i, \theta), \end{aligned} \quad (9)$$

the residuals of the Dirichlet and Neumann boundary conditions are expressed as $r_{BC}(z, \theta) = \mathcal{B}(\hat{u}(z, \theta)) - g_{BC}(z)$, ($z \in \partial\Omega \times (0, T)$), and \mathcal{B} and g_{BC} are the operator and source term associated with the given boundary condition, respectively. The weights w_f , w_{BC} , and w_{IC} penalize the loss terms associated with the governing PDEs, boundary, and initial conditions, respectively. In Equation 9, $\{z_{BC}^i\}_{i=1}^{N_{BC}}$ are the locations where the boundary conditions are enforced, $\{x_{IC}^i\}_{i=1}^{N_{IC}}$ are the locations where the initial conditions are enforced, and $\{z_f^i\}_{i=1}^{N_f}$ is the set of *residual* points where the PDE's residuals are minimized. Here, N_f is the number of residual points, N_{BC} is the number of boundary points, and N_{IC} is the number of initial condition points. These points form a set of training points, with the number of training points given by $n_h = N_f + N_{BC} + N_{IC}$.

The DNN's parameters θ are found by minimizing the loss function $J(\theta)$:

$$\theta = \min_{\theta^*} J(\theta^*). \quad (10)$$

The resulting DNN $\hat{u}(z, \theta)$ satisfies Equation 1, including the boundary and initial conditions up to the approximation error $|u(x) - \hat{u}(x, \theta)|$ and the error in training $\hat{u}(z, \theta)$. It was shown in (Shin et al., 2020) for some classes of PDEs that if the DNN is large enough to approximate the solution (i.e., the approximation error is negligible), the PINN solution of a PDE (i.e., a solution obtained by minimizing the PDE residuals, including the residuals of the boundary and initial conditions) converges to the strong solution of the PDE as the number of training points n_h approaches infinity.

Remark: For a finite n_h , properly selecting the w_f , w_{BC} , and w_{IC} weights is a critical step to obtain an accurate solution. The loss function (8) enforces PDE and initial and boundary conditions as penalty terms rather than hard constraints. This approach is commonly used in the PINN literature (He et al., 2020; Lu et al., 2019; Mao et al., 2020; Raissi et al., 2019, 2020; Sirignano & Spiliopoulos, 2018; Tartakovsky et al., 2020; Weinan & Yu, 2018) because solving unconstrained optimization problems such as Equation 10 is generally easier than solving a constrained optimization problem.

It was also shown that weights can be adjusted during the minimization process to achieve faster convergence and mitigate gradient pathologies (van der Meer et al., 2020; Wang et al., 2020). However, such weight adjustment incurs additional computational cost, which can be significant in data assimilation and multi-physics problems with multi-term loss functions.

Another approach for imposing boundary conditions is to use a superposition of a particular solution that satisfied the boundary conditions and a general solution (Lagaris et al., 1997, 2000; Yadav et al., 2016). Originally, this approach was limited to simple domain geometries and linear boundary conditions, but it has recently been extended to complex boundary geometries by using the level set method and additional neural networks that are separately trained to satisfy the boundary conditions (Berg & Nyström, 2018; Sun et al., 2020; Zhu et al., 2020). Extending this approach to nonhomogeneous, nonlinear boundary conditions still remains a challenge.

2.3. Training Algorithm

The optimization problem (10) is non-convex. Common methods for minimizing loss functions in DNN training with and without physics constraints include the stochastic-gradient descent Adam method (Kingma & Ba, 2014) and the limited memory BFGS with box constraints (L-BFGS-B) (Byrd et al., 1995). We find that the errors in the PINN method strongly depend on which of these two algorithms is used. Therefore, in this work, we use a two-step optimization algorithm that was found to perform well in the application of the PINN method for parameter estimation (He et al., 2020; Lu et al., 2019). In this two-step algorithm, the Adam method is used first for a prescribed number of iterations, and then the L-BFGS-B method is used until the solution of the optimization problem converges with the prescribed tolerance. At the beginning of the Adams step, the DNN weights are randomly initialized using the Xavier scheme (Glorot & Bengio, 2010).

In the following numerical experiments, DNN training is performed on a PC with eight Intel Xeon E5-1620 3.5 GHz processors. Depending on a problem, the training time ranges from 4 min (approximately 0.5 core-hours) to 3.25 h (26 core-hours).

3. Forward Flow and Advection-Dispersion Equations

In this section, we present four numerical experiments to demonstrate the effectiveness of the PINN approach for solving forward ADEs. First, we compare the PINN solutions of the 1D (Section 3.1) and 2D (Section 3.2) time-dependent ADEs with the analytical solutions that are commonly used to benchmark numerical grid-based methods (Borker et al., 2017; Huang et al., 2008; Hughes & Wells, 2005; Mojtabi & Deville, 2015). In Section 3.3, we investigate the effect of grid orientation on the ADE solution with $Pe \gg 1$, where the crosswind diffusion could lead to numerical instabilities in discretization-based methods.

In Section 3.4, we consider a two-dimensional system of the steady-state Darcy flow equation with a heterogeneous conductivity field and a time-dependent ADE with the anisotropic dispersion coefficient. This problem presents a challenge for numerical approximations of off-diagonal entries in the dispersion tensor, often resulting in nonphysical negative solutions (Herrera et al., 2010; Herrera & Valocchi, 2006; Lipnikov et al., 2007; Tan et al., 2016). This example aims to show that PINN can provide comparable solutions to state-of-the-art numerical solvers such as the Subsurface Transport Over Multiple Phases (STOMP) finite volume code (White et al., 1995) for ADEs with anisotropic dispersion coefficients and nonuniform velocity fields.

The quantitative comparison of PINN solutions with the reference (analytical or grid-based numerical) solutions is given in terms of the relative L_2 error

$$\epsilon = \frac{\| \mathbf{u} - \hat{\mathbf{u}} \|}{\| \mathbf{u} \|} \quad (11)$$

that we compute on a uniform grid over the space-time domain, where the vectors \mathbf{u} and $\hat{\mathbf{u}}$ denote the reference solution and the PINN solution (evaluated at the grid points), respectively. Therefore, the reference solution is used as the validation data set.

We define the DNN size as $n_l \times m_l$, where n_l is the number of hidden layers and m_l is the number of neurons in each hidden layer. For the time-dependent two-dimensional problems considered in the study, we use the $n_t \times n_1 \times n_2$ notation to define N_f , N_{BC} , and N_{IC} in Equation 9. In this notation, n_t denotes the number of time steps along the temporal coordinate (including the initial time), $N_{IC} = n_1 \times n_2$, and $N_{BC} = n_t \times 2 (n_1 + n_2)$. Unless stated otherwise, the number of residual points is $N_f = n_t \times n_1 \times n_2$. The residual data points are randomly distributed over the space-time domain.

3.1. One-Dimensional Time-Dependent ADE

Consider the following one-dimensional ADE

$$\frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} = \kappa \frac{\partial^2 u}{\partial x^2}, \quad -1 < x < 1, \quad t > 0 \quad (12)$$

Table 1
Training Parameters for One-Dimensional Time-Dependent Advection Dispersion Equation

1 st Step: Adam			2 nd Step: L-BFGS-B	
Learning rate	Batch size	Iteration	ftol	gtol
0.001	500	20,000	1.0×10^{-10}	1.0×10^{-8}

with the initial and boundary conditions:

$$\begin{cases} u(x, t = 0) = -\sin(\pi x), & -1 < x < 1 \\ u(x = 0, t) = 0, & u(x = 1, t) = 1, & t > 0, \end{cases} \quad (13)$$

where the velocity is set to $a = 1$. For this problem, we study the accuracy of the PINN method for different $Pe = l \times a/\kappa$, where $l = 2$ is the domain size, against the analytical solution (Mojtabi & Deville, 2015):

$$\begin{aligned} u(x, t) = & 16\pi^2 \kappa^3 a e^{\frac{c}{2\kappa} \left(x - \frac{c}{2} t \right)} \\ & \times \left[\sinh\left(\frac{a}{2\kappa}\right) \sum_{p=0}^{N=\infty} \frac{(-1)^p 2p \sin(p\pi x) e^{-\kappa p^2 \pi^2 t}}{a^4 + 8(a\pi\kappa)^2 (p^2 + 1) + 16(\pi\kappa)^4 (p^2 - 1)^2} \right. \\ & \left. + \cosh\left(\frac{a}{2\kappa}\right) \sum_{p=0}^{N=\infty} \frac{(-1)^p (2p + 1) \cos\left(\frac{2p + 1}{2} \pi x\right) e^{-\kappa \frac{(2p + 1)^2}{4} \pi^2 t}}{a^4 + (a\pi\kappa)^2 (8p^2 + 8p + 10) + (\pi\kappa)^4 (4p^2 + 4p - 3)^2} \right]. \end{aligned} \quad (14)$$

Here, we use $N = 800$ to evaluate the analytical solution.

The PINN solution is obtained on the time domain $T = (0, 2)$. We enforce the initial condition at $N_{IC} = 100$ points and boundary condition at $N_{BC} = 200$ points (times), as well as minimize the PDE residual at $N_f = 200 \times 100$ collocation points. The details of the two-step training scheme are given in Table 1. The PINN solution is obtained in less than 0.5 core-hours.

The snapshots of the PINN and analytical solutions as functions of x at $t = 0.8, 1.0$, and 1.6 for $Pe = 62.8$ ($\kappa = 0.1/\pi$) are given in Figure 1. The PINN solution is obtained with a 4×40 DNN and has a near-perfect agreement with the analytical solution, with the relative error $\epsilon = 8.92 \times 10^{-4}$. Table 2 gives ϵ as a function of the DNN size for this case. It shows that the error can be generally reduced by increasing both the number of hidden layers and the width of the layers. However, it is observed that for the same number of residual points, the 4×50 DNN produces a larger error than the 4×30 DNN. We note that the PINN errors can be reduced by increasing the number of residual points, because, in general, larger DNNs require more residual points for training. For all considered cases, the errors stay below 1%.

Figure 2 shows the PINN and analytical solutions for $Pe = 628$ ($\kappa = 0.01/\pi$) as functions of x at three different times. In Figures 2a–2c, the PINN solution is obtained with a 4×40 DNN, while the solution in Figures 2e–2d is obtained with a smaller 3×40 DNN. Note that the analytical solution 14 develops oscillations for large Pe (including the $Pe = 628$ considered here) and that an approximate analytical solution obtained by a perturbation method in (Mojtabi & Deville, 2015) for the region $x > at - 1$ is used in Figure 2 as a ref-

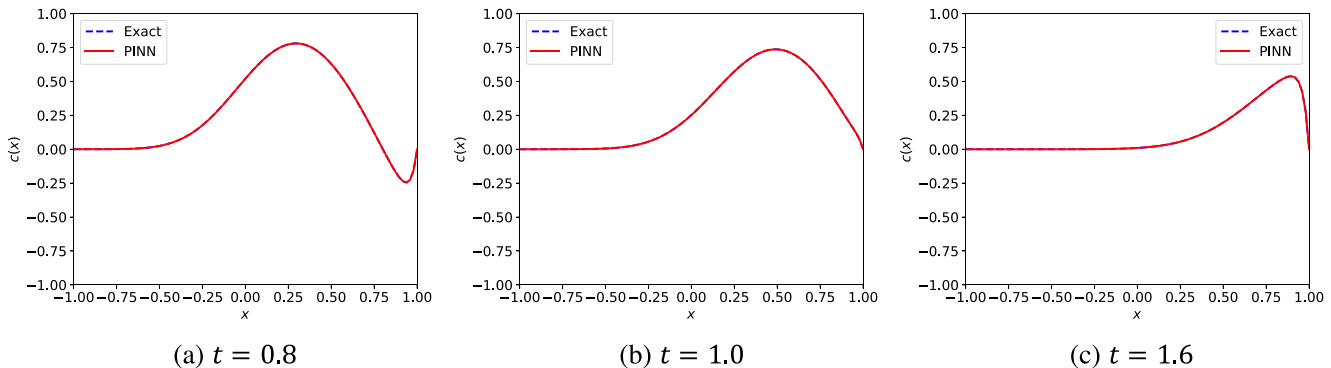


Figure 1. Comparison of the physics-informed neural network (PINN) and analytical solutions of the time-dependent advection-dispersion equation (ADE) (12) as functions of x at times $t = 0.8, 1.0$, and 1.6 . $Pe = 62.8$, and the deep neural network (DNN) size is 4×40 .

