Physics of Fluids

RESEARCH ARTICLE | MAY 24 2023

# Ensemble physics informed neural networks: A framework to improve inverse transport modeling in heterogeneous domains

Maryam Aliakbari; Mohammadreza Soltany Sadrabadi ; Peter Vadasz ; Amirhossein Arzani



*Physics of Fluids* 35, 053616 (2023)
https://doi.org/10.1063/5.0150016

Physics of Fluids ARTICLE pubs.aip.org/aip/pof

# Ensemble physics informed neural networks: A framework to improve inverse transport modeling in heterogeneous domains

Cite as: Phys. Fluids **35**, 053616 (2023); doi: 10.1063/5.0150016
Submitted: 10 March 2023 · Accepted: 10 May 2023 ·
Published Online: 24 May 2023



Maryam Aliakbari,[1] Mohammadreza Soltany Sadrabadi,[1] Peter Vadasz,[1] and Amirhossein Arzani[2,3,a)]

## AFFILIATIONS

[1]Department of Mechanical Engineering, Northern Arizona University, Flagstaff, Arizona 86011, USA
[2]Department of Mechanical Engineering, University of Utah, Salt Lake City, Utah 84112, USA
[3]Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, Utah 84112, USA

[a)]Author to whom correspondence should be addressed: amir.arzani@sci.utah.edu

## ABSTRACT

Modeling fluid flow and transport in heterogeneous systems is often challenged by unknown parameters that vary in space. In inverse modeling, measurement data are used to estimate these parameters. Due to the spatial variability of these unknown parameters in heterogeneous systems (e.g., permeability or diffusivity), the inverse problem is ill-posed and infinite solutions are possible. Physics-informed neural networks (PINN) have become a popular approach for solving inverse problems. However, in inverse problems in heterogeneous systems, PINN can be sensitive to hyperparameters and can produce unrealistic patterns. Motivated by the concept of ensemble learning and variance reduction in machine learning, we propose an ensemble PINN (ePINN) approach where an ensemble of parallel neural networks is used and each sub-network is initialized with a meaningful pattern of the unknown parameter. Subsequently, these parallel networks provide a basis that is fed into a main neural network that is trained using PINN. It is shown that an appropriately selected set of patterns can guide PINN in producing more realistic results that are relevant to the problem of interest. To assess the accuracy of this approach, inverse transport problems involving unknown heat conductivity, porous media permeability, and velocity vector fields were studied. The proposed ePINN approach was shown to increase the accuracy in inverse problems and mitigate the challenges associated with non-uniqueness.



## I. INTRODUCTION

Recent advances in artificial intelligence (AI) have motivated new research directions in scientific computing for modeling physical systems. Traditionally, the broad field of AI has been divided into deductive reasoning and inductive learning.[1] In deductive reasoning, general and often interpretable domain knowledge is used to formulate a problem and guide predictions that move from general knowledge to specific conclusions in a typically mathematically well-defined fashion. In inductive learning, one starts from specific observations and moves toward a generalized theory in a data-driven and statistical manner. In modeling complex physical systems, pure domain knowledge or pure data are often not sufficient for real-world applications.[2] Therefore, new approaches where the deductive and inductive approaches are combined have emerged.[2] Physics-informed neural networks (PINN) are a recent paradigm in this area where governing equations in the form of differential equations are combined with measurement data to solve physical systems.[3,4]

While solving well-posed forward problems with PINN is currently not as efficient as traditional numerical approaches, inverse problems on the other hand have been a promising area for PINN research. The simplicity and flexibility of PINN models implemented in modern software frameworks allow one to apply the same code with minor modifications to study both forward and inverse problems, while this is a tedious task using traditional inverse modeling approaches such as the adjoint method.[5] Inverse problems arise in different scenarios. For instance, in practice, it might be experimentally easier to measure a certain physical variable to infer about a variable that is more difficult to measure (e.g., inferring fluid flow velocity from concentration measurements[6]). Solving problems where certain boundary/initial conditions or parameters are unknown but instead measurement data are available[7] and data-driven design of constitutive properties to achieve a desired task (e.g., material design[8]) are some other examples. PINN has been applied to various inverse modeling problems that arise in different fields such as heat transfer,[9,10] fluid mechanics,[11–13] and solid mechanics,[14] among others.

A key fundamental challenge in solving inverse problems is the lack of a unique solution. This is particularly exacerbated when trying to infer heterogeneous domain properties (e.g., spatially varying material properties) with sparse measurements as there are often many possible solutions. Modeling transport in heterogeneous domains poses different challenges.[15,16] Conceptually, the unknown heterogeneous parameters do not necessarily depend continuously on the measured data, which results in high sensitivity of the output of an inverse problem (the unknown parameter) to small perturbations in the input data.[17] Mathematically, this could be explained by considering an analogy with a linear system of equations $\mathbf{Ax} = \mathbf{b}$ in a linear regression problem where the data matrix $\mathbf{A}$ is a rectangular matrix. The solution to this problem is $\mathbf{x} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{b}$. However, $(\mathbf{A}^\top\mathbf{A})^{-1}$ is not necessarily invertible (an ill-posed problem) or could be ill-conditioned making the solution very sensitive to small perturbations in input data. A possible solution is to use Tikhonov regularization ($L_2$ regularization), which results in a solution $\mathbf{x} = (\mathbf{A}^\top\mathbf{A} + \lambda I)^{-1}\mathbf{A}^\top\mathbf{b}$, where $\lambda > 0$ is a regularization parameter, which enforces the solution to be smaller in an $L_2$ metric. The new matrix $\mathbf{A}^\top\mathbf{A} + \lambda I$ is always invertible,[18] and we now have a stable solution. In this example, the regularization could be perceived as domain knowledge (deductive reasoning) that augments the inductive approach to machine learning.

The above example highlights the lack of robustness in certain machine learning tasks. Robustness is a key issue and research direction in scientific machine learning.[19] Scientific machine learning models need to be robust with respect to small perturbations in their parameters and the input data. Similar to how deductive reasoning in the above example improved robustness, there is a need for new approaches in inverse problems that can improve the non-uniqueness challenge and sensitivity to model parameters. Specifically, the non-uniqueness issue in inverse modeling with PINN has received less attention. In an exploratory analysis, in the Appendix, we demonstrate that different strategies and traditional regularization approaches for inverse modeling with PINN lead to different results in identifying the unknown heterogeneous parameters that could be different from their ground-truth values. Interestingly, all of these results are "correct" in the sense that they minimize the loss function. In other words, the challenge in inverse modeling with PINN is not due to the lack of expressive power of deep neural networks but rather due to the optimization challenges associated with the complex loss landscape[20,21] in addition to the fundamental challenge discussed above, which is inherent to all inverse problems. Different neural network design strategies can land the network to different local minima and therefore different answers to the inverse problem.

In this study, we hypothesized that an ensemble of parallel neural networks initialized with an arbitrary but meaningful pattern of the unknown parameter could be used together with transfer learning to design an ensemble physics-informed neural network (ePINN) framework to guide the PINN solution toward a more robust and accurate solution. Ensemble neural networks have been used for uncertainty quantification in traditional deep learning models.[22] Additionally, neural additive models that rely on an ensemble of neural networks have been proposed to improve the interpretability of neural network predictions.[23] The idea of averaging different predictions in an ensemble fashion for variance reduction and improving predictions in machine learning is not new and is known as bagging or bootstrap aggregating.[24,25]

The manuscript is organized as follows. The problem statement, the proposed ePINN method, and test cases are explained in Sec. II. In Sec. III, the results of the proposed approach are presented and compared with the vanilla PINN method. The proposed approach and results are discussed in Sec. IV.

