entropy

Article

Physics-Informed Neural Networks for Solving Coupled Stokes–Darcy Equation

Ruilong Pu and Xinlong Feng *

College of Mathematics and System Sciences, Xinjiang University, Urumqi 830017, China
* Correspondence: fxlmath@xju.edu.cn; Tel.: +86-991-858-5505

Abstract: In this paper, a grid-free deep learning method based on a physics-informed neural network is proposed for solving coupled Stokes–Darcy equations with Bever–Joseph–Saffman interface conditions. This method has the advantage of avoiding grid generation and can greatly reduce the amount of computation when solving complex problems. Although original physical neural network algorithms have been used to solve many differential equations, we find that the direct use of physical neural networks to solve coupled Stokes–Darcy equations does not provide accurate solutions in some cases, such as rigid terms due to small parameters and interface discontinuity problems. In order to improve the approximation ability of a physics-informed neural network, we propose a loss-function-weighted function strategy, a parallel network structure strategy, and a local adaptive activation function strategy. In addition, the physical information neural network with an added strategy provides inspiration for solving other more complicated problems of multi-physical field coupling. Finally, the effectiveness of the proposed strategy is verified by numerical experiments.

Keywords: Stokes–Darcy equation; interface conditions; deep learning method; deep neural network; physics-informed neural network

check for updates

Citation: Pu, R.; Feng, X. Physics-Informed Neural Networks for Solving Coupled Stokes–Darcy Equation. *Entropy* **2022**, *24*, 1106. https://doi.org/10.3390/e24081106

Academic Editor: António Lopes

Received: 8 July 2022
Accepted: 4 August 2022
Published: 11 August 2022

1. Introduction

The coupled Stokes–Darcy equation studied in this paper has many applications in the physical and engineering sciences. For example, in reservoir modeling, to model heterogeneous porous media, the permeability field is often assumed to be a multiscale function with high contrast and discontinuous features. There are also model studies of the evolution of fibroblast shape and position under stress [1]. The model is based on the idea of continuum mechanics to describe the stress-induced phase transition, the cell body is modeled as a linear elastic matrix, and the cell body surface evolves according to a specific dynamic relationship. In this model, the stress tensor has discontinuities at the cell surface due to changes in the strain tensor due to cell contraction. The stiffness term in the Stokes–Darcy system due to small parameters and the discontinuity in the normal velocity due to the imbalance of the normal stress at the interface make our problem difficult to solve. Moreover, for complex area problems, curved interface problems, and high-dimensional problems, it will be difficult to mesh generation. Therefore, knowing how to design an accurate, efficient, and stable meshless numerical approximation algorithm has become the focus of literature research. Studies [2–7] are relevant here, and interested readers can read the research.

In recent years, deep learning methods have achieved unprecedented success in various application fields such as computer vision, speech recognition, natural language processing, audio recognition, social network filtering, and bioinformatics. In some cases, they are better than human experts [8,9]. Driven by these exciting developments, people began to make an in-depth study on how to use deep neural networks to solve partial differential equations [10–13]. In particular, Raissi et al. proposed physics-informed neural networks(PINNs) to help solve partial differential equations and data-driven discovery [14]. The results show that given certain initial conditions and boundary conditions, PINNs can solve some partial differential equations very well. Since then, the door to solve partial differential equations using deep neural networks has been opened, and some works [15–24] based on PINNs have been published one after another. For example, Even Lu Lu et al. expounded the difference between the traditional finite element method and the deep neural network in solving partial differential equations from the selection of basis functions, the solution process, the error source, and the error order in [25]. Even the famous Mishra [26] et al. and Jagtap [27,28] et al. made a theoretical analysis on the errors generated in the training process of PINNs, in their respective works. Mishra et al. conducted a generalization error analysis to solve a class of inverse problems of PINNs, and Jagtap et al. carried out an error analysis on PINNs approaching the Navier–Stokes equation and generalized an error of extended physics-informed neural networks (XPINNs).