Table 2

Relative L_2 Error ϵ for the ADE (12) With $\kappa = 0.1/\pi$ ($Pe = 62.8$)

DNN size	ϵ
2×30	9.186×10^{-3}
2×50	4.152×10^{-3}
3×30	1.676×10^{-3}
3×50	1.218×10^{-3}
4×30	5.854×10^{-4}
4×50	6.381×10^{-4}

Note. The error is calculated on a uniform 100×100 grid over the space-time domain $\Omega \times T$.

erence. We can see that for this Pe , the DNN size plays a more significant role than for $Pe = 62.8$. The solution with 4×40 DNN provides a perfect fit with the analytical solution, while the solution with the smaller DNN has a maximum point error close to 100%. The reason for this is that the solution develops very large gradients near the $x = 1$ boundary, and a sufficiently large DNN is needed to accurately approximate it. We note that in numerical discretization-based solutions of ADEs, an increase in Pe often requires a finer mesh to maintain the same accuracy (Borker et al., 2017; Yadav et al., 2016).

Figure 3 shows the total loss and the individual loss terms versus the number of iterations in the two-step training scheme for $Pe = 62.8$ and 628. In this optimization scheme, the Adam optimizer terminates at the prescribed number of iterations (20,000) and switches to the L-BFGS-B optimizer to achieve the final convergence. The mini-batch stochastic gradient descent-based Adam algorithm produces relatively large oscillations at the late stage of the training process, but it usually results in better DNN generalization properties (He et al., 2020) than those when the L-BFGS-B algorithm is used alone. Using the L-BFGS-B optimizer at the second stage of training reduces oscillations, increases the convergence rate, and reduces the error in the PINN solution. For example, in the PINN solution with $Pe = 62.8$, the ϵ errors are 9.401×10^{-3} and 8.918×10^{-4} at the end of the Adam and L-BFGS-B steps, respectively. The use of L-BFGS-B alone might result in the algorithm being trapped in a local minimum corresponding to a relatively large value of the loss function. The comparison of Figure 3a ($Pe = 68.2$) and 3b ($Pe = 628$) demonstrates that the final loss value increases with increasing Pe , which is mainly due to the larger final value of the residual loss (L_r). Also, the number of iterations required to achieve the same tolerance increases with increasing Pe . For these two examples, it takes nearly 5,000 more iterations during the quasi-Newton L-BFGS-B for $Pe = 628$ than

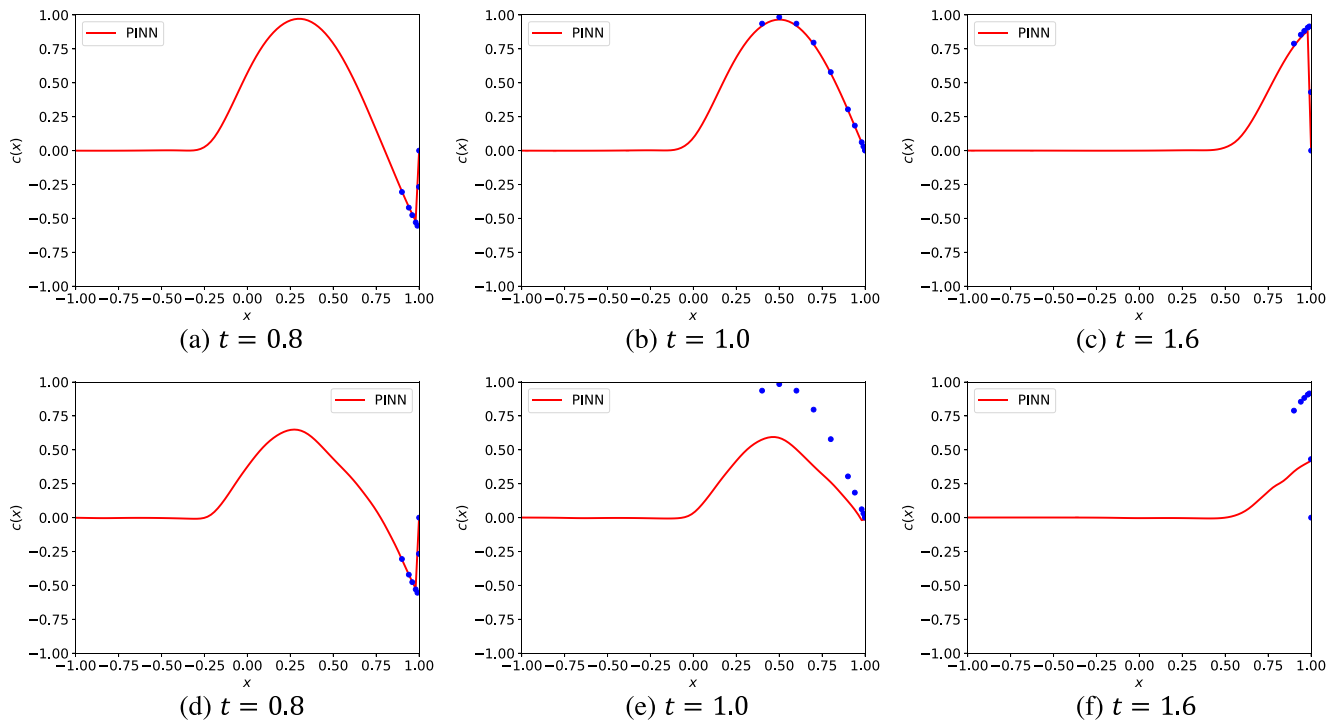


Figure 2. Comparison of the physics-informed neural network (PINN) and analytical solutions of the time-dependent advection-dispersion equation (ADE) (12) as functions of x at times $t = 0.8, 1.0$, and 1.6 for $Pe = 628$. PINN solutions in (a)–(c) are obtained with the deep neural network (DNN) size 4×40 and in (d)–(f) with the DNN size 3×40 . The approximate analytical solution (blue dots) is valid for $x > at - 1$.

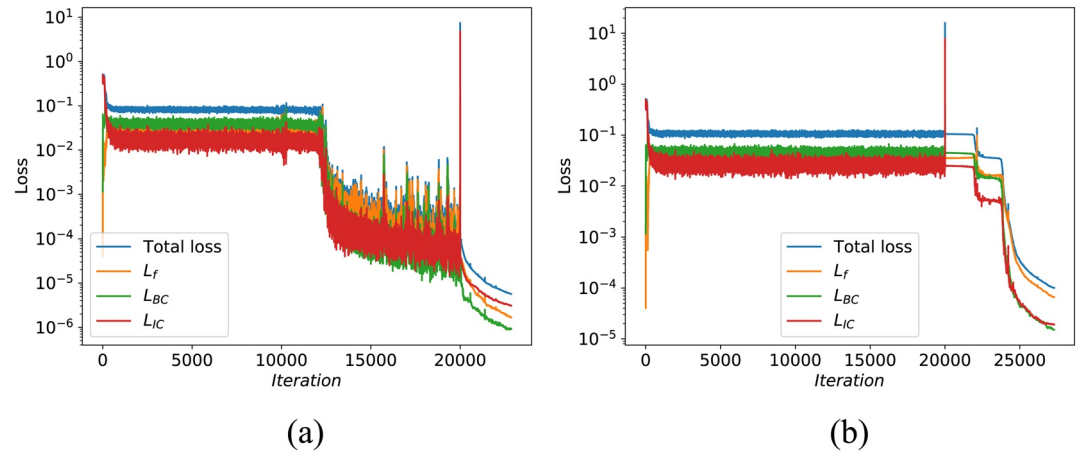


Figure 3. Loss functions for the advection-dispersion equation (ADE) (12) with: (a) $\kappa = 0.1/\pi$ ($Pe = 62.8$) and (b) $\kappa = 0.01/\pi$ ($Pe = 628$). The deep neural network (DNN) size 4×40 is used for the physics-informed neural network (PINN) approach.

for $Pe = 62.8$. In the loss function, we set $w_f = 1$ and $w = w_{BC} = w_{IC} = 1$ or 10 . For this problem, we find that both values of w produce similar PINN solutions.

3.2. Two-Dimensional Time-Dependent ADE: Instantaneous Gaussian Source

Here, we study the performance of the PINN method for the two-dimensional ADE:

$$\begin{cases} u_t + \nabla \cdot (-\kappa \nabla u + au) = 0, & x \in \Omega = (0,1) \times (0,1) \quad t \in (0,T) \\ a = [\cos(\phi) \quad \sin(\phi)]^T \quad \phi = 22.5^\circ \\ u = \frac{1}{4t+1} \exp\left(-\frac{\|x - at\|^2}{\kappa(4t+1)}\right), & x \in \partial\Omega \quad t \in (0,T) \\ u(x,t=0) = \exp\left(-\frac{\|x\|^2}{\kappa}\right), & \text{in } \Omega. \end{cases} \quad (15)$$

This equation allows the analytical solution

$$u = \frac{1}{4t+1} \exp\left(-\frac{\|x - at\|^2}{\kappa(4t+1)}\right). \quad (16)$$

We obtain PINN solutions for $\kappa = 0.02$ and 0.005 , corresponding to $Pe = 50$ and 200 , on the time domain $(0, 0.6)$ with the residual points spanning the time interval $(0, 0.5)$. The time-space grid is set as $100 \times 40 \times 40$. Here, we use the same hyperparameters as in Table 1 for the two-step minimization of the loss function.

The snapshots of the PINN solutions at $t = 0.1, 0.5$, and 0.6 for $Pe = 50$ and 200 and the corresponding errors with respect to the analytical solution are shown in Figures 4 and 5. Here, the ADE residuals and boundary conditions are not enforced for $t > 0.5$, and the PINN solutions at $t = 0.6$ could be thought of as an extrapolation by the neural networks that are trained for $t \leq 0.5$. Figures 4 and 5 show that for both Pe values, the maximum point error is less than 3×10^{-3} for $t \leq 0.5$. For $t = 0.6$, the maximum errors are less than 0.01 .