## II. METHODS

### A. Problem statement

We consider the governing equations as follows:

$$L(\mathbf{u}(\mathbf{x}); k(\mathbf{x})) = 0 \quad \mathbf{x} \in \Omega, \tag{1a}$$

$$B(\mathbf{u}(\mathbf{x})) = \mathbf{g} \quad \mathbf{x} \in \partial\Omega, \tag{1b}$$

where $L$ is the differential operator representing the governing equations, $k(\mathbf{x})$ is a model parameter in the form of a scalar or vector field, and $B$ is a boundary condition operator that determines the specified boundary conditions on the boundary $\partial\Omega$. $\mathbf{u}(\mathbf{x})$ represents a variable like velocity or temperature as a function of space, where $\mathbf{x} \in \Omega$. In this paper, we present examples where the operator $L$ is the diffusion, convection–diffusion, Darcy, and Navier–Stokes equations and consider multiphysics coupling between these physics. In a usual forward problem, we are given the boundary conditions, parameters, and the operator, and the solution $\mathbf{u}(\mathbf{x})$ could be obtained using standard numerical methods. In this paper, we focus on inverse problems where one of these essential pieces of information is not provided, rendering the problem ill-defined. In inverse problems, instead, we are provided with a set of measurement data $D$ in the form

$$D = \{(\mathbf{x}_i, \mathbf{u}_i)\,;\, i = 1, 2, \ldots, n\}, \tag{2}$$

where the solution is provided at $n$ measurement points $\mathbf{x}_i$. In this work, we are interested in inverse problems, where the parameter appearing in the governing equation is a function of space $[k(\mathbf{x})]$ and is unknown. Therefore, our problem is a parameter identification problem. In our examples, this represents an unknown heterogeneous heat conductivity in a solid domain, permeability in a porous medium, and a fluid flow velocity vector field. As discussed above, PINN could be used to solve this problem. PINN formulates a nonlinear optimization problem where the solution $\mathbf{u}(\mathbf{x})$ and the unknown parameter $k(\mathbf{x})$ are simultaneously obtained such that the governing equations [Eq. (1)] and the provided measurement data [Eq. (2)] are satisfied. Alternatively, classical optimization approaches could also be used to solve these inverse problems.[5] A key fundamental challenge in all of these approaches is the lack of a unique solution. In heterogeneous domains, there are many possible correct parameters $k(\mathbf{x})$ with very different patterns. In this study, we hypothesized that prior knowledge about some basic possible patterns in the parameter $k(\mathbf{x})$ could be used in designing the PINN architecture to mitigate this challenge.

### B. Physics-informed neural network (PINN)

First, we provide a very brief conceptual overview of PINN and refer the readers to the original work for a detailed presentation.[3] In PINN, we are interested in solving differential equations usually by also considering arbitrary specified measurement data. Two key fundamental features of neural networks are used in formulating PINN. First, neural networks are function approximators; therefore, we represent each dependent variable in our governing equation as a function

of space/time using a neural network where space (x,y,z) and/or time (t) are input parameters and the output of the neural networks is the dependent variable of interest **u**. Second, using automatic differentiation the partial derivative of each output with respect to any input could be calculated and differential equations could be formed by combining these partial derivatives. The parameters of the neural networks (function approximations) are obtained by formulating a global optimization problem such that the governing equations, boundary conditions, and specified data are satisfied.

In our inverse problem, we also represent the unknown heterogeneous parameter $k$ as a function of space using an additional neural network, which is optimized along with the dependent variables. The weights and biases of the neural networks representing the function approximations for the dependent variable **u** and parameter $k$ are optimized such that the following loss function is minimized

$$\mathcal{L} = \lambda_E \mathcal{L}_F + \lambda_b \mathcal{L}_{BC} + \lambda_d \mathcal{L}_{data}, \tag{3}$$

where the governing equations (Navier–Stokes, convection–diffusion, diffusion, and Darcy equations) are used as the physics loss function $\mathcal{L}_F$, $\mathcal{L}_{BC}$ represents the residuals of the boundary conditions, and $\mathcal{L}_{data}$ represents the residuals of the measurement data points, which are minimized to enforce these conditions. Positive hyperparameters $\lambda_E$, $\lambda_b$, and $\lambda_d$ are used for weighting the contribution of each term. $\lambda_E = 1$ is considered in all cases. Fully connected neural networks are used for all variables.

### C. Ensemble physics-informed neural network (ePINN)

In this work, we propose ePINN to mitigate the non-uniqueness issue in inverse problems and produce more meaningful solutions. Our proposed approach could also improve the convergence accuracy of PINN as shown later in the results. In ePINN, an ensemble of parallel neural networks is combined with a main network as shown in Fig. 1 to represent the unknown parameter $k(\mathbf{x})$,

$$\text{Parallel sub-networks} \begin{cases} X_i^j = \sigma(W_i^j X_i^{j-1} + b_i^j), & 1 \le j \le L-1, \quad (4a) \\ X_i^j = W_i^j X_i^{j-1} + b_i^j, & j = L, \quad (4b) \end{cases}$$

$$\text{Main network} \begin{cases} Z^k = \sigma(W'^k [X_1^L, \ldots, X_N^L] + b'^k), & k = 1, \quad (5a) \\ Z^k = \sigma(W'^k Z^{k-1} + b'^k), & 2 \le k \le L-1, \quad (5b) \\ Z^k = W'^k Z^{k-1} + b'^k, & k = L, \quad (5c) \end{cases}$$

where $X_i$ $(i = 1, \ldots, N)$ represent the N parallel neural networks and $Z$ refers to the main neural network. In this study, $N = 10$ was used in all cases to define ten different patterns. The superscripts j and k are used to identify the layers in each sub-network and the main neural network, respectively, and they range from 1 to L. $W^j$ and $b^j$ denote weight and bias of the jth layer in each parallel network, and $W'^k$ and $b'^k$ denote weight and bias of the $k$th layer in the main neural network.

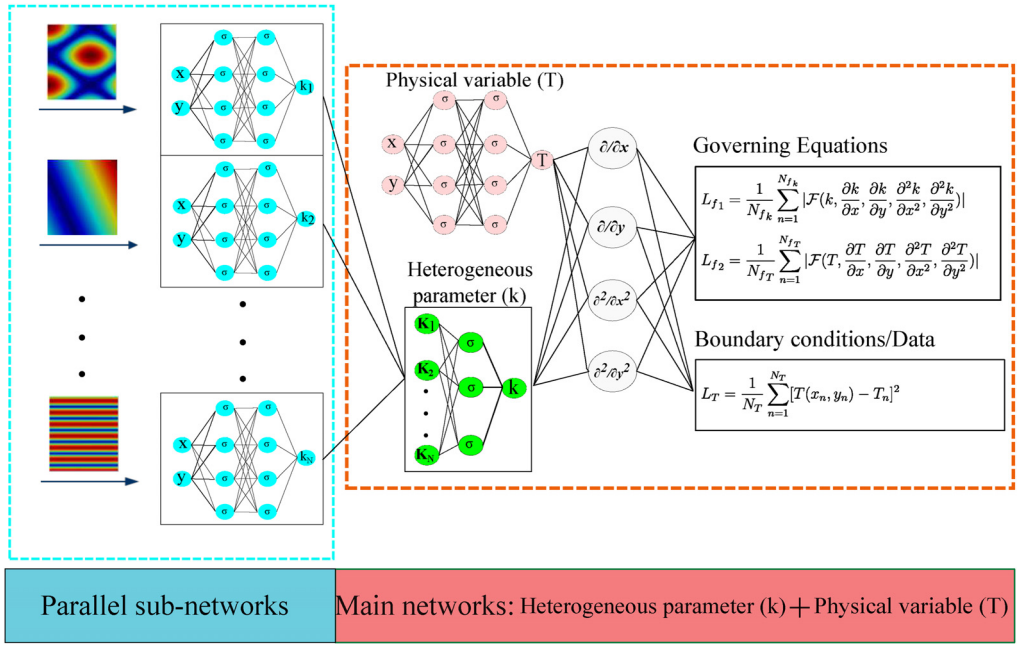


**FIG. 1.** A schematic overview of the proposed ePINN approach. A set of $N$ parallel networks define $k_i$ patterns for the unknown heterogeneous parameter. Each of these patterns is pre-defined for an arbitrary smooth function that represents prior knowledge about possible patterns and ranges in the solution. Each of these parallel networks is trained to represent these smooth patterns. Namely, $k_i$ vs input coordinates data are sampled from the pre-defined functions and are used as training data to train these $N$ parallel neural networks in a data-driven fashion without any physics. The output of each parallel network $(k_1, k_2, \ldots, k_N)$ is fed into the input layer of the part of the main neural network that represents the final heterogeneous parameter $k$. The physical variable $T$ is represented with a separate neural network. Finally, the physical variable $T$ and the heterogeneous parameter $k$ are solved using PINN.