This paper mainly wants to study a new method for the numerical approximation of a meshless deep neural network [29] to solve the problems we care about. Our main goal is to study strategies to improve the ability of deep neural network to solve the Stokes–Darcy model, and to improve the approximation ability of PINNs to solve the Stokes–Darcy model. We propose strategies to improve the accuracy and efficiency of deep neural networks in this paper and provide several numerical examples to demonstrate our approach. It should be noted that due to the randomness of the initial parameters when training the network, our numerical results will fluctuate within a certain range.

The structure of this paper is as follows: an introduction to the Stokes–Darcy fluid coupling problem and its mathematical model is given in Section 2. In Section 3, the related knowledge of the neural network and strategies to improve the approximation properties of PINNs are introduced. The accuracy and reliability of the proposed strategy are verified by numerical examples in Section 4. Concluding remarks and outlook are given in Section 5.

2. Problem Setup

In this part, we specifically introduce the mathematical model of the problem and the corresponding interface conditions. The Stokes–Darcy fluid coupling system is discussed in a given region Ω; Γ divides region Ω into Ω_s and Ω_d, representing the region of Stokes flow and Darcy flow, respectively. For simplicity, Γ_s and Γ_d represent the boundaries of Ω_s and Ω_d, except the interface. $\mathbf{n}$ and $\boldsymbol{\tau}$ are used to represent the external normal vector and tangent vector respectively. On interface Γ, $\mathbf{n}_s$ is used to represent the normal vector from region Stokes to region Darcy, $\mathbf{n}_d$ is used to represent the normal vector from region Darcy to region Stokes, and $\boldsymbol{\tau}$ is the tangent vector.

In order to better describe the Stokes–Darcy fluid-coupling system mathematically, first of all, the motion of fluids in Ω_s and Ω_d is described by the Stokes equation and Darcy law, respectively. We often need to distinguish between physical quantities in Ω_s and Ω_d, especially when they are at the interface Γ. Therefore, the relevant physical quantities in the Ω_s region are represented by the symbols with the subscript s, and the relevant physical quantities in the Ω_d region are represented by the symbols with the subscript d, as follows:

$$\mathbf{u}_s = \mathbf{u}|_{\Omega_s}, \mathbf{u}_d = \mathbf{u}|_{\Omega_d}, p_s = p|_{\Omega_s}, p_d = p|_{\Omega_d}.$$

So we can get the governing system [7,30,31] as follows:
Fluid region (Stokes equations)

$$\begin{aligned} -\nabla \cdot \mathbb{T}(\mathbf{u}_s, p_s) =& \mathbf{f}_s, \quad \text{in} \quad \Omega_s, \\ \nabla \cdot \mathbf{u}_s =& 0, \quad \text{in} \quad \Omega_s. \end{aligned} \tag{1}$$

where $\mathbf{u}_s$ is the fluid velocity, p_s is the kinematic pressure, $\mathbf{f}_s$ is the external force, $\nu > 0$ is the kinematic viscosity of the fluid, $\mathbb{T}(\mathbf{u}_s, p_s) = 2\nu\mathbb{D}(\mathbf{u}_s) - p_s\mathbb{I}$ is the stress tensor, and $\mathbb{D}(\mathbf{u}_s) = \frac{1}{2}(\nabla\mathbf{u}_s + \nabla\mathbf{u}_s^T)$ is the deformation tensor, and $\mathbb{I}$ is the unit vector.

Rock matrix (Darcy equations)

$$\begin{aligned} \nu\mathbb{K}^{-1}\mathbf{u}_d + \nabla p_d =& \mathbf{f}_d, \quad \text{in} \quad \Omega_d, \\ \nabla\cdot\mathbf{u}_d =& 0, \quad \text{in} \quad \Omega_d. \end{aligned} \tag{2}$$

The above equation is the Darcy equation describing fluid flow in the porous media region, see [2,30–32], $\mathbf{u}_d$ is the fluid velocity in the porous medium, p_d is the dynamic pressure, and $\mathbf{f}_d$ is the external force source term. The permeability $\mathbb{K}$ is a positive definite symmetric tensor allowed to vary in space.