For this problem, we also investigate the effect of the $w = w_{BC} = w_{IC}$ weights in the loss function (8) on the accuracy of the solution ($w_f = 1$ is set in all simulations). Here, we consider $w = 1, 10$, and 100 . Figures 4 and 5 show that the weights have a significant impact on the accuracy of the PINN solution. For $w = 1$ (see Figures 4a and 5a), the maximum errors are located near the $x_1 = 0$ and $x_2 = 0$ boundaries where the (Dirichlet) boundary conditions are prescribed. These errors are reduced as the weights increase to 10 and 100 ,

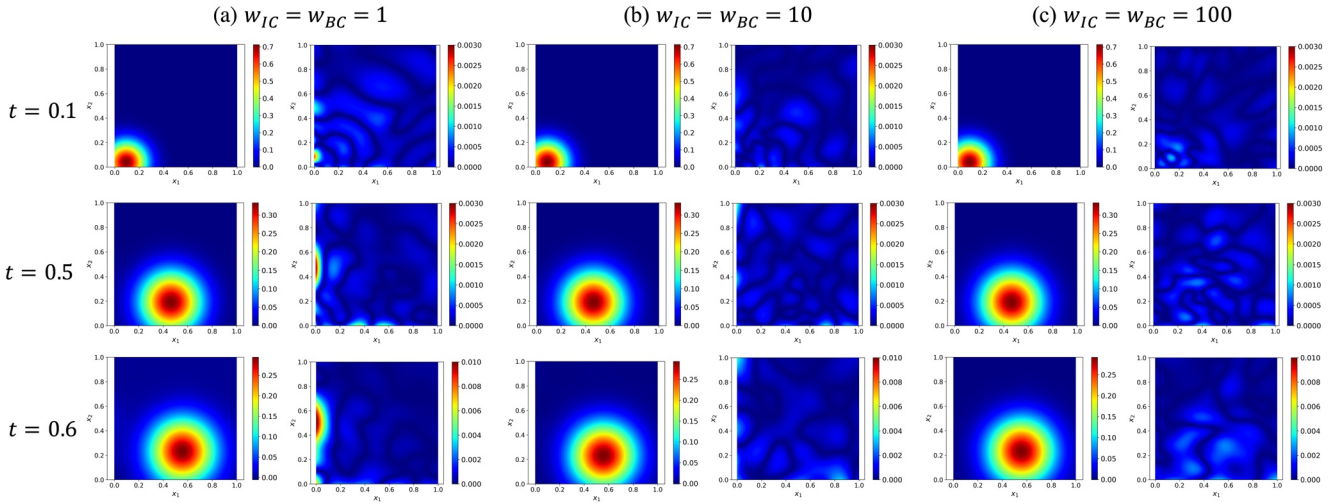


Figure 4. Snapshots of the physics-informed neural network (PINN) solution of the time-dependent advection-dispersion equation (ADE) (15) with $Pe = 50$ (left columns) and the corresponding point errors with respect to the analytical solution (16) (right columns) obtained with: (a) $w = w_{IC} = w_{BC} = 1$, (b) $w = w_{IC} = w_{BC} = 10$, and (c) $w = w_{IC} = w_{BC} = 100$. For $t = 0.5$, the maximum point errors for $w = 1, 10$, and 100 are 2.96×10^{-3} , 1.33×10^{-3} , and 0.97×10^{-3} , respectively.

respectively. The larger weights assigned to J_{BC} and J_{IC} in Equation 8 strongly penalize the initial and boundary conditions relative to the PDE residuals that are shown to reduce the errors in the solution. However, a very large w can also lead to a decrease in accuracy because the effect of the PDE residuals will become negligible. Here, for $Pe = 50$, the relative L_2 errors for $w = 1, 10$, and 100 are $\epsilon = 2.41 \times 10^{-3}$, 1.49×10^{-3} , and 1.51×10^{-3} , respectively. For $Pe = 200$, the relative L_2 errors for $w = 1, 10$, and 100 are $\epsilon = 5.64 \times 10^{-3}$, 2.64×10^{-3} , and 4.11×10^{-3} , respectively. The corresponding maximum point errors for $t = 0.5$ are also reported in Figures 4 and 5. These results show that for the considered problem, $w = 10$ yields optimal solutions for both Pe . Therefore, we use $w = 10$ in the rest of this work.

We note that Equation 15 was solved in Dwivedi and Srinivasan (2020) with the so-called physics informed extreme learning machine method. In this method, only one hidden layer is used and the coefficients associated with the last (output) neural network layer are computed by pseudo-inverse operation. We find that the PINN errors near the boundaries are significantly smaller than those in Dwivedi and Srinivasan (2020). We attribute the better performance of the PINN method to assigning larger weights to the boundary and initial

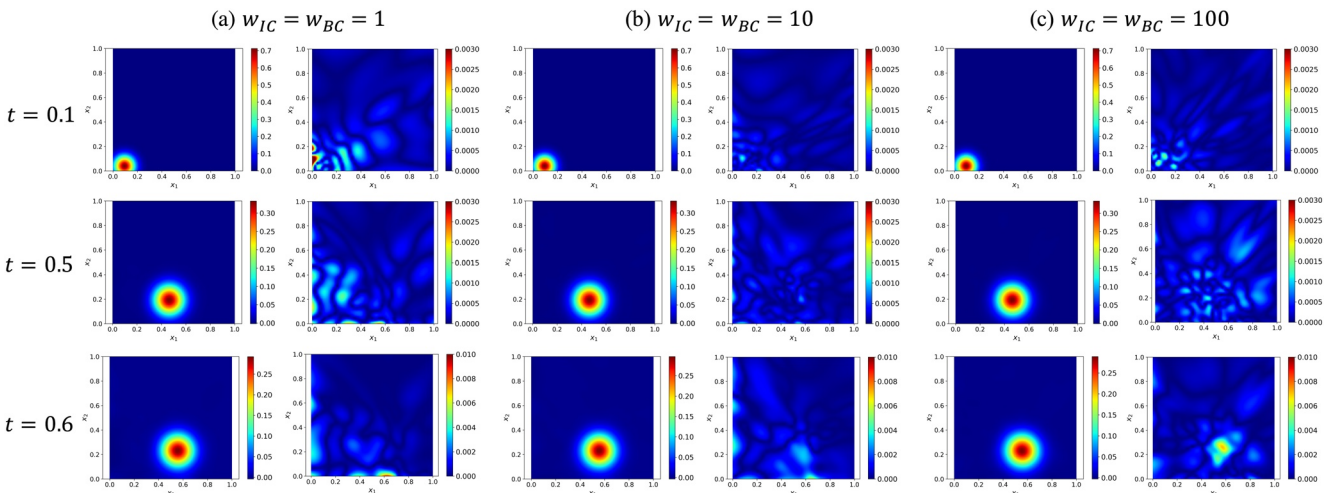


Figure 5. Same as in Figure 4, but with $Pe = 200$. For $t = 0.5$, the maximum point errors for $w = 1, 10$, and 100 are 2.21×10^{-3} , 1.14×10^{-3} , and 1.07×10^{-3} , respectively.

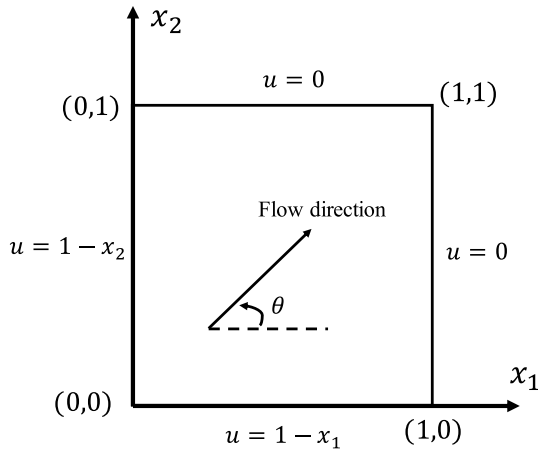


Figure 6. Problem setting for the steady-state advection-dispersion equation (ADE) with Dirichlet boundary conditions. The velocity vector is $\mathbf{v} = (\cos [\pi/4], \sin [\pi/4])^T$.

condition penalty terms in the loss function. The larger error in Dwivedi and Srinivasan (2020) can also be due to the ill-conditioned pseudo-inverse problem.

Finally, we note that the PINN method is able to provide an accurate solution outside of the time interval where the PDE residuals and boundary conditions are enforced in the loss function. Not surprisingly, the accuracy of the PINN extrapolation also depends on the choice of w . For example, the relative L_2 errors at $t = 0.6$ for $Pe = 50$ are $\epsilon = 1.46 \times 10^{-2}$, 0.80×10^{-2} , and 0.91×10^{-2} for the cases of $w = 1, 10$, and 100 , respectively. As with the solution for $t \leq 0.5$, the smallest error is achieved with $w = 10$.

3.3. Two-Dimensional Steady-State ADE With Dirichlet Boundary Conditions. Comparison With the Finite Element Solutions

Here, we compare the PINN solution of a steady-state two-dimensional ADE with the Galerkin FEM (GFEM) and streamline-upwind SUPG solutions that were obtained in (Lin & Atluri, 2000). The direction of the

advection velocity and the boundary conditions are shown in Figure 6. In the FE solutions, the rectangular grid is aligned with the (x_1, x_2) coordinate system and forms a 45° angle with the uniform advection velocity.

The accuracy of grid-based numerical solutions depends on the grid orientation relative to the direction of flow, especially for large Pe problems (Almeida & Silva, 1997; Brooks & Hughes, 1982; Franca et al., 1992; Hillman & Chen, 2016; Hughes et al., 1986; Lin & Atluri, 2000). Specifically, instability and/or excessive diffusion can develop if the proper crosswind diffusion is not well represented by the numerical schemes.

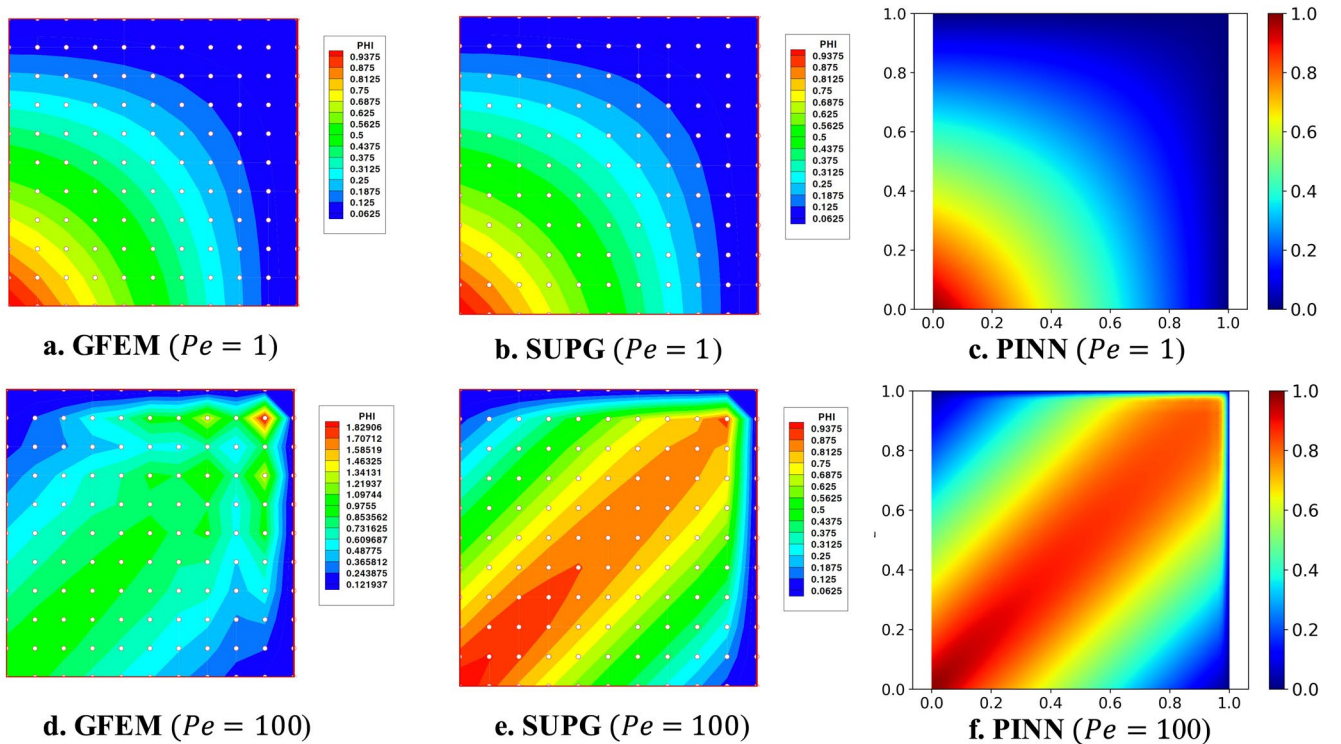


Figure 7. The comparison of the physics-informed neural network (PINN) and finite elements solutions for $Pe = 1$ and 100 . The PINN solution is obtained with $N_f = 41 \times 41$ residual points, the deep neural network (DNN) size 3×30 , and $w_{BC} = 10$. The Galerkin FEM (GFEM) and streamline-upwind Petrov-Galerkin method (SUPG) solutions in subfigures (a), (b), (d), and (e) are reproduced from Lin and Atluri (2000).