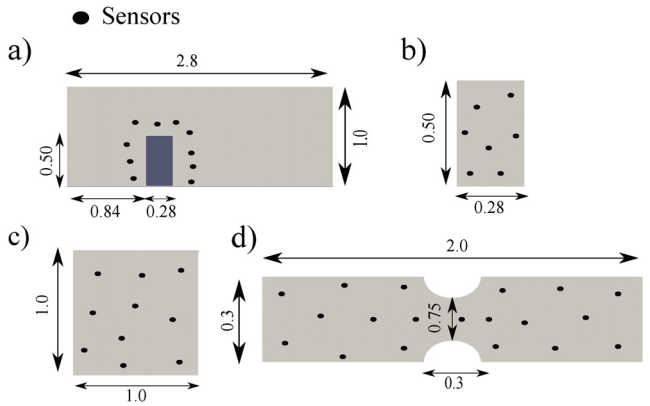


**FIG. 2.** The models used for each test case are shown. The black dots show the sensor locations. (a) Test case 1: multiphysics heat transfer in a fin, (b) test case 2: diffusion in a rectangle, (c) test case 3: porous medium transport, and (d) test case 4: blood flow in a blocked vessel.

$\sigma$ is a nonlinear activation function. Equation (4) shows how the output of each sub-network $X_i$ is calculated. Equation (5) shows how the parallel sub-network outputs are fed into the main network to generate the final prediction.

Each parallel sub-network is initialized with a meaningful pre-specified pattern. The output of each sub-network is used as the input to the main neural network to approximate the unknown parameter of the system. These arbitrary patterns could be a standard smooth function that provides a meaningful spatial variation pattern and consists of a range of values relevant to the problem of interest. We could think about these patterns as basis functions that are composed in a nonlinear fashion by ePINN's main network to guide the final approximation of the unknown parameter $k(\mathbf{x})$. The functions that are used to generate these initial patterns for each test case are provided in Table II in the Appendix. Each sub-network is trained to represent one of these pre-specified functions in a data-driven fashion. Subsequently, using transfer learning, the weights and biases for these sub-networks are frozen and only the main network is trained. We also provide a comparison to the case where the parallel networks are initialized with the patterns but not frozen, which could be perceived as a case where the basis functions are initialized but also allowed to be changed during training.

## D. Test case problem formulation

We illustrate the performance of the ePINN method in the context of four benchmark examples. An overview of the geometry used in these problems and the location of the sparse measurement data used to solve the inverse problem are shown in Fig. 2. For all test cases, the data-driven deep neural networks representing the parallel sub-networks (pre-specified patterns) were trained with 200 epochs with a constant learning rate of $3 \times 10^{-3}$. The final ePINN simulation used 10 000 epochs with a learning rate varying between $3 \times 10^{-4}$ and $3 \times 10^{-6}$. For all test cases, five sets of simulations were conducted: (1) ePINN where all parallel sub-networks are frozen and not updated. (2) A variant of ePINN where the sub-networks are initialized with the pre-defined patterns but updated during training. (3) A PINN representation of ePINN where ePINN architecture is used but all layers are randomly initialized. (4) A vanilla PINN network with a similar size as ePINN. (5) A larger vanilla PINN architecture. For all test cases, the Swish activation function and constant $\lambda$ weights were used except for test case 4 where an adaptive activation function[26] and trainable weight $\lambda$[27] were used for improved performance.

For each test case, a computational fluid dynamics (CFD) simulation was carried out to validate the results and generate sparse measurements. Test cases 1 and 2 were performed in the finite volume solver ANSYS Fluent where the least-square cell-based method for gradient calculations and the second-order upwind method for all equations (momentum and energy) were selected. Test cases 1 and 2 were modeled with a total number of 60k and 35k quadrilateral elements, respectively. Test cases 3 and 4 were performed in the finite element solver FEniCS with a total number of 10k and 19k triangular elements, respectively, and with quadratic (second-order) shape functions. The ground-truth functions for the heterogeneous parameters that were used to generate the CFD results are shown in Table I. All problems defined below are dimensionless.

### 1. Test case 1: Infer heterogeneous heat conductivity in 2D multiphysics heat transfer in a fin

As the first test case, we solved a 2D steady-state problem over a solid rectangular fin ($[0.84, 1.12]\times[0, 0.5]$) located inside a fluid domain ($[0, 2.8]\times[0, 1]$) as shown in Fig. 2. The fluid flow and convective heat transfer around the fin were coupled to heat conduction in the fin.

The governing equations include the incompressible Navier–Stokes equations and convection–diffusion equation in the fluid coupled with heterogeneous conduction in the solid

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0, \tag{6a}$$

$$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right), \tag{6b}$$

$$u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right), \tag{6c}$$

$$u\frac{\partial T_f}{\partial x}+v\frac{\partial T_f}{\partial y}-k_f\left(\frac{\partial^2 T_f}{\partial x^2}+\frac{\partial^2 T_f}{\partial y^2}\right)=0, \tag{6d}$$

$$\nabla\cdot(k_s\nabla T)=\frac{\partial k_s}{\partial x}\frac{\partial T_s}{\partial x}+\frac{\partial k_s}{\partial y}\frac{\partial T_s}{\partial y}+k_s\left(\frac{\partial^2 T_s}{\partial x^2}+\frac{\partial^2 T_s}{\partial y^2}\right)=0, \tag{6e}$$

**TABLE I.** The functions k(x,y) that were used to define the unknown parameter in each test case.

| Test case 1 | Test case 2 | Test case 3 | Test case 4 |
|---|---|---|---|
| $y$ | $\sin(5x)+\sin(10y)+\cos(20y)$ | $\exp(\sin(10x)+y)$ | Velocity vector from CFD simulation |

where $u$ and $v$ are velocity in $x$ and $y$ direction; $T_f$ and $T_s$ denote temperature in the fluid and solid domains, respectively; and a thermal diffusivity (the reciprocal of the fluid Peclet number) of $k_f = 0.02$ was considered for the fluid. A dimensionless density $\rho = 1$ and kinematic viscosity (the reciprocal of the Reynolds number) $\nu = 0.01$ were selected with a parabolic velocity profile at the inlet producing a peak velocity of $u = 0.5$. No-slip boundary condition was used at the walls. $k_s(x, y)$ is the unknown spatially varying heat conductivity in the solid that we are interested in estimating. For thermal boundary conditions, the inlet temperature was set to zero and the base temperature of the fin was set to one (non-dimensional). At the fluid–solid interface, equal heat flux and temperature of the solid and fluid were enforced. Zero flux was assumed at the other boundaries. In generating the measurement data with CFD simulations, the heat conductivity of the solid was given as a function of space as defined in Table I. The goal of the inverse problem is to identify this conductivity using the sparse temperature measurement data sampled at points shown in Fig. 2(a). It is noteworthy that in this example the sparse measurements are inside the fluid domain and the goal is to find the conductivity in the solid domain.

The number of hidden layers to approximate velocity and pressure was eight with 150 neurons per layer, while nine hidden layers and 180 neurons per layer were selected for approximating temperature. The neural network to approximate heat conductivity $k_s$ in ePINN consisted of ten sub-networks each including three hidden layers and 40 neurons per layer connected to the main neural network with four hidden layers and 40 neurons per layer. $\lambda_b = 30$ and $\lambda_d = 10$ were used to combine the boundary condition and data loss, respectively.

### 2. Test case 2: Infer heterogeneous heat conductivity in 2D diffusion

As the second test case, we consider pure diffusion in a heterogenous solid and use a complex unknown conductivity pattern (Table I) to highlight the effectiveness of the ePINN approach in solving more complex patterns of the unknown parameter. The geometry is a rectangle in the region $[0, 0.28]\times[0, 0.5]$ [Fig. 2(b)] that is held at a constant hot temperature at the bottom ($T = 1$) and low temperature at the top ($T = 0$) with insulated side walls. It is assumed that the heat conductivity of the solid $k_s$ is unknown and varies with the domain position (Table I). The governing equation is a steady diffusion equation with a constant heat source

$$\nabla \cdot (k_s \nabla T) = 1\,. \tag{7}$$

We have sparse sensors of temperature in the solid domain [Fig. 2(b)], and the goal is to find the heat conductivity of the solid and temperature patterns inside the domain. The number of hidden layers to approximate temperature was nine with 200 neurons per layer. The neural network to approximate heat conductivity consisted of ten sub-networks each including three hidden layers and 40 neurons per layer and a final neural network with four hidden layers and 40 neurons per layer. $\lambda_b = 30$ and $\lambda_d = 30$ were used.

### 3. Test case 3: Infer heterogeneous permeability in 2D porous medium transport

As the next test case, a square in the region $[0,1]\times[0,1]$ filled with a fluid with a dynamic viscosity of $\mu = \frac{\mu_*}{\mu_0} = 10$ (where $\mu_*$ and $\mu_0$ represent dimensional dynamic viscosity and the reference value of dynamic viscosity, respectively) is considered. The governing equations are continuity and Darcy equations

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{8a}$$

$$u + \frac{k}{\mu}\frac{\partial p}{\partial x} = 0, \tag{8b}$$

$$v + \frac{k}{\mu}\frac{\partial p}{\partial y} = 0, \tag{8c}$$

where $p$ is pressure, and $u$ and $v$ represent the components of the Darcy flux (filtration velocity). $k$ is the heterogeneous permeability, which is unknown in this problem. The free slip boundary condition is applied at the top and bottom walls. Also, high pressure at the left wall ($p = 1$) and low pressure at the right wall ($p = 0$) are assumed. Since the permeability is assumed unknown (Table I), we have some sparse measurements of the Darcy flux (filtration velocity) inside the domain as shown in Fig. 2(c) and the goal is to solve the problem to find the unknown permeability, pressure, and Darcy flux (filtration velocity).

Six hidden layers with 170 neurons per layer were selected for the neural networks to approximate pressure and Darcy flux. The neural network to approximate permeability consisted of ten sub-networks each including five hidden layers and 60 neurons per layer and the main neural network with three hidden layers and 60 neurons per layer was added to the parallel neural networks. $\lambda_b = 60$ and $\lambda_d = 80$ were used.

### 4. Test case 4: Infer velocity in 2D flow in a stenosed vessel with mass transport

As the last test case, we consider a steady 2D multiphysics problem inside an idealized stenosed artery [Fig. 2(d)], which is a common cardiovascular disease where blood flow patterns are of interest.[28,29] Data are generated by solving the Navier–Stokes equations coupled with convection–diffusion transport. In the inverse problem, we are interested in using sparse measurements of concentration to infer the unknown steady velocity vector field. We only use the continuity equation and the convection–diffusion equation to find the velocity

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{9a}$$

$$u\frac{\partial C}{\partial x} + v\frac{\partial C}{\partial y} - D\left(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2}\right) = 0, \tag{9b}$$

where $u$ and $v$ are velocity in $x$ and $y$ direction, $C$ denotes concentration, and a diffusion coefficient (the reciprocal of the mass transfer Peclet number) of $D = 0.05$ was considered. In the inverse problem, we assume that there is no boundary condition available for the flow field, and instead, we have some sparse measurements of concentration in the domain as shown in Fig. 2(d). By using this information, the goal is to solve for velocity and concentration inside the entire domain. In terms of concentration boundary condition, a constant Neumann boundary condition of $-\frac{\partial C}{\partial n} = 0.0001$ was prescribed at the wall and zero concentration was used at the inlet. To generate data, a parabolic velocity profile with a peak Reynolds number of $\mathrm{Re} = 150$

was applied at the inlet, no-slip BCs were assumed at the walls, and zero pressure was used at the outlet.

The neural networks used in approximating the concentration had eight hidden layers with 200 neurons per layer. The ePINN network to approximate velocity consisted of ten sub-networks each with five hidden layers and 100 neurons per layer and a final neural network with three hidden layers and 100 neurons per layer. For this problem, we have used a neural network with an adaptive swish activation function[26] and an adaptive $\lambda_b$ and $\lambda_d$[27] for more accurate solutions.

## III. RESULTS

In the Appendix (Sec. V), we demonstrate the non-uniqueness challenge in solving inverse problems in heterogeneous domains. We show how various solutions are obtained by using different PINN training strategies for our first test case problem. These results motivate the proposed ePINN strategy to guide the solution. In the following, we present the ePINN results for the four different test case problems and compare them to PINN. For the first two test cases, we also investigated how increasing the number of layers affected the convergence in the vanilla PINN approach.

### A. Test case 1: Infer heterogeneous heat conductivity in 2D multiphysics heat transfer in a fin

The heat conductivity found by ePINN and PINN is shown in Fig. 3(a) for the first test case. The results show that only heat conductivity found by the ePINN approach matches the ground truth. An ePINN architecture without freezing the parallel sub-networks (either initialized with transfer learning or randomly initialized) did not converge to the ground-truth solution and PINN yielded a very different pattern. The error obtained by ePINN is shown in Fig. 3(b), and small localized errors could be seen at the boundary. The temperature patterns obtained by ePINN agree very well with the ground-truth results as shown in Fig. 3(c). Interestingly, the loss vs epoch plot shows that ePINN can accelerate convergence compared to a PINN approach with similar architecture to ePINN's trainable main network [Fig. 3(d)]. However, a PINN approach with a larger network structure provides a smaller loss (more accurate temperature predictions) but finds a heat conductivity that is different from the ground truth (non-uniqueness challenge).

### B. Test case 2: Infer heterogeneous heat conductivity in 2D diffusion

The results for test case 2 are shown in Fig. 4. The heat conductivity contours for the ground truth, ePINN, and PINN predictions are shown [Fig. 4(a)]. The absolute error with respect to the ground truth shown in Fig. 4(b) demonstrates that in this test case, the ePINN approach cannot precisely recover the ground truth pattern because of the complex spatial variability of the heterogeneous parameter in this test case compared to the previous example. However, ePINN captures the qualitative pattern pretty well, while PINN converges to a very different pattern. The temperature patterns obtained by ePINN match the ground truth very well [Fig. 4(c)]. The loss vs epoch plot in Fig. 4(d) shows the same trend as the previous test case where the ePINN approach can accelerate PINN convergence and improve accuracy compared to a PINN network with similar architecture as the main network of ePINN. Similar to the last case, a more expressive PINN network can improve vanilla PINN's prediction of temperature; however, it finds a different conductivity.

### C. Test case 3: Infer heterogeneous permeability in 2D porous medium transport

The solution for test case 3 is shown in Fig. 5. The results show that the ePINN approach significantly improves the discovery of the ground-truth permeability pattern with very small errors. Also, the pressure and Darcy flux patterns [Figs. 5(c) and 5(d)] found by ePINN are in very good agreement with the ground truth.

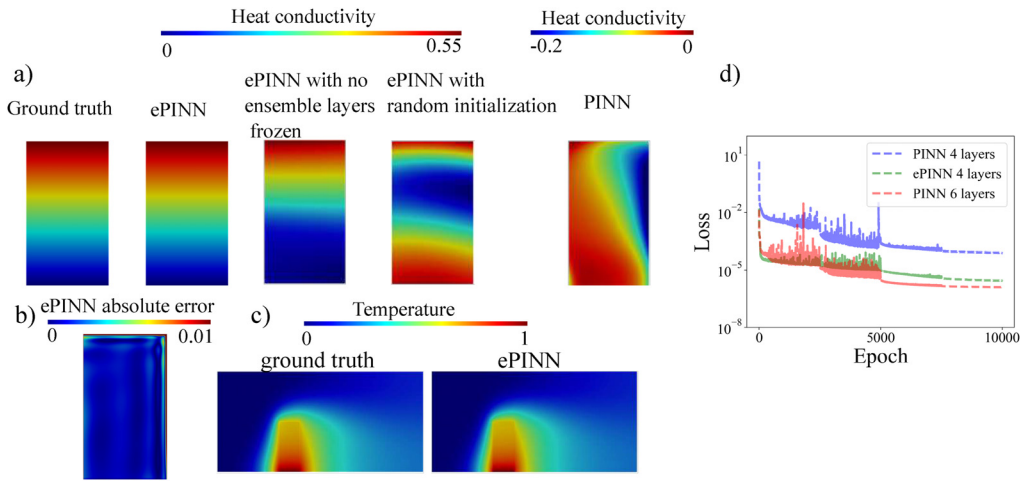


**FIG. 3.** (a) The ground truth, ePINN (ensemble layers frozen), ePINN with no ensemble layers frozen, ePINN with random initialization, and PINN results for heat conductivity in the 2D multiphysics heat transfer problem (test case 1) are shown. (b) The ePINN absolute error with respect to the ground-truth data is shown. (c) The ground truth and ePINN temperature patterns are shown. (d) The mean square error (MSE) loss vs the epoch (iteration) number is plotted to compare ePINN convergence with PINN (same neural network size as the main ePINN network and a larger neural network size with six hidden layers and 130 neurons per layer).