Table 3
PINN for the Steady-State ADE With Dirichlet Boundary Conditions:
Parameters in the Training Algorithm

1 st Step: Adam			2 nd Step: L-BFGS-B	
Learning rate	Batch size	Iteration	<i>ftol</i>	<i>gtol</i>
0.001	100	10,000	1.0×10^{-10}	1.0×10^{-8}

Abbreviations: ADE, advection-dispersion equation; PINN, physics-informed neural network.

Figures 7a and 7b demonstrate that the GFEM and SUPG methods give similar solutions for $Pe = 1$. On the other hand, for $Pe = 100$, the GFEM solution develops instabilities (Figure 7d), while the SUPG solution remains stable (Figure 7e) because of the use of the upwind scheme (Brooks & Hughes, 1982; Hughes et al., 1986).

Figures 7c and 7f depict the PINN solutions for $Pe = 1$ and 100, respectively, that are in a close agreement with the SUPG solutions. In the PINN solutions, the collocation points are uniformly spaced on the 41×41 mesh, the boundary conditions are enforced at 41 points at each boundary, and $w = w_{BC} = 10$. The parameters for the training algorithms are listed in Table 3. While both the PINN and SUPG methods provide stable

solutions for high and low Pe , the SUPG method requires the calibration of the stability parameter for a given mesh size and the flow direction relative to the mesh orientation. The PINN method does not use a mesh to discretize spatial derivatives. Therefore, the distribution of the collocation points relative to the direction of the flow does not affect the accuracy of the PINN solution. We also observe that the solutions do not produce oscillations at the dispersion front (over- and/or undershoots) that are often present in numerical grid-based solutions of ADEs. We attribute this phenomenon to computing spatial derivatives analytically rather than using a numerical discretization.

It is also worth noting that it is easy to locally introduce additional residual points in the regions with high concentration gradients to further improve the accuracy of the PINN method for high Pe problems. Such an approach was used in (Mao et al., 2020) for fluid-flow simulations and has a similar effect to that of adaptive mesh refinement (h-refinement) in discretization-based methods without the challenge of discretizing derivatives on a multi-resolution mesh.

3.4. Two-Dimensional Time-Dependent ADE With a Non-Uniform Velocity Field

So far, we have considered ADEs with a uniform velocity field and isotropic dispersion coefficient $\mathbf{D} = \kappa \mathbf{I}$, where \mathbf{I} is the identity tensor. In this section, we solve the coupled ADE and Darcy flow equations, where the velocity field is not uniform and is given by the solution of the Darcy equation, and \mathbf{D} is an anisotropic tensor. The ADE takes the following form:

$$\begin{cases} u_t + \nabla \cdot [\mathbf{v}(\mathbf{x})u(\mathbf{x})] = \nabla \cdot [\mathbf{D}\nabla u(\mathbf{x})], & \mathbf{x} \in \Omega, \quad t \in (0, T) \\ u(\mathbf{x}, t) = u_D(x_2), & x_1 = 0 \\ \partial u(\mathbf{x}, t) / \partial x_1 = 0, & x_1 = L_1 \\ \partial u(\mathbf{x}, t) / \partial x_2 = 0, & \text{on } x_2 = 0 \text{ and } x_2 = L_2 \\ u(\mathbf{x}, t = 0) = 0, & \mathbf{x} \in \Omega \end{cases} \quad (17)$$

where \mathbf{D} is defined in Equation 2 and \mathbf{v} is the average pore velocity

$$\mathbf{v}(\mathbf{x}) = -\frac{K(\mathbf{x})}{\phi} \nabla h(\mathbf{x}), \quad (18)$$

and the hydraulic head h is given by the steady-state Darcy equation:

$$\begin{cases} \nabla \cdot [K(\mathbf{x})\nabla h(\mathbf{x})] = 0, & \mathbf{x} \in \Omega \\ h(\mathbf{x}) = H_2, & x_1 = L_1 \\ -K(\mathbf{x})\partial h(\mathbf{x}) / \partial x_1 = q, & x_1 = 0 \\ -K(\mathbf{x})\partial h(\mathbf{x}) / \partial x_2 = 0, & x_2 = 0 \text{ or } x_2 = L_2. \end{cases} \quad (19)$$

Here, ϕ is the porosity and $K(\mathbf{x})$ is the known hydraulic conductivity that is, in general, defined at a set of points. Usually, $K(\mathbf{x})$ is estimated on a mesh from a calibration study.

The parameters in the above equations are set to: $L_1 = 1$ m, $L_2 = 0.5$ m, $H_2 = 0$ m, $q = 1$ m/hr, $u_D(x_2) = c \exp(-\frac{(x_2 - L_2/2)^2}{\epsilon^2})$, $c = 1$ Kg/m³, $\epsilon = 0.25$ m, $\phi = 0.317$, $D_w = 0.09$ m²/hr, $\tau = \phi^{1/3} = 0.681$,

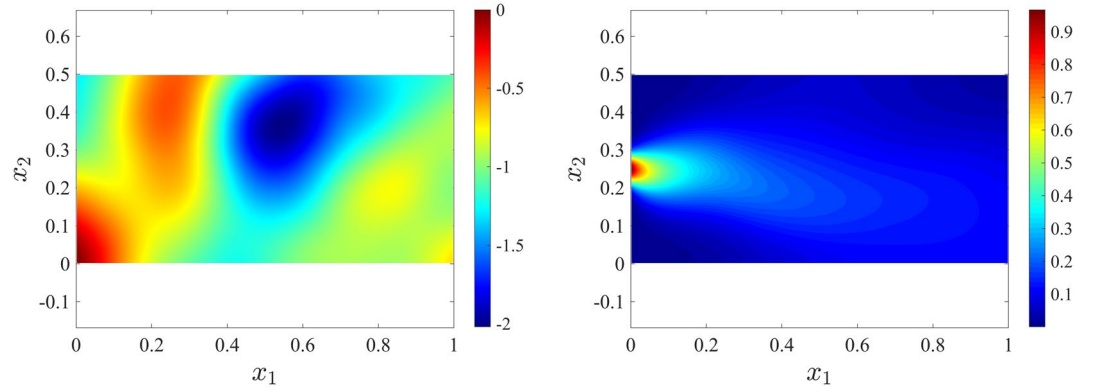


Figure 8. Reference fields: (a) log $K(\mathbf{x})$ and (b) concentration $u(\mathbf{x})$ at $t = 20$ min.

$\alpha_L = 0.01$ m, and $\alpha_T = 0.001$ m. In this example, we consider a heterogeneous conductivity field (see Figure 8a) that is generated as a realization of the lognormal process with the correlation length $\lambda = 0.5$. The reference fields $u(\mathbf{x})$ and $h(\mathbf{x})$ are obtained by solving the ADE and Darcy equations numerically using the STOMP code on the 256×128 spatial grid. A snapshot of the concentration reference field u at $t = 20$ min is shown in Figure 8b.

In the PINN approach for Equation 17–19, we start by approximating the $K(\mathbf{x})$ field with a DNN $\hat{K}(\mathbf{x}, \psi)$ that is trained using a subset of the K values. The remaining values of K are used to test $\hat{K}(\mathbf{x}, \psi)$ and verify that there is no over-fitting. Next, we use PINN to solve Equation 19 by approximating the hydraulic head as $h(\mathbf{x}) \approx \hat{h}(\mathbf{x}, \gamma)$ and computing γ from the minimization problem

$$\gamma = \min_{\gamma^*} \left[\frac{1}{N_f} \sum_{i=1}^{N_f} r_f(\mathbf{x}_f^i, \gamma^*)^2 + J_1(\gamma^*) + J_2(\gamma^*) + J_3(\gamma^*) + J_4(\gamma^*) \right], \quad (20)$$

where

$$r_f(\mathbf{x}, \gamma) = \nabla \cdot [\hat{K}(\mathbf{x}) \nabla \hat{h}(\mathbf{x}, \gamma)] \quad (21)$$

are the residuals evaluated at N_f residual points \mathbf{x}_f^i ,

$$J_1(\gamma) = \frac{1}{N_1} \sum_{i=1}^{N_1} (\hat{h}(\mathbf{x}_{BC1}^i, \gamma) - H_2)^2 \quad (22)$$

is the loss of the Dirichlet BC evaluated at N_1 points at $x_1 = L_1$,

$$J_2(\gamma) = \frac{1}{N_2} \sum_{i=1}^{N_2} \left(K(\mathbf{x}_{BC2}^i, \gamma) \frac{\partial \hat{h}(\mathbf{x}, \gamma)}{\partial x_1} \Big|_{\mathbf{x}=\mathbf{x}_{BC2}^i} + q \right)^2 \quad (23)$$

is the loss of the Neumann boundary condition at the boundary $x_1 = 0$ evaluated at N_2 points,

$$J_3(\gamma) = \frac{1}{N_3} \sum_{i=1}^{N_3} \left(K(\mathbf{x}_{BC3}^i, \gamma) \frac{\partial \hat{h}(\mathbf{x}, \gamma)}{\partial x_2} \Big|_{\mathbf{x}=\mathbf{x}_{BC3}^i} \right)^2 \quad (24)$$

is the loss of the Neumann boundary condition at the boundary $x_2 = 0$ evaluated at N_3 points, and

$$J_4(\gamma) = \frac{1}{N_4} \sum_{i=1}^{N_4} \left(K(\mathbf{x}_{BC4}^i, \gamma) \frac{\partial \hat{h}(\mathbf{x}, \gamma)}{\partial x_2} \Big|_{\mathbf{x}=\mathbf{x}_{BC4}^i} \right)^2 \quad (25)$$

Table 4

Training Parameters for the Two-Dimensional Time-Dependent ADE Problem With a Non-Uniform Velocity Field

	1 st Step: Adam			2 nd Step: L-BFGS-B	
	Learning rate	Batch size	Iteration	<i>ftol</i>	<i>gtol</i>
Darcy Equation 19	0.001	Full batch	30,000	1.0×10^{-10}	1.0×10^{-5}
AD Equation 17	0.0002	1000	200,000	1.0×10^{-10}	1.0×10^{-5}

Abbreviation: ADE, advection-dispersion equation.

is the loss of the Neumann boundary condition at the boundary at $x_2 = L_2$ evaluated at N_4 points. After the DNN weights γ are determined, the velocity is computed as

$$\hat{\mathbf{v}}(\mathbf{x}, \gamma) = -\frac{\hat{K}(\mathbf{x}, \gamma)}{\phi} \nabla \hat{h}(\mathbf{x}, \gamma). \quad (26)$$

In the above, all spatial derivatives of DNNs are computed using automatic differentiation (Baydin et al., 2015). The ADE is solved as described in Section 2 with the residual (5) written in terms of $\hat{\mathbf{v}}(\mathbf{x}, \gamma)$:

$$r_f(\mathbf{z}, \theta) = \hat{u}_t(\mathbf{z}, \theta) + \nabla \cdot (-\mathbf{D} \nabla \hat{u}(\mathbf{z}, \theta) + \hat{\mathbf{v}}(\mathbf{x}, \gamma) \hat{u}(\mathbf{z}, \theta)), \quad \mathbf{z} \in \Omega \times (0, T). \quad (27)$$

The parameters γ in the $\mathbf{v}(\mathbf{x}, \gamma)$ DNN are frozen when training the ADE solution $\hat{u}(\mathbf{z}, \theta)$. The residuals $r_f(\mathbf{z}, \theta)$ are minimized at $N_f = 40,000$ residual points. Parameters in the two-step optimization for solving Equations 17 and 19 are summarized in Table 4. The DNNs training takes approximately 14 core-hours. Considering the high nonlinearities in concentration and large numbers of residual points, we use a smaller learning rate (0.0002) in Adams algorithms for ADE than the learning rate of 0.001 that we used in the previous examples.

Figure 9 shows the PINN solution $\hat{h}(\mathbf{x}, \gamma)$ for the hydraulic head and the point errors in this solution. We see a good agreement with the reference solution, with maximum point errors of less than 0.09 and a relative L_2 error of $\epsilon_h = 1.63 \times 10^{-3}$. The velocity field given by the $\mathbf{v}(\mathbf{x}, \gamma)$ DNN (26) is shown in Figure 10. We solve the ADE (17) for $t \in [0, 20]$ min that is enough for the solution $u(\mathbf{x}, t)$ to reach a steady state. In the PINN method, we enforce the initial and boundary conditions on the time-space grid $40 \times 256 \times 128$, which is consistent with the time-space discretization in the STOMP simulation. The weight w is set to 10.

The PINN solution for the concentration field $u(\mathbf{x}, t)$ and the point errors (the difference between the PINN and STOMP solutions) at $t = 2$ min, $t = 6$ min, and $t = 20$ min are shown in Figure 11, with the relative L_2 errors 2.20×10^{-2} , 1.36×10^{-2} , and 1.21×10^{-2} , respectively, and the maximum point error below 0.03. From

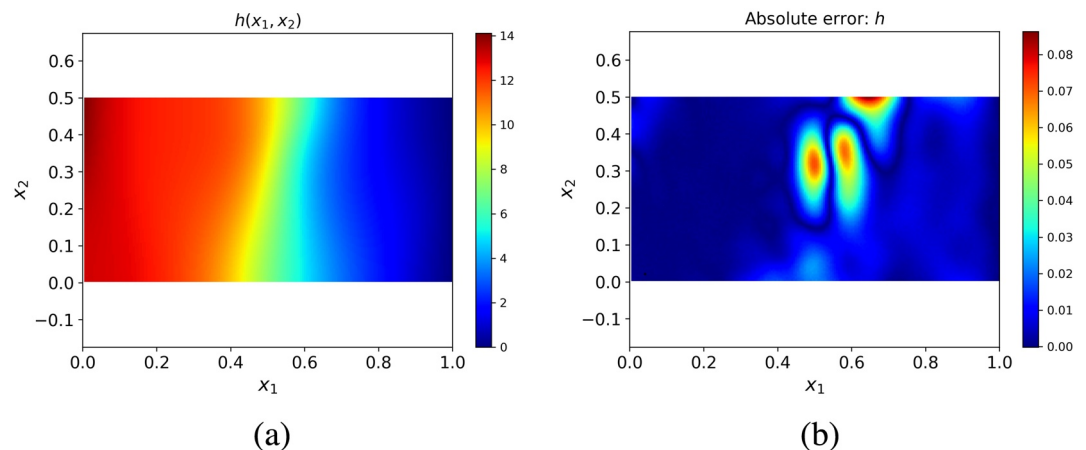


Figure 9. (a) The hydraulic head h obtained as a physics-informed neural network (PINN) solution of the Darcy Equation 19 and (b) the corresponding absolute point errors with respect to the reference Subsurface Transport Over Multiple Phases (STOMP) solution.

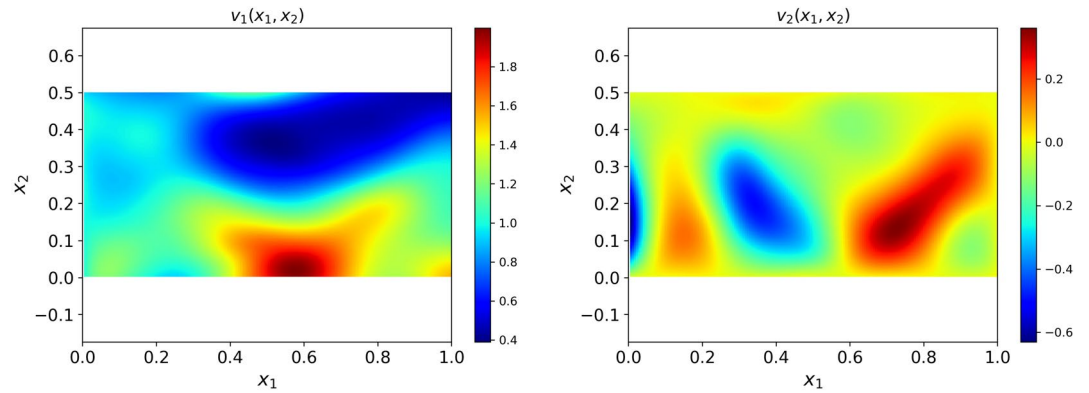


Figure 10. The x_1 and x_2 components of the velocity vector field \mathbf{v} obtained by the physics-informed neural network (PINN) method from Equation 26.

the error plots in Figures 11d–11f, we observe that the maximum errors in the PINN solution develop at the front of the plume where the concentration gradient is the largest. In addition, the point errors reduce with time as the plume becomes more diffused.

Finally, we note that approximating a more complex conductivity field (i.e., a field with a smaller correlation length) requires a larger DNN size (He et al., 2020). Therefore, the cost of solving the coupled Darcy and advection dispersion equations would also increase with the increasing complexity of the conductivity field. This is equivalent to an increase in the cost of traditional numerical methods that require a finer mesh to discretize a more complex conductivity field.

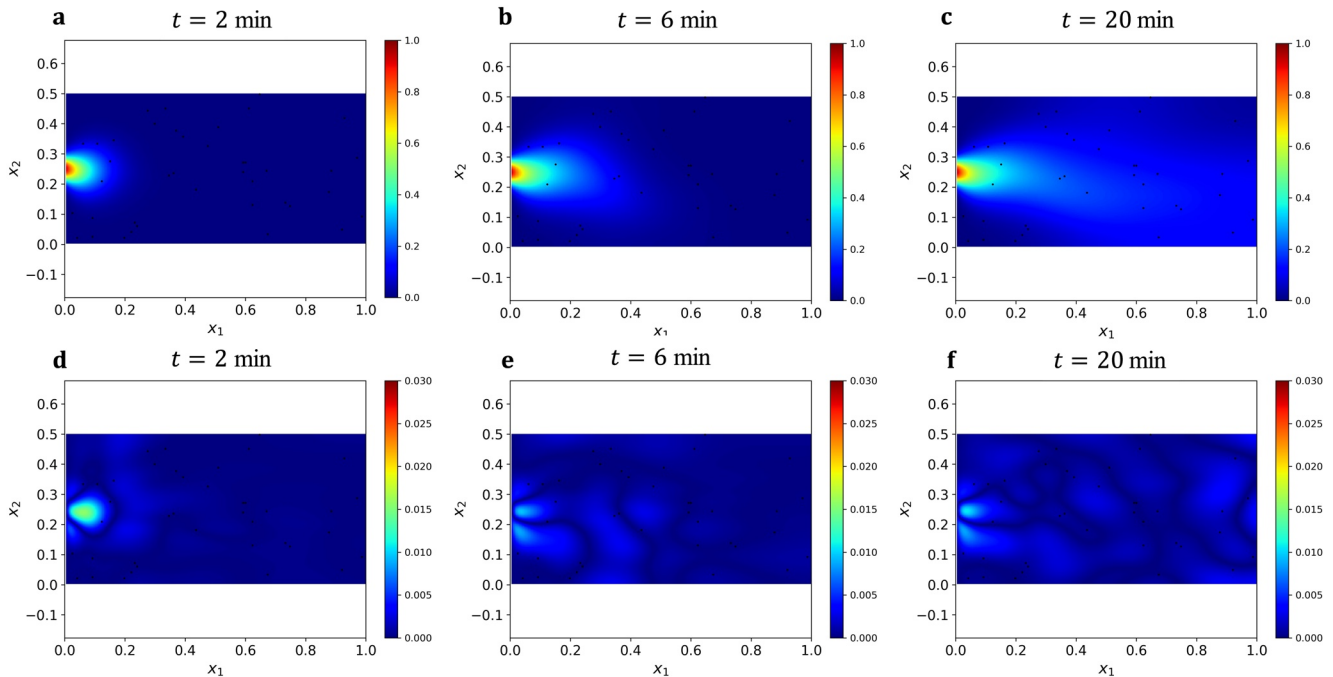


Figure 11. The concentration u obtained as a physics-informed neural network (PINN) solution of advection-dispersion equation (ADE) (17) at (a) $t = 2$ min, (b) $t = 6$ min, and (c) $t = 20$ min. The absolute point errors with respect to the Subsurface Transport Over Multiple Phases (STOMP) solution are shown in subfigures (d)–(f). The deep neural network (DNN) size is 5×60 .

Table 5
Training Parameters for the Backward ADE With Uniform Velocity Field

1 st Step: Adam			2 nd Step: L-BFGS-B	
Learning rate	Batch size	Iteration	<i>ftol</i>	<i>gtol</i>
0.001	500	30,000	1.0×10^{-10}	1.0×10^{-8}

4. Backward ADEs

4.1. Backward ADE With Uniform Velocity Field

In this section, we consider the backward form of the ADE (15), also known as a final boundary value problem, where the terminal condition is given at $t = T$:

$$\begin{cases} u_t + \nabla \cdot (-\kappa \nabla u + \mathbf{a}u) = q, & \mathbf{x} \in \Omega = (0,1) \times (0,1) \quad t \in (0,T) \\ \mathbf{a} = [\cos(\phi), \sin(\phi)]^T \quad \phi = 22.5^\circ \\ u = \frac{1}{4t+1} \exp\left(-\frac{\|\mathbf{x} - \mathbf{a}t\|^2}{\kappa(4t+1)}\right), & \mathbf{x} \in \partial\Omega \quad t \in (0,T) \\ u(\mathbf{x}, t = T) = \frac{1}{4T+1} \exp\left(-\frac{\|\mathbf{x} - \mathbf{a}T\|^2}{\kappa(4T+1)}\right), & \mathbf{x} \in \Omega \end{cases} \quad (28)$$

The solution $u(\mathbf{x}, t)$ is sought for $0 \leq t < T$, where $T = 0.5$. Note that backward diffusion problems are normally ill-posed and that standard numerical methods require a form of regularization for solving such equations (Xiong et al., 2006).