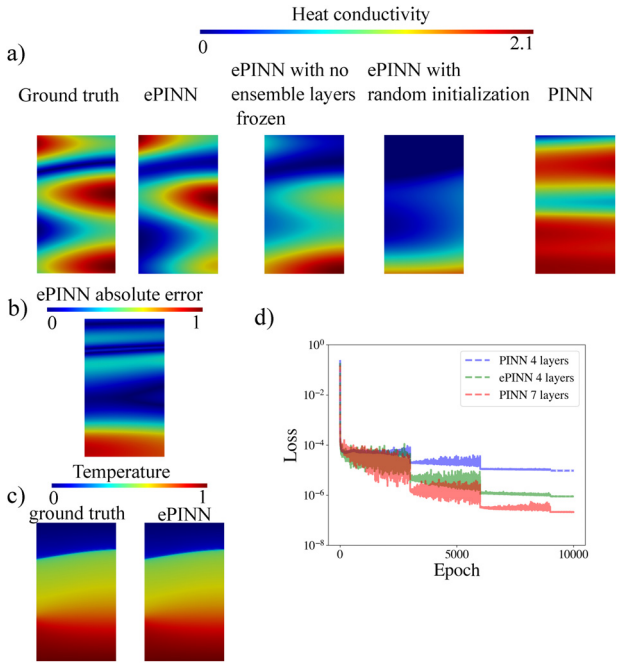


**FIG. 4.** (a) The ground truth, ePINN (ensemble layers frozen), ePINN with no ensemble layers frozen, ePINN with random initialization, and PINN heat conductivity results for the 2D diffusion problem (test case 2) are shown. (b) The ePINN absolute error with respect to the ground-truth data is shown. (c) The ground truth and ePINN temperature patterns are shown. (d) The mean square error (MSE) loss vs the epoch (iteration) number is plotted to compare ePINN convergence with PINN (same neural network size as the main network used in ePINN and larger neural network size with seven hidden layers and 150 neurons per layer).

### D. Test case 4: Infer velocity in 2D flow in a stenosed vessel with mass transport

The velocity and concentration results from test case 4 simulation are shown in Fig. 6. In this problem, the goal was to identify the velocity vector field that produced sparse concentration measurements. It could be seen that the ePINN approach is efficient in finding velocity vectors without any boundary conditions available just by using sparse concentration data. The PINN approach obtains velocity vector fields that are unrealistic (e.g., they demonstrate backflow at the region upstream of the blocked artery). The other panels in Fig. 6(a) demonstrate the very different and physically unrealistic velocity patterns obtained by different approaches. The absolute error in Fig. 6(b) shows a localized high error near the wall regions at the inlet. This can be attributed to the fact that the flow becomes more fully developed as it progresses and subsequently becomes less sensitive to the inlet profile. Figure 6(c) shows that the concentration pattern found by ePINN matches the ground truth very well. Moreover, Fig. 6(d) shows that the adaptive activation function and weights reduce the loss function up to two orders of magnitude and provide more accurate solutions.

### E. Sensitivity analysis

Figure 7 displays the impact of the pre-defined initial patterns and neural network size and some hyperparameters on test case 1. To examine the sensitivity of ePINN to the sub-network initialization patterns, all coefficients in Table II were increased by 10%. Additionally, in a separate simulation, a different neural network structure size and loss weight $\lambda$ were employed. The main neural network here was assumed to have three hidden layers and 50 neurons per layer with $\lambda_b = 20$ and $\lambda_d = 15$ used to merge the boundary condition and data loss. The ground-truth heat conductivity is shown in Fig. 7(a). The heat conductivity pattern obtained by ePINN with perturbed initial

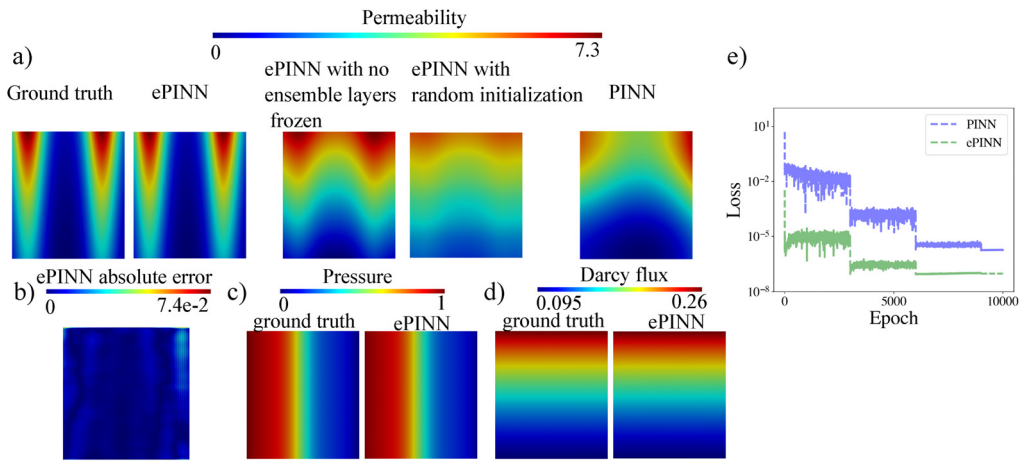


**FIG. 5.** (a) The ground truth, ePINN (ensemble layers frozen), ePINN with no ensemble layers frozen, ePINN with random initialization, and PINN permeability results for the 2D heterogeneous porous medium (test case 3) are shown. (b) The ePINN absolute error with respect to the ground-truth data is shown. (c) The ground truth and ePINN pressure patterns are shown. (d) The ground truth and ePINN Darcy flux (filtration velocity) patterns are shown. (e) The mean square error (MSE) loss vs the epoch (iteration) number is plotted to compare ePINN convergence with PINN (same neural network structure size as the main network used in ePINN).

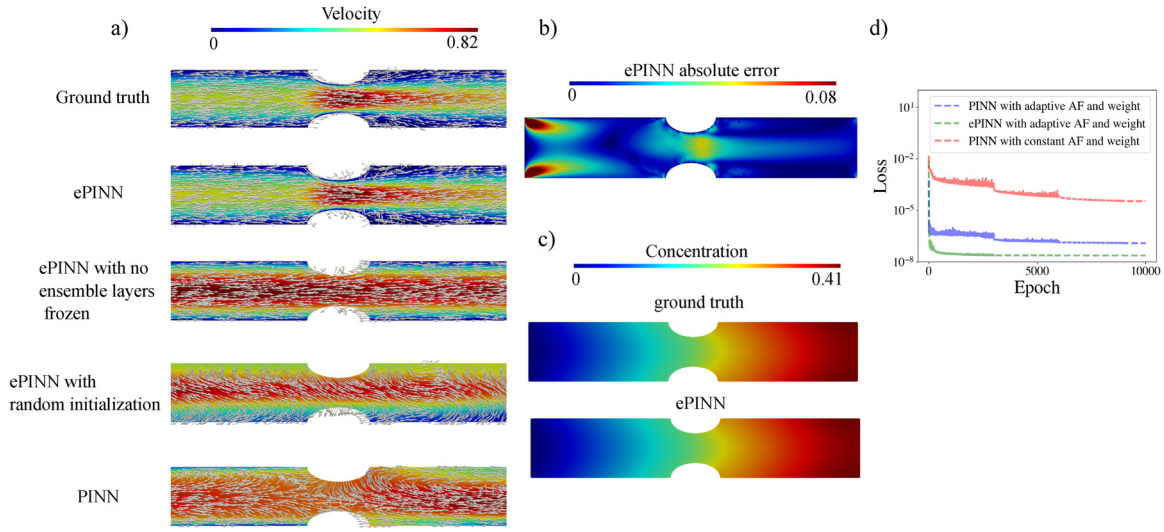


**FIG. 6.** (a) The ground truth, ePINN (ensemble layers frozen), ePINN with no ensemble layers frozen, ePINN with random initialization, and PINN velocity results for the 2D stenosis (test case 4) are shown. (b) The ePINN absolute error with respect to the ground-truth data is shown. (c) The ground truth and ePINN concentration patterns are shown. (d) The mean square error (MSE) loss vs the epoch (iteration) number is plotted to compare ePINN convergence with PINN. Vanilla PINN simulation convergence with and without adaptive activation function (AF) and loss weights are compared.