In this example, we set the DNN size to 3×30 and the weight value to $w = 10$. The training hyperparameters are listed in Table 5. It takes approximately 1 core-hour to complete the training of the PINN model. Table 6 shows the relative L_2 errors ϵ in the PINN solution of the backward problem 28 at different times ($t = 0.0, 0.1, 0.3, 0.5$) and Péclet numbers ($Pe = 1, 10, 200$). As expected, ϵ increases as time decreases toward $t = 0$ but remains smaller than 2×10^{-2} in all considered examples. The PINN solutions $\hat{u}(\mathbf{x}, t)$ at $t = 0$ and the corresponding error distributions for different Pe are given in Figure 12. The approximation errors slightly increase with Pe . It appears that these errors are related to the gradients in the solutions and not to the loss of information (that leads to ill-posedness of the backward ADE problem). This is important because the former source of errors can be reduced by increasing the DNN size and the number of residual points.

4.2. Backward ADE With Non-Uniform Velocity Field

Here, we consider a backward problem corresponding to the ADE in Section 3.4. We assume that $u(\mathbf{x}, t)$ at $t = T = 10$ min (see Figure 13f) is known and aim to find $u(\mathbf{x}, t)$ for $t < 10$ min. The same hyperparameters as in the forward ADE problem are used (see Table 4), except the number of Adam iterations is set to 400,000. We use a $21 \times 256 \times 128$ time-space grid and 20,000 residual points. The training time for this simulation is approximately 26 core-hours.

The distributions of the absolute errors between the PINN solution and the reference concentration field $u(\mathbf{x}, t)$ (obtained as the STOMP solution of the forward ADE in Section 3.4) at $t = 0, 1, 2, 4, 8$, and 10 min are shown in Figure 14, and the associated maximum point errors are summarized in Table 7. The errors increase as the backward solution evolves from $t = 10$ min to $t = 0$ min. For $t = 0$ min, a large maximum point error around 0.963 exists near the boundary $x = 0$ where the time-independent concentration u_D is prescribed. This is due to the discontinuity in the solution u at the boundary $x = 0$ and the initial zero concentration in the reference solution. However, for $t > 0$, the maximum point errors in the backward solution do not exceed 5×10^{-2} , with the largest errors observed around the plume where large gradients develop.

Table 6
The Relative L_2 Errors of the PINN Solutions of the Two-Dimensional Backward ADE (28) With Respect to the Reference Solutions for Different Péclet Numbers

Time	$Pe = 1$	$Pe = 10$	$Pe = 200$
$t = 0.0$	6.205×10^{-3}	1.214×10^{-2}	2.077×10^{-2}
$t = 0.1$	8.834×10^{-4}	2.967×10^{-3}	7.344×10^{-3}
$t = 0.3$	1.672×10^{-4}	7.694×10^{-4}	4.449×10^{-3}
$t = 0.5$	1.173×10^{-4}	3.318×10^{-4}	3.218×10^{-3}

Note. The DNN size 3×30 and the weight $w = 10$ are used in the PINN solutions.

5. Data Assimilation

An important feature of the PINN method for ADEs and other PDEs is that the measurements of state variables can be incorporated in the solution without any modifications of the algorithms. When solutions are unstable (small perturbations in the solution can grow infinitely in time) or nonunique (as is the case with some backward ADEs), the state measurements can stabilize and regularize such solutions.

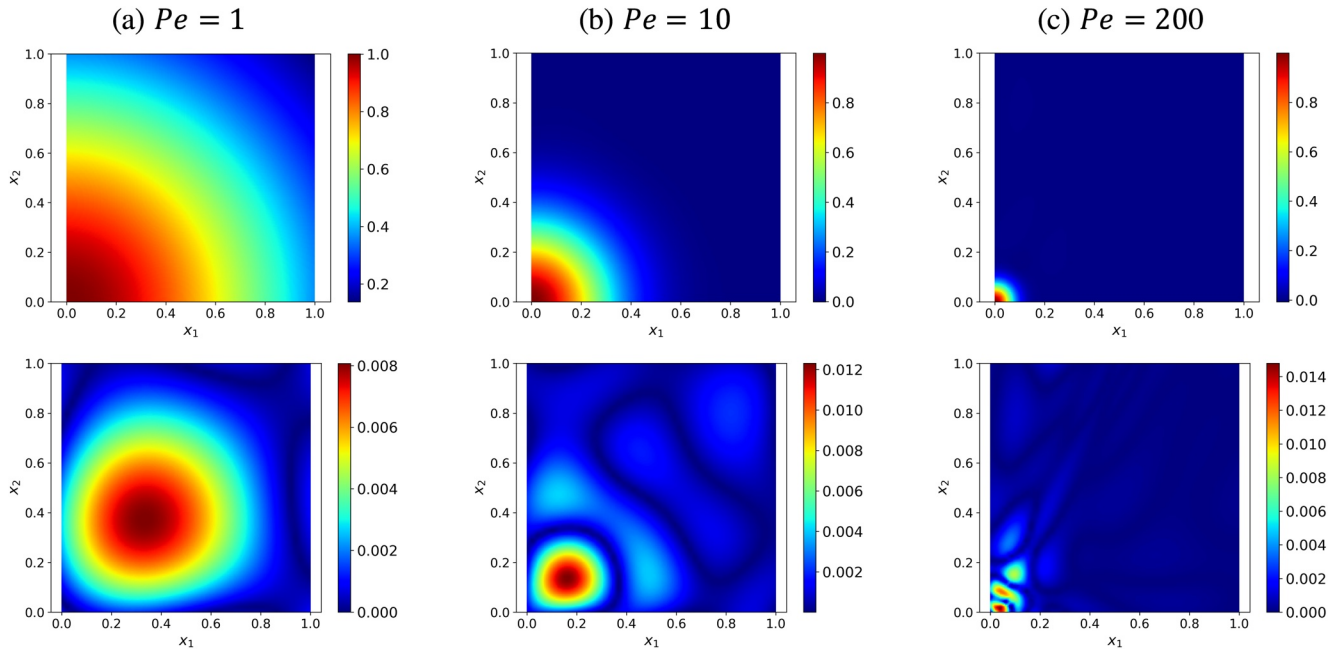


Figure 12. Upper row: the physics-informed neural network (PINN) solution of the backward advection-dispersion equation (ADE) (28) at $t = 0$ with (a) $Pe = 1$, (b) $Pe = 10$, and (c) $Pe = 200$. Lower row: the absolute point errors with respect to the reference solutions. The maximum point errors for these three cases are 8.07×10^{-3} , 1.23×10^{-2} , and 1.40×10^{-2} , respectively.

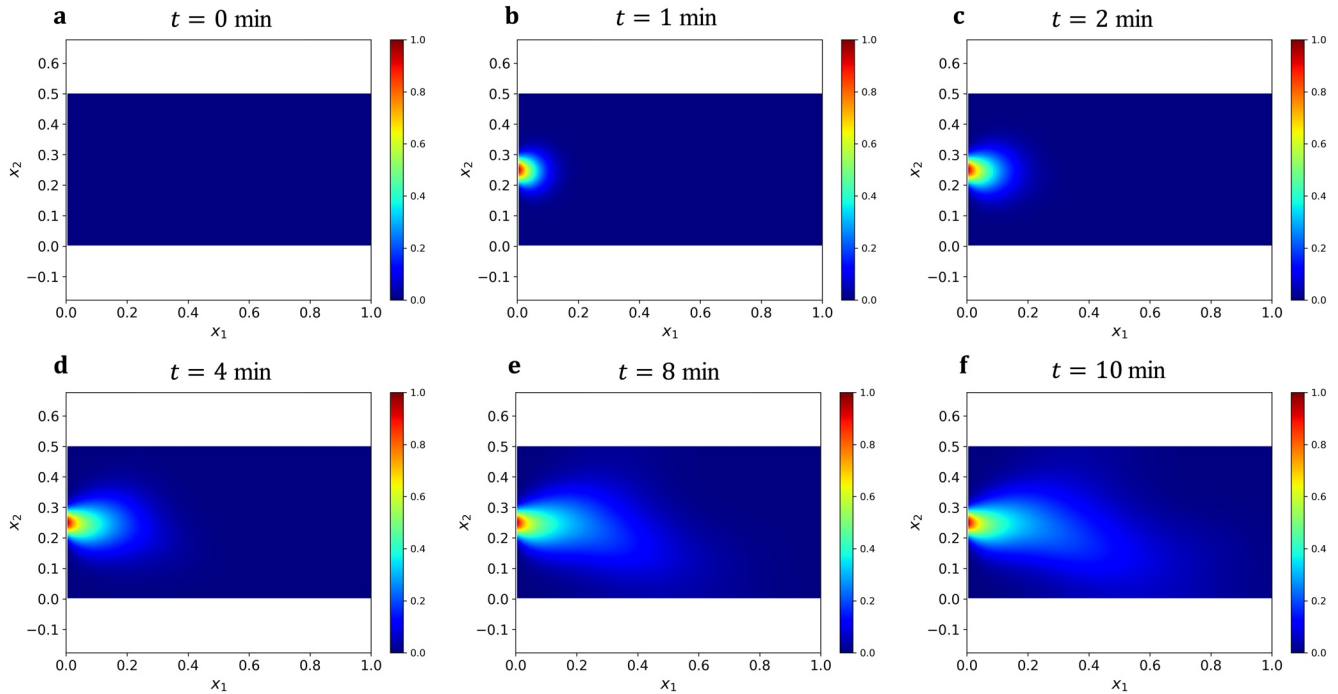


Figure 13. The reference concentration u obtained as a Subsurface Transport Over Multiple Phases (STOMP) solution of the ADE (17) at different times.

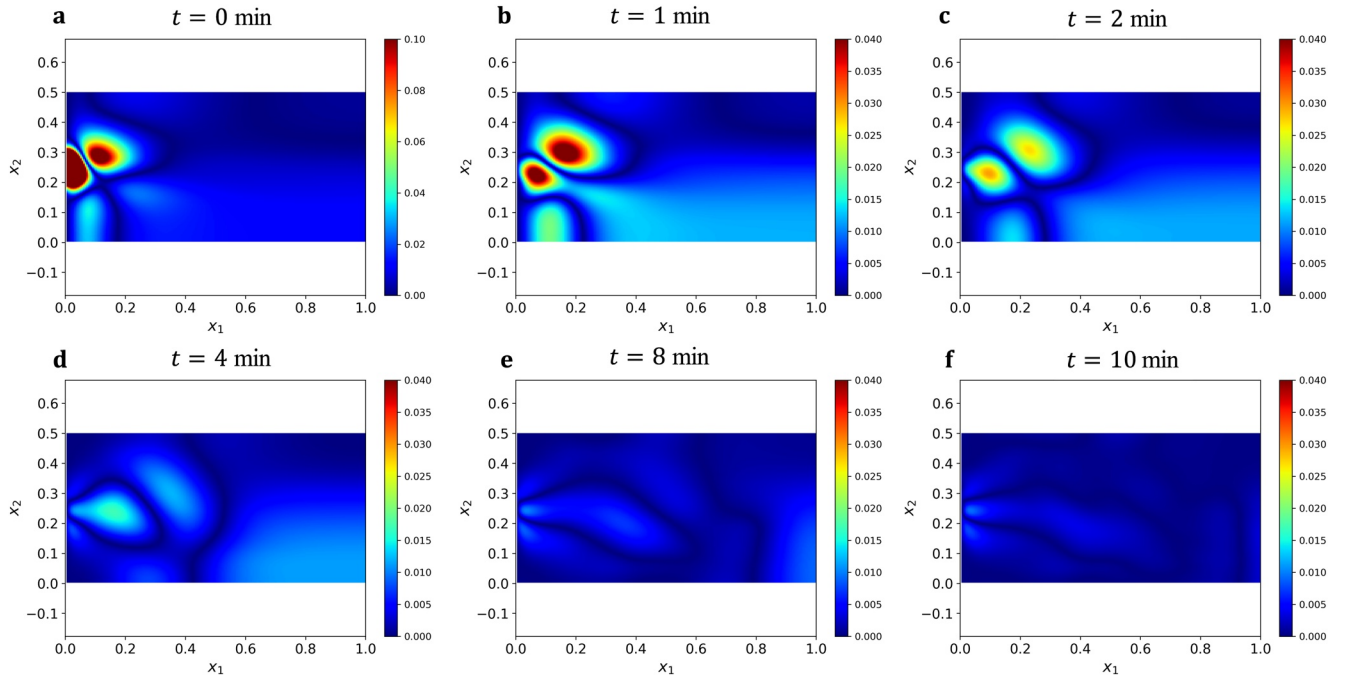


Figure 14. The absolute errors in the physics-informed neural network (PINN) solution $\hat{u}(\mathbf{x}, t, \theta)$ with respect to the reference field $u(\mathbf{x}, t)$ at different times. The deep neural network (DNN) size is 5×60 .