patterns is shown in Fig. 7(b), while the impact of neural network structure size and hyperparameters is shown in Fig. 7(c). Although using a perturbed initial pattern and hyperparameters slightly increases the absolute error, it still produces qualitatively accurate results compared to PINN, which was more sensitive to hyperparameters, as shown in the Appendix (Fig. 9).

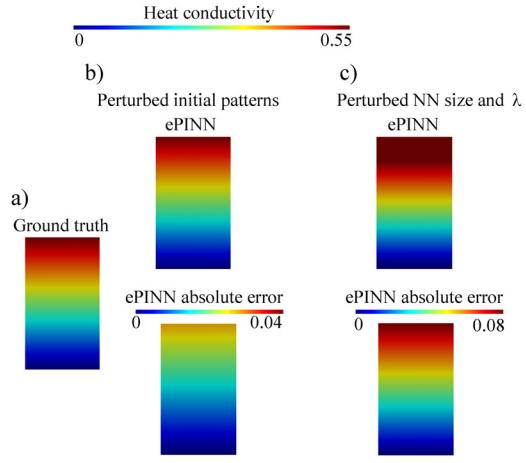


**FIG. 7.** The effect of perturbing the initial patterns in the parallel sub-networks and neural network parameters on the ePINN results is shown for the 2D multiphysics heat transfer problem (test case 1). (a) The ground-truth heat conductivity pattern. (b) ePINN prediction and error with perturbed initial patterns, where all coefficients in Table II are increased by 10%. (c) ePINN prediction and error using different neural network structure sizes and hyperparameters, where the main network utilizes three hidden layers and 50 neurons per layer and $\lambda_b = 20$ and $\lambda_d = 15$.

## IV. DISCUSSION

In this study, we introduced a method for solving various inverse problems by combining an ensemble of parallel neural networks with physics-informed neural networks. The proposed ePINN approach involved initializing the unknown parameters of the system using several initial patterns and subsequently feeding these patterns into a PINN network to make the final prediction. The use of an ensemble of initial patterns was helpful in solving inverse problems involving heterogeneous parameters, where PINN alone may not be able to find realistic patterns due to the non-uniqueness of the problem. The initial pattern used in each sub-network could be any traditional function that provides a meaningful and relevant pattern for the problem at hand. The generated data by the arbitrary functions were approximated by the parallel deep neural networks using a low number of epochs (around 200) to speed up training and avoid overfitting by early stopping.

The performance of the proposed ePINN approach was evaluated in different fluid flow and transport problems. CFD simulations were carried out with a heterogeneous parameter to provide data for the inverse problems. The results showed that the ePINN approach guided convergence toward an appropriate solution, speeded up convergence, and outperformed the vanilla PINN approach. However, increasing the neural network size of the vanilla PINN approach resulted in a smaller loss, indicating that convergence was not an issue, but rather the non-uniqueness of inverse problems. Indeed, this was verified in the Appendix where we have shown the convergence of vanilla PINN to different heterogeneous patterns based on different hyperparameters and architectures. Although the total runtime for ePINN was longer than vanilla PINN as more training was required, however, the ePINN approach achieved a lower loss for the same architecture and also converged to the desired solution.

The vanilla PINN approach could be viewed as a hybrid deductive (physics-informed) and inductive (data-driven) approach for solving inverse problems, and our proposed ePINN framework builds on PINN by providing additional deductive reasoning through an ensemble of pre-trained neural networks. These pre-trained parallel networks represent our prior knowledge of the type of patterns that we expect to see in our unknown parameter, and when these networks are frozen in ePINN, they form a basis for the final solution, which is achieved by a nonlinear combination of these basis functions using the main network. In other words, we are using a deductive approach where we use a knowledge base that forms a foundation for further inferences and learning. We could further make an analogy with the long-term learning view behind transfer learning.[1] Namely, the parallel sub-networks in ePINN represent our hypotheses for the unknown pattern and in a broader sense could represent long-term and reusable models from a given knowledge base, while the main network and its training with PINN represents a short-term and data-driven model that is combined with our long-term knowledge base. While in this work we demonstrated that such an *a priori* knowledge base could improve inverse modeling with PINN, we did not specify a rigorous approach for the development of this knowledge base. Future research should further investigate how a long-term knowledge basis could be developed for specific problems to guide inverse modeling in heterogeneous domains. We suggest that a library of knowledge regarding the unknown parameter should be created based on existing experimental data and known constraints, and ePINN should sample from this library for its parallel sub-networks.

Ensemble methods in deep learning have been used for other applications. Deep ensembles have been used for uncertainty quantification in deep learning[22] and PINN.[30] In this setting, each member of the ensemble is trained independently on the same task but with different strategies, and the predictions are combined at the end to produce the final prediction. While our ePINN approach operates differently, the pretraining with the ensemble sub-networks could be viewed as a way to reduce uncertainty due to the lack of uniqueness in inverse problems. Random forests are one of the most popular ensemble learning methods in machine learning that use an ensemble of decision trees.[25] Ensemble methods have been used in the forward PINN modeling of unsteady systems for improving time-stepping over large time intervals.[31] An ensemble of parallel PINNs has been proposed for an asymptotic expansion solution of thin boundary layers with the perturbation theory.[32] In a somewhat similar approach to ensemble learning, a set of pre-trained neural networks trained on simple geometries with different boundary conditions have been assembled in an iterative fashion to solve forward problems on larger more challenging domains.[33]

Transfer learning has been recently used to improve inverse modeling with PINN by integrating interpolated and less accurate data in offline training together with transfer learning to improve convergence.[34] A similar approach was used by our group where low-fidelity CFD data solved for the same problem were used to pretrain PINN and improve PINN convergence with transfer learning for forward problems.[35]

Our study has some limitations, and our model could be improved. A Bayesian framework is a more natural approach for solving inverse problems and could be integrated with PINN[17,36] to provide an estimation of parameter distributions with inherent uncertainty quantification. We assumed the input data were clean but in practice, this could be noisy and corrupt data, which needs to be handled within PINN.[37] We did not provide a systematic approach for defining the patterns in the parallel sub-networks, which should be investigated in future work. More broadly, this is an open area of research for forward and inverse problems. For instance, while Fourier features have been shown to significantly improve PINN convergence in oscillatory problems, an incorrect selection of the Fourier feature frequencies moves the network away from the favorable initialization offered by Fourier features.[38] It is important to initialize the network in the vicinity of the desired solution to guide the optimization process and ultimately improve the solution. Moreover, our study focused on inverse modeling at the continuum scale. An intriguing avenue for future research is to extend our work to a multiscale inverse problem where one is interested in pore-scale properties.[39,40] Of course, the challenges associated with inverse modeling and uniqueness are expected to be exacerbated. Finally, the proposed approach was used for solving 2D steady-state problems with simple geometries and idealized measurement data. Future investigations are needed to check the efficiency of this method for 3D time-dependent problems with realistic (noisy) measurements.

## V. CONCLUSION

In conclusion, we have proposed an ensemble PINN approach to improve PINN performance in inverse problems in heterogeneous domains. An ensemble of pre-trained neural networks on different patterns has the potential to not only improve PINN accuracy in inverse problems but also guide PINN convergence to a more desirable solution pattern among different possible solutions.