The measurements of $u(\mathbf{x}, t)$, $\{u^*(\mathbf{z}_i)\}_{i=1}^{N_m}$ ($\mathbf{z}_i = (\mathbf{x}, t)_i$ is the point in time and space where the measurement is collected) can be incorporated in the PINN solution of the backward ADEs by adding the term $J_m(\theta) = \frac{1}{N_m} \sum_{i=1}^{N_m} (u^*(\mathbf{z}_i) - \hat{u}(\mathbf{z}_i, \theta))^2$ into the loss function (8), yielding

$$J(\theta) = w_f J_f(\theta) + w_{BC} J_{BC}(\theta) + w_{TC} J_{TC}(\theta) + w_m J_m(\theta), \quad (29)$$

where w_m is the weight corresponding to the J_m loss. Note that the loss function (29) for the backward equation has the term $w_{TC} J_{TC}(\theta) = w_{TC} \frac{1}{N_{TC}} \sum_{i=1}^{N_{TC}} (u_{TC}(\mathbf{x}_i) - \hat{u}(\mathbf{x}_i, t = T, \theta))^2$, which is a mean square difference with respect to the terminal condition instead of the $J_{IC}(\theta)$ term in the loss function (8) for the forward ADE. In this example, we set $w = w_{IC} = w_{BC} = 10$ and $w_m = 10$.

We applied the PINN method to solve the backward ADE in Section 4.2 under the assumption that $N_m = 40$ measurements of u are available at 40 spatial locations randomly distributed in Ω and uniformly distributed in time over 21 time intervals. We use the same training parameters as for the backward ADE problem in Section 4.2 that results in approximately the same training time of 26 core-hours. This indicates that adding data does not increase the computational cost of the PINN method.

Figure 15 shows the errors in the resulting PINN solution $\hat{u}(\mathbf{x}, t, \theta)$ with respect to the reference $u(\mathbf{x}, t)$ field, with the maximum point errors provided in Table 8. The comparison of these errors with errors in Figure 14 and Table 7 shows that adding 40 measurements reduces errors in the estimated u by more than 50%.

Table 7

The Maximum Point Errors of the PINN Solution With Respect to the Reference STOMP Solution for the Backward ADE With Non-Uniform Velocity Field

$t = 0 \text{ min}$	$t = 1 \text{ min}$	$t = 2 \text{ min}$	$t = 4 \text{ min}$	$t = 8 \text{ min}$	$t = 10 \text{ min}$
9.63×10^{-1}	4.73×10^{-2}	2.97×10^{-2}	1.70×10^{-2}	1.00×10^{-2}	0.93×10^{-2}
Abbreviations: ADE, advection-dispersion equation; PINN, physics-informed neural network; STOMP, subsurface transport over multiple phases.					

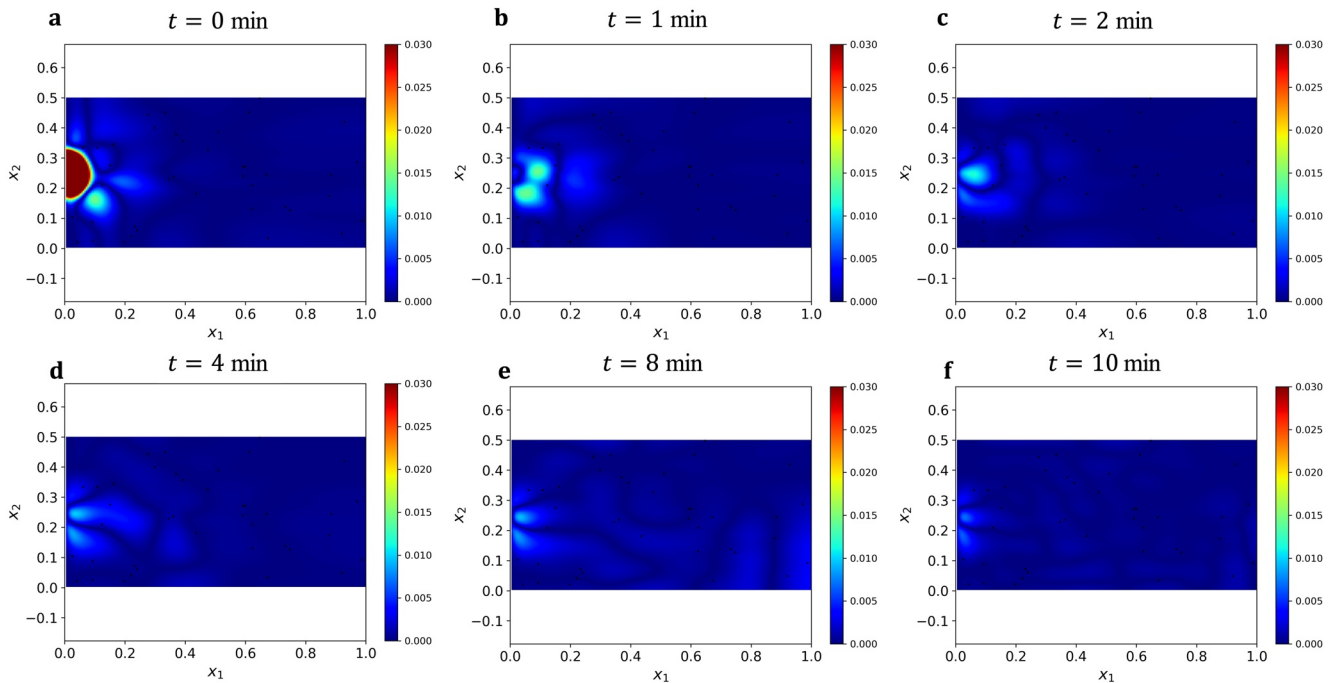


Figure 15. Absolute error in the physics-informed neural network (PINN) solution of the backward advection-dispersion equation (ADE) conditioned on 40 u measurements. The absolute errors are computed with respect to the reference $u(\mathbf{x}, t)$ field. The deep neural network (DNN) size is 5×60 .

Table 8

The Maximum Point Errors of the PINN Solution With Respect to the Reference STOMP Solution for the Backward ADE With Data Assimilation

$t = 0 \text{ min}$	$t = 1 \text{ min}$	$t = 2 \text{ min}$	$t = 4 \text{ min}$	$t = 8 \text{ min}$	$t = 10 \text{ min}$
9.62×10^{-1}	1.45×10^{-2}	1.16×10^{-2}	0.99×10^{-2}	0.93×10^{-2}	0.70×10^{-2}

Abbreviations: ADE, advection-dispersion equation; PINN, physics-informed neural network; STOMP, subsurface transport over multiple phases.

6. Conclusion

We present the PINN method for solving the coupled Darcy equation and ADE and test it for one- and two-dimensional forward and backward ADEs for a range of Pe . We use a weighted sum of residual terms in the loss function and show that the residuals of the initial and boundary conditions should have larger weights than the PDE residuals to obtain accurate solutions. For coupled Darcy flow and advection-dispersion equations with space-dependent hydraulic conductivity and velocity fields, we find that the PINN solutions for the hydraulic head and concentration agree well with numerical solutions obtained with the finite volumes method. We also show that for advection-dominated transport, the PINN method outperforms the GFEM and is comparable in accuracy with the SUPG method, which requires the calibration of a stability parameter specific to a given mesh and flow direction. Next, we demonstrate that the PINN method remains accurate for the backward ADEs, with the relative errors in most cases staying under 5% compared to the reference concentration field. Finally, we show that when available, the concentration measurements can be easily incorporated in the PINN method and significantly improve (by more than 50% in the considered cases) the accuracy of the PINN solution of the backward ADE.

Data Availability Statement

The data and codes used in this paper are available at <https://doi.org/10.5281/zenodo.5022650>.

Acknowledgments

This research was partially supported by the U.S. Department of Energy (DOE) Advanced Scientific Computing (ASCR) program. Pacific Northwest National Laboratory is operated by Battelle for the DOE under Contract DE-AC05-76RL01830.