## ACKNOWLEDGMENTS

We acknowledge the gracious support provided by NSF that supported this work (NSF Grant Nos. 2205265 and 2247173). We also acknowledge the kind support by the Mechanical Engineering Department at Northern Arizona University (NAU) and high-performance computing support provided by NAU's Advanced Research Computing team and the Monsoon cluster.

## AUTHOR DECLARATIONS

### Conflict of Interest

The authors have no conflicts to disclose.

### Author Contributions

**Maryam Aliakbari:** Conceptualization (equal); Data curation (lead); Formal analysis (lead); Methodology (lead); Software (lead); Writing – original draft (equal); Writing – review & editing (supporting). **Mohammadreza Soltany Sadrabadi:** Data curation (supporting); Formal analysis (supporting); Software (supporting); Writing – review & editing (supporting). **Peter Vadasz:** Conceptualization (supporting); Formal analysis (supporting); Investigation (supporting); Methodology (supporting); Supervision (supporting); Writing – original draft (supporting); Writing – review & editing (supporting). **Amirhossein Arzani:** Conceptualization (equal); Formal analysis (supporting); Methodology (supporting); Supervision (lead); Writing – original draft (equal); Writing – review & editing (lead).

## DATA AVAILABILITY

The Pytorch codes and data that support the findings of this study are openly available in Github at https://github.com/amir-cardiolab/ensemble-PINN-inverse, Ref. 41.

## APPENDIX A: ENSEMBLE SUB-NETWORK PATTERNS DEFINED FOR EACH PROBLEM

The initial pattern of each of the ten sub-networks is provided in Table II for different test cases. The initial patterns can be any arbitrary function if they provide a meaningful pattern and range relevant to the problem of interest. For test cases 1 and 2, the presented equations are used to generate ten different patterns for heat conductivity in the range of the domain. For test cases 3 and 4, patterns are generated to guide permeability and velocity predictions, respectively. Additionally, the data are rescaled between 0 and 0.5 for test case 1 and between 0 and 1 for other test cases. In test case 4, where the velocity field is being predicted the patterns are defined such that they satisfy the continuity condition. A purely data-driven deep neural network is used to learn a nonlinear mapping from the input coordinates to the generated data based on these equations. Transfer learning is utilized to initialize each sub-network in PINN with the learned data mapping, and at last, the output of each network is used as the input to the main network in ePINN to guide the final solution.

## APPENDIX B: THE EFFECT OF NEURAL NETWORK SETUP ON THE PREDICTED SOLUTION

We revisit test case 1 where fluid flow and convective heat transfer around a fin were coupled with heat conduction in a solid fin with heterogeneous conductivity. In an exploratory study, we demonstrate here that different strategies in the neural network setup in vanilla PINN can lead to different heat conductivity solutions. In Fig. 8, we show the identified conductivity pattern by PINN in the inverse problem together with the temperature distribution and compare it to the ground-truth results obtained by Fluent. Interestingly the temperature pattern found by PINN matches the ground truth very well; however, a different conductivity pattern is identified due to the non-uniqueness challenge.

In order to examine the difficulties associated with obtaining a unique solution, we conducted an analysis of various parameters that have the potential to impact the solution. These included factors such as the number of sparse measurements, neural network size, hyperparameters, initialization method, and activation function. One challenge that arose during training was that the neural network generated negative output values. To address this, we adjusted the activation functions toward positive values to produce positive output values. To determine the efficacy of this modified activation function, we assessed whether it should be applied to all layers of the neural network or solely to the final layer. As evidenced

**TABLE II.** Equations used for initialization of the ten parallel sub-networks in different test cases.

| Test case 1<br>$x \in [0, 0.28], y \in [0, 0.5]$ | Test case 2<br>$x \in [0, 0.28], y \in [0, 0.5]$ | Test case 3<br>$x \in [0, 1], y \in [0, 1]$ | Test case 4<br>$x \in [-1, 1], y \in [-0.15, 0.15]$ |
|---|---|---|---|
| $\exp(0.3x) + \exp(2y)$ | $\exp(0.3x) + 0.37\exp(2y) - 1$ | $\exp(x) + \exp(y)$ | $u = 25 - x^2 - y^2, v = 2xy - x^2$ |
| $\sin(2x) - \sin(10y)$ | $\exp(x) + y - 0.5$ | $\exp(x) + y$ | $u = \frac{x^2 - y^2}{2\pi}, v = -\frac{xy}{\pi}$ |
| $x + 0.5y - 0.84$ | $0.5y - x + 0.75$ | $x + 0.5y - 0.4$ | $u = 5\sin(20x), v = -100y\cos(20x)$ |
| $0.04\exp(x) + y$ | $\cos(20y)$ | $\exp(0.6y) - x$ | $u = y^{0.9}, v = (x-1)^4$ |
| $2x^2 - y^2$ | $\cos(9y) + \sin(10x) + \sin(8y)$ | $\sin(8y) + \cos(5x)$ | $u = x^{0.9}, v = -\left(\sin(x) + \cos\left(\frac{x}{10}\right) + 0.9yx^{-0.1}\right)$ |
| $\exp(0.6y) - x$ | $\cos(8x) + \sin(10y)$ | $\cos(7y) + \sin(x)$ | $u = (x-1)^4, v = -4y(x-1)^3 + x^{0.9}$ |
| $\sin(8x) + y$ | $\cos(8y) + \sin(10x)$ | $\cos(20y)$ | $u = (x-1)^2, v = -2y(x-1) + 0.1\pi\sin(2x)$ |
| $\sin(10y)$ | $\sin(8x) + y + 0.5$ | $\exp(-y)$ | $u = (x-1)^3, v = -3y(x-1)^2 + 0.1\pi\sin(2x)$ |
| $\cos(8y)$ | $\sin(100y)$ | $\sin(8x) + y$ | $u = y\sin(20y), v = (-x + 0.25)\sin(20x)$ |
| $\sin(3y) - \cos(4x)$ | $\cos(x) + \sin(x) + \cos(5y)$ | $\sin(10y)$ | $u = y\cos(20y), v = -(x + 0.25)\cos(20x)$ |

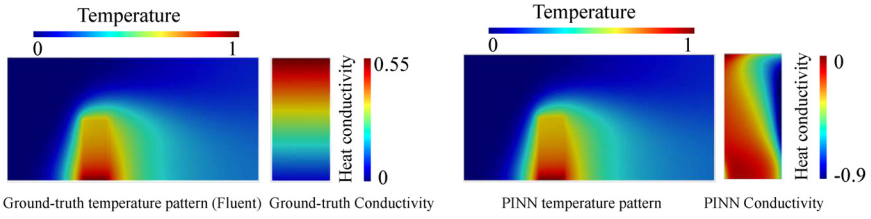


**FIG. 8.** Temperature and heat conductivity patterns solved in Fluent and PINN.

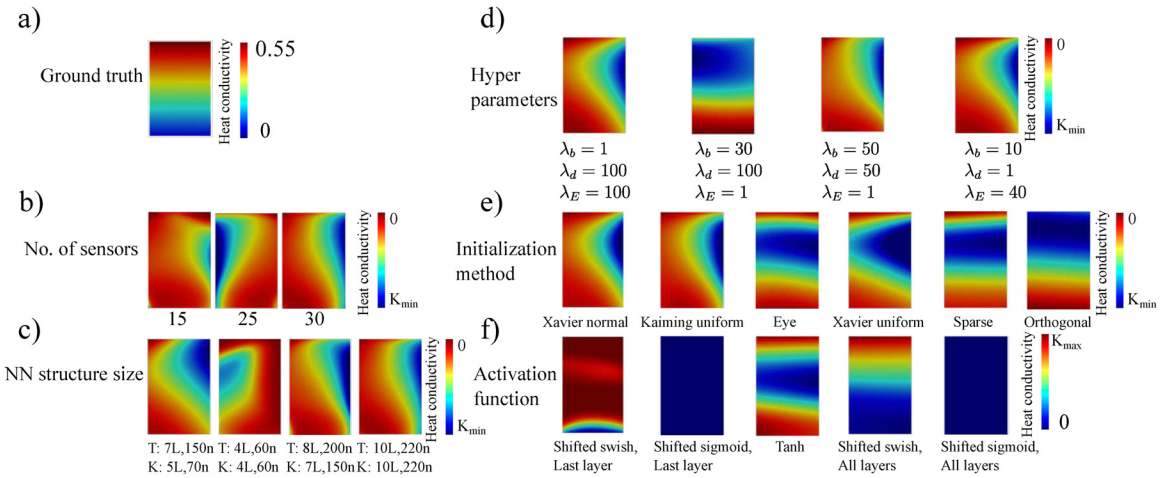


**FIG. 9.** Heat conductivity patterns found from PINN by considering the effect of different parameters (test case 1). $K_{min} < 0$ and $K_{max} > 0$ denote the heat conductivity generated as the output of the neural network and varies for each specific case. (a) The ground truth heat conductivity is shown. (b) The effect of the number of sensors is shown. (c) The effect of various neural network (NN) sizes are shown. The variables T and K correspond to neural networks that are utilized to predict temperature and heat conductivity, respectively. *L* and *n* refer to the number of layers and neurons employed in each neural network. (d) The effect of hyperparameters $\lambda_b$, $\lambda_d$, and $\lambda_E$ used to assign weights to the boundary condition, data, and equations loss, respectively, according to $\mathcal{L} = \lambda_E \mathcal{L}_F + \lambda_b \mathcal{L}_{BC} + \lambda_d \mathcal{L}_{data}$ is shown. (e) The effect of different initialization methods (explained in PyTorch documentation) is shown. (f) The effect of activation functions on the results. Shifted activation functions are used to produce positive output values. This is applied to all layers of the neural network or solely to the final layer.

by the conductivity results depicted in Fig. 9, none of the results perfectly align with the ground truth, underscoring the lack of a unique solution to the inverse problem.

## REFERENCES


[1]C. C. Aggarwal, *Artificial Intelligence: A Textbook* (Springer Nature, 2021).
[2]J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating physics-based modeling with machine learning: A survey," arXiv:2003.04919 (2020).
[3]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
[4]G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," Nat. Rev. Phys. **3**(6), 422–440 (2021).
[5]D. Givoli, "A tutorial on the adjoint method for inverse problems," Comput. Methods Appl. Mech. Eng. **380**, 113810 (2021).
[6]M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," Science **367**(6481), 1026–1030 (2020).
[7]A. Arzani, J. Wang, and R. M. D'Souza, "Uncovering near-wall blood flow from sparse data with physics-informed neural networks," Phys. Fluids **33**(7), 071905 (2021).
[8]S. L. Brunton and J. N. Kutz, "Methods for data-driven multiscale model discovery for materials," J. Phys.: Mater. **2**(4), 044002 (2019).
[9]S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks for heat transfer problems," J. Heat Transfer **143**, 060801 (2021).
[10]R. Laubscher, "Simulation of multi-species flow and heat transfer using physics-informed neural networks," Phys. Fluids **33**(8), 087101 (2021).
[11]S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," Acta Mech. Sin. **37**, 1727–1738 (2021).
[12]H. Eivazi, M. Tahani, P. Schlatter, and R. Vinuesa, "Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations," Phys. Fluids **34**(7), 075117 (2022).
[13]H. Wang, Y. Liu, and S. Wang, "Dense velocity reconstruction from particle image velocimetry/particle tracking velocimetry using a physics-informed neural network," Phys. Fluids **34**(1), 017116 (2022).
[14]E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, "A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics," Comput. Methods Appl. Mech. Eng. **379**, 113741 (2021).
[15]B. Yan, D. R. Harp, B. Chen, and R. Pawar, "A physics-constrained deep learning model for simulating multiphase flow in 3D heterogeneous porous media," Fuel **313**, 122693 (2022).
[16]H. Zuo, Z. Yang, S. Deng, and H. Li, "High-order asymptotic solutions for gas transport in heterogeneous media with multiple spatial scales," Phys. Fluids **35**(1), 013106 (2023).
[17]Y. Li, Y. Wang, and L. Yan, "Surrogate modeling for Bayesian inverse problems based on physics-informed neural networks," J. Comput. Phys. **475**, 111841 (2023).
[18]C. C. Aggarwal, *Linear Algebra and Optimization for Machine Learning* (Springer, 2020), Vol. 156.
[19]N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis *et al.*, "Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence," Technical Report No. 1478744, 2019.
[20]A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, "Characterizing possible failure modes in physics-informed neural networks," Adv. Neural Inf. Process. Syst. **34**, 26548–26560 (2021).
[21]R. Mojgani, M. Balajewicz, and P. Hassanzadeh, "Kolmogorov n–width and Lagrangian physics-informed neural networks: A causality-conforming manifold for convection-dominated PDEs," Comput. Methods Appl. Mech. Eng. **404**, 115810 (2023).
[22]M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," Inf. Fusion **76**, 243–297 (2021).
[23]R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. E. Hinton, "Neural additive models: Interpretable machine learning with neural nets," Adv. Neural Inf. Process. Syst. **34**, 4699–4711 (2021).
[24]G. Seni and J. F. Elder, *Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions*, Synthesis Lectures Data Mining Knowledge Discovery (Springer Cham, 2010), Vol. 2, pp. 1–126.

[25]T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer, 2009), Vol. 2.
[26]A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," J. Comput. Phys. **404**, 109136 (2020).
[27]L. McClenny and U. Braga-Neto, "Self-adaptive physics-informed neural networks using a soft attention mechanism," arXiv:2009.04544 (2020).
[28]A. Arzani, "Coronary artery plaque growth: A two-way coupled shear stress–driven model," Int. J. Numer. Methods Biomed. Eng. **36**(1), e3293 (2020).
[29]V. Carvalho, D. Pinho, R. A. Lima, J. C. Teixeira, and S. Teixeira, "Blood flow modeling in coronary arteries: A review," Fluids **6**(2), 53 (2021).
[30]A. F. Psaros, X. Meng, Z. Zou, L. Guo, and G. E. Karniadakis, "Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons," J. Comput. Phys. **477**, 111902 (2023).
[31]K. Haitsiukevich and A. Ilin, "Improved training of physics-informed neural networks with model ensembles," arXiv:2204.05108 (2022).
[32]A. Arzani, K. W. Cassel, and R. M. D'Souza, "Theory-guided physics-informed neural networks for boundary layer problems with singular perturbation," J. Comput. Phys. **473**, 111768 (2023).
[33]H. Wang, R. Planas, A. Chandramowlishwaran, and R. Bostanabad, "Mosaic flows: A transferable deep learning framework for solving PDEs on unseen domains," Comput. Methods Appl. Mech. Eng. **389**, 114424 (2022).
[34]C. Xu, B. T. Cao, Y. Yuan, and G. Meschke, "Transfer learning based physics-informed neural networks for solving inverse problems in tunneling," arXiv:2205.07731 (2022).
[35]M. Aliakbari, M. Mahmoudi, P. Vadasz, and A. Arzani, "Predicting high-fidelity multiphysics data from low-fidelity fluid flow and transport solvers using physics-informed neural networks," Int. J. Heat Fluid Flow **96**, 109002 (2022).
[36]L. Yang, X. Meng, and G. E. Karniadakis, "B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data," J. Comput. Phys. **425**, 109913 (2021).
[37]W. Peng, W. Yao, W. Zhou, X. Zhang, and W. Yao, "Robust regression with highly corrupted data via physics informed neural networks," arXiv:2210.10646 (2022).
[38]S. Wang, H. Wang, and P. Perdikaris, "On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks," Comput. Methods Appl. Mech. Eng. **384**, 113938 (2021).
[39]M. V.Farahani and M. M. Nezhad, "On the effect of flow regime and pore structure on the flow signatures in porous media," Phys. Fluids **34**(11), 115139 (2022).
[40]B. Vazic, B. E. Abali, and P. Newell, "Generalized thermo-mechanical framework for heterogeneous materials through asymptotic homogenization," Continuum Mech. Thermodyn. **35**(1), 159–181 (2023).
[41]M. Analiabkari (2023). "Ensemble-PINN-inverse," Github. https://github.com/amir-cardiolab/ensemble-PINN-inverse