References

- Almeida, R. C., & Silva, R. S. (1997). A stable petrov-galerkin method for convection-dominated problems. *Computer Methods in Applied Mechanics and Engineering*, 140(3–4), 291–304. [https://doi.org/10.1016/s0045-7825\(96\)01108-5](https://doi.org/10.1016/s0045-7825(96)01108-5)
- Atmadja, J., & Bagtzoglou, A. C. (2001). State of the art report on mathematical methods for groundwater pollution source identification. *Environmental Forensics*, 2(3), 205–214. <https://doi.org/10.1006/enfo.2001.0055>
- Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2015). Automatic differentiation in machine learning: A survey. *The Journal of Machine Learning Research*, 18, 1–43. <https://doi.org/10.1016/j.advwatres.2018.01.009>
- Berg, J., & Nyström, K. (2018). A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317, 28–41. <https://doi.org/10.1016/j.neucom.2018.06.056>
- Borker, R., Farhat, C., & Tezaur, R. (2017). A high-order discontinuous Galerkin method for unsteady advection–diffusion problems. *Journal of Computational Physics*, 332, 520–537. <https://doi.org/10.1016/j.jcp.2016.12.021>
- Brooks, A. N., & Hughes, T. J. (1982). Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32(1–3), 199–259. [https://doi.org/10.1016/0045-7825\(82\)90071-8](https://doi.org/10.1016/0045-7825(82)90071-8)
- Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16, 1190–1208. <https://doi.org/10.1137/0916069>
- Cai, S., Wang, Z., Fuest, F., Jeon, Y. J., Gray, C., & Karniadakis, G. E. (2021). Flow over an espresso cup: Inferring 3-d velocity and pressure fields from tomographic background oriented schlieren via physics-informed neural networks. *Journal of Fluid Mechanics*, 915, A102. <https://doi.org/10.1017/jfm.2021.135>
- Cecchi, M. M., & Pirozzi, M. A. (2005). High order finite difference numerical methods for time-dependent convection-dominated problems. *Applied Numerical Mathematics*, 55(3), 334–356. <https://doi.org/10.1016/j.apnum.2005.04.038>
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4), 303–314. <https://doi.org/10.1007/bf02551274>
- Ding, H., & Zhang, Y. (2009). A new difference scheme with high accuracy and absolute stability for solving convection-diffusion equations. *Journal of Computational and Applied Mathematics*, 230, 600–606. <https://doi.org/10.1016/j.cam.2008.12.015>
- Dwivedi, V., & Srinivasan, B. (2020). Physics informed extreme learning machine (PIELM)—A rapid method for the numerical solution of partial differential equations. *Neurocomputing*, 391, 96–118. <https://doi.org/10.1016/j.neucom.2019.12.099>
- Ewing, E. R., & Wang, H. (2001). A summary of numerical methods for time-dependent advection-dominated partial differential equations. *Journal of Computational and Applied Mathematics*, 128(1–2), 423–445. [https://doi.org/10.1016/S0377-0427\(00\)00522-7](https://doi.org/10.1016/S0377-0427(00)00522-7)
- Franca, L. P., & Farhat, C. (1995). Bubble functions prompt unusual stabilized finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 123(1–4), 299–308. [https://doi.org/10.1016/0045-7825\(94\)00721-x](https://doi.org/10.1016/0045-7825(94)00721-x)
- Franca, L. P., Frey, S. L., & Hughes, T. J. (1992). Stabilized finite element methods: I. Application to the advective-diffusive model. *Computer Methods in Applied Mechanics and Engineering*, 95(2), 253–276. [https://doi.org/10.1016/0045-7825\(92\)90143-8](https://doi.org/10.1016/0045-7825(92)90143-8)
- Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, 9, 249–256.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge, MA: MIT Press.
- Heinrich, J., Huyakorn, P., Zienkiewicz, O., & Mitchell, A. (1977). An upwind finite element scheme for two-dimensional convective transport equation. *International Journal for Numerical Methods in Engineering*, 11(1), 131–143. <https://doi.org/10.1002/nme.1620110113>
- He, Q. Z., Barajas-Solano, D., Tartakovsky, G., & Tartakovsky, A. M. (2020). Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Advances in Water Resources*, 141, 103610. <https://doi.org/10.1016/j.advwatres.2020.103610>
- Herrera, P. A., Valocchi, A. J., & Beckie, R. D. (2010). A multidimensional streamline-based method to simulate reactive solute transport in heterogeneous porous media. *Advances in Water Resources*, 33(7), 711–727. <https://doi.org/10.1016/j.advwatres.2010.03.001>
- Herrera, P., & Valocchi, A. (2006). Positive solution of two-dimensional solute transport in heterogeneous aquifers. *Groundwater*, 44(6), 803–813. <https://doi.org/10.1111/j.1745-6584.2006.00154.x>
- Hillman, M., & Chen, J. S. (2016). Nodally integrated implicit gradient reproducing kernel particle method for convection dominated problems. *Computer Methods in Applied Mechanics and Engineering*, 299, 381–400. <https://doi.org/10.1016/j.cma.2015.11.004>
- Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2, 359–366. [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8)
- Huang, Q., Huang, G., & Zhan, H. (2008). A finite element solution for the fractional advection-dispersion equation. *Advances in Water Resources*, 31(12), 1578–1589. <https://doi.org/10.1016/j.advwatres.2008.07.002>
- Hughes, T. J., Franca, L. P., & Hulbert, G. M. (1989). A new finite element formulation for computational fluid dynamics: VIII. The galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73, 173–189. [https://doi.org/10.1016/0045-7825\(89\)90111-4](https://doi.org/10.1016/0045-7825(89)90111-4)
- Hughes, T. J., Mallet, M., & Akira, M. (1986). A new finite element formulation for computational fluid dynamics: II. Beyond SUPG. *Computer Methods in Applied Mechanics and Engineering*, 54(3), 341–355. [https://doi.org/10.1016/0045-7825\(86\)90110-6](https://doi.org/10.1016/0045-7825(86)90110-6)
- Hughes, T. J., & Wells, G. N. (2005). Conservation properties for the Galerkin and stabilised forms of the advection-diffusion and incompressible Navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 194(9–11), 1141–1159. <https://doi.org/10.1016/j.cma.2004.06.034>
- Huyakorn, P. S. (2012). *Computational methods in subsurface flow*. academic press.
- Ingham, D., & Ma, L. (2005). Fundamental equations for CFD river flow simulations. In *Computational fluid dynamics. Applications in environmental hydraulics* (p. 19–49). Chichester, England: Wiley.
- Khodayi-Mehr, R., & Zavanos, M. M. (2019). VarNet: Variational neural networks for the solution of partial differential equations. *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, 120, 298–307. <http://arxiv.org/abs/1912.07443>
- Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization* (Vol. 1–15). <http://arxiv.org/abs/1412.6980>
- Kissas, G., Yang, Y., Hwuang, E., Witschey, W. R., Detre, J. A., & Perdikaris, P. (2020). Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 358, 112623. <https://doi.org/10.1016/j.cma.2019.112623>
- Lagaris, I. E., Likas, A. C., & Papageorgiou, D. G. (2000). Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5), 1041–1049. <https://doi.org/10.1109/72.870037>

- Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1997). Artificial neural networks for solving ordinary and partial differential equations. *arXiv.org*, 9(5), 26. <http://arxiv.org/abs/physics/9705023>
- Lee, H., & Kang, I. S. (1990). Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1), 110–131. [https://doi.org/10.1016/0021-9991\(90\)90007-N](https://doi.org/10.1016/0021-9991(90)90007-N)
- Lin, H., & Atluri, S. N. (2000). Meshless local Petrov-Galerkin (MLPG) method for convection-diffusion problems. *CMES - Computer Modeling in Engineering and Sciences*, 1(2), 45–60.
- Lipnikov, K., Shashkov, M., Svyatskiy, D., & Vassilevski, Y. (2007). Monotone finite volume schemes for diffusion equations on unstructured triangular and shape-regular polygonal meshes. *Journal of Computational Physics*, 227(1), 492–512. <https://doi.org/10.1016/j.jcp.2007.08.008>
- Lu, L., Meng, X., Mao, Z., & Karniadakis, G. E. (2019). DeepXDE: A deep learning library for solving differential equations (Vol. 1–17). <http://arxiv.org/abs/1907.04502>
- Mao, Z., Jagtap, A. D., & Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360, 112789. <https://doi.org/10.1016/j.cma.2019.112789>
- Mojtabi, A., & Deville, M. O. (2015). One-dimensional linear advection-diffusion equation: Analytical and finite element solutions. *Computers and Fluids*, 107, 189–195. <https://doi.org/10.1016/j.compfluid.2014.11.006>
- Neupauer, R. M., & Wilson, J. L. (1999). Adjoint method for obtaining backward-in-time location and travel time probabilities of a conservative groundwater contaminant. *Water Resources Research*, 35(11), 3389–3398. <https://doi.org/10.1029/1999wr900190>
- Noorshad, J., & Mehran, M. (1982). An upstream finite element method for solution of transient transport equation in fractured porous media. *Water Resources Research*, 18(3), 588–596. <https://doi.org/10.1029/wr018i003p00588>
- Noye, B. J., & Tan, H. H. (1988). A third-order semi-implicit finite difference method for solving the one-dimensional convection-diffusion equation. *International Journal for Numerical Methods in Engineering*, 26, 1615–1629. <https://doi.org/10.1002/nme.1620260711>
- Pang, G., Lu, L., & Karniadakis, G. E. (2019). fPINNS: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4), A2603–A2626. <https://doi.org/10.1137/18m1229845>
- Patil, S., & Chore, H. (2014). Contaminant transport through porous media: An overview of experimental and numerical studies. *Advances in Environmental Research*, 3(1), 45–69. <https://doi.org/10.12989/aer.2014.3.1.045>
- Pinder, G. F., & Gray, W. G. (2013). *Finite element simulation in surface and subsurface hydrology*. Elsevier.
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 686–707. <https://doi.org/10.1016/j.jcp.2018.10.045>
- Raissi, M., Yazdani, A., & Karniadakis, G. E. (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 4741, 1–1030. <https://doi.org/10.1126/science.aaw4741>
- Rao, C., Sun, H., & Liu, Y. (2020). *Physics informed deep learning for computational elastodynamics without labeled data* (Vol. 1–26). <http://arxiv.org/abs/2006.08472>
- Reyes, B., Howard, A. A., Perdikaris, P., & Tartakovsky, A. M. (2020). *Learning unknown physics of non-Newtonian fluids*.
- Rigal, A. (1994). High order difference schemes for unsteady one-dimensional diffusion-convection problems. *Journal of Computational Physics*, 114(1), 59–76. <https://doi.org/10.1006/jcph.1994.1149>
- Samaniego, E., Anitescu, C., Goswami, S., Nguyen-Thanh, V. M., Guo, H., Hamdia, K., et al. (2020). An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, 362, 112790. <https://doi.org/10.1016/j.cma.2019.112790>
- Shin, Y., Darbon, J., & Karniadakis, G. E. (2020). On the convergence and generalization of physics informed neural networks. *Computer Physics Communications*, 28, 2042–2074. <https://doi.org/10.4208/cicp.OA-2020-0193>
- Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375, 1339–1364. <https://doi.org/10.1016/j.jcp.2018.08.029>
- Sun, L., Gao, H., Pan, S., & Wang, J.-X. (2020). Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361, 112732. <https://doi.org/10.1016/j.cma.2019.112732>
- Tan, J., Tartakovsky, A. M., Ferris, K., & Ryan, E. M. (2016). Investigating the effects of anisotropic mass transport on dendrite growth in high energy density lithium batteries. *Journal of The Electrochemical Society*, 163(2), A318–A327. <https://doi.org/10.1149/2.0951602jes>
- Tartakovsky, A. M., Marrero, C. O., Perdikaris, P., Tartakovsky, G. D., & Barajas-Solano, D. (2020). Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems. *Water Resources Research*, 56, e2019WR026731. <https://doi.org/10.1029/2019WR026731>
- van der Meer, R., Oosterlee, C., & Borovikh, A. (2020). Optimally weighted loss functions for solving PDEs with neural networks. *arXiv preprint*, 1–34.
- Wang, H. Q., & Lacroix, M. (1997). Optimal weighting in the finite difference solution of the convection-dispersion equation. *Journal of Hydrology*, 200, 228–242. [https://doi.org/10.1016/S0022-1694\(97\)00020-6](https://doi.org/10.1016/S0022-1694(97)00020-6)
- Wang, S., Teng, Y., & Perdikaris, P. (2020). Understanding and mitigating gradient pathologies in physics-informed neural networks (Vol. 1–28). <http://arxiv.org/abs/2001.04536>
- Weinan, E., & Yu, B. (2018). The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1), 1–12. <https://doi.org/10.1007/s40304-018-0127-z>
- White, M. D., Oostrom, M., & Lenhard, R. J. (1995). Modeling fluid flow and transport in variably saturated porous media with the STOMP simulator. 1. Nonvolatile three-phase model description. *Advances in Water Resources*, 18, 353–364. [https://doi.org/10.1016/0309-1708\(95\)00018-E](https://doi.org/10.1016/0309-1708(95)00018-E)
- Wilson, J., & Liu, J. (1994). Backward tracking to find the source of pollution. *Water Manage. Risk Remedy*, 1, 181–199.
- Xiong, X.-T., Fu, C.-L., & Qian, Z. (2006). Two numerical methods for solving a backward heat conduction problem. *Applied Mathematics and Computation*, 179(1), 370–377. <https://doi.org/10.1016/j.amc.2005.11.114>
- Yadav, N., Yadav, A., & Kim, J. H. (2016). Numerical solution of unsteady advection dispersion equation arising in contaminant transport through porous media using neural networks. *Computers and Mathematics with Applications*, 72(4), 1021–1030. <https://doi.org/10.1016/j.camwa.2016.06.014>
- Zhu, Q., Liu, Z., & Yan, J. (2020). Machine learning for metal additive manufacturing: Predicting temperature and melt pool fluid dynamics using physics-informed neural networks. *Computational Mechanics*, 67, 619–635. <https://doi.org/10.1007/s00466-020-01952-9>