Outer boundary

$$\begin{aligned} \mathbf{u}_s =& 0, \quad \text{on} \quad \Gamma_s, \\ \mathbf{u}_d\cdot\mathbf{n}_s =& 0, \quad \text{on} \quad \Gamma_d. \end{aligned} \tag{3}$$

Here, for simplicity, we consider the Dirichlet boundary conditions on the Stokes side and the Neumann boundary conditions on the Darcy side.

Obviously, the pressure is unique under an additional constant, so we can assume that

$$\int_\Omega p \, \mathrm{dxdy} = 0.$$

Interface conditions

$$\mathbf{u}_s\cdot\mathbf{n}_s + \mathbf{u}_d\cdot\mathbf{n}_d = 0, \quad \text{on} \quad \Gamma, \tag{4a}$$

$$2\nu\mathbf{n}_s\cdot D(\mathbf{u}_s)\cdot\mathbf{n}_s = p_s - p_d, \quad \text{on} \quad \Gamma, \tag{4b}$$

$$2\mathbf{n}_s\cdot D(\mathbf{u}_s)\cdot\boldsymbol{\tau} = -\alpha K^{-1/2}\mathbf{u}_s\cdot\boldsymbol{\tau}, \quad \text{on} \quad \Gamma. \tag{4c}$$

Here, the (4a) is a continuity condition of normal velocity at an interface obtained by conservation of mass, (4b) is a continuity condition of normal stress of fluid at an interface obtained through normal force balance, and (4c) is a famous Beaver–Joseph–Saffman (BJS) interface condition [30,33–35], where parameter α is a constant associated with friction.

3. Numerical Method

In this part, we will adopt the fully connected deep neural network (DNN) as our basic network to solve the problem. At the same time, the PINNs algorithm framework and some extensions of the algorithm are introduced.

3.1. Network Formation

DNN is a widely parallel connected network composed of multiple simple units. Its organizational structure can simulate the interactive response of a biological nervous system to real-world objects. From the perspective of computer science, the neural network can be regarded as a mathematical model with multiple parameters. This is the result of nested functions, such as $y_i = f_{act}(\sum_i W_i x_i + b_i)$. We connect each neuron of each layer together. Taking the neural network of the L-layer as an example, the output of the neural network is as follows:

$$\mathcal{U}(\mathbf{x},\theta) = W_{N_L-1} f_{act}(\cdots W_2 f_{act}(W_1(\mathbf{x}) + b_1) + b_2 \cdots) + b_{N_L-1}, \tag{5}$$

where W_i is the weighting coefficient matrix and b_i is the bias vector. All the undetermined parameters $\theta = \{W_i, b_i\}_{i=1,2,\cdots,N_L-1} \in \Theta$ in (5), and Θ is the parameter space. The (5) can also be written as

$$\mathcal{U}_\theta(\mathbf{z}) = (\mathcal{N}_D \circ \sigma \circ \mathcal{N}_{D-1} \circ \sigma \circ \mathcal{N}_{D-2} \circ \cdots \circ \sigma \circ \mathcal{N}_1)(\mathbf{z}). \tag{6}$$

Here, $\mathcal{N}_1 = \mathbf{W}_1\mathbf{z} + \mathbf{b}_1$, $\mathbf{z}$ is the input variable of neural network, σ stands the activation function, and D represents the number of layers of the neural network.

3.2. Physics-Informed Neural Networks

In [14], the authors propose to use deep neural networks to approximate the solution of partial differential equations, which can be called u-networks, and then use automatic differential techniques to obtain the differential operators of the equation. They then obtain the f-network satisfying the physical information of the equation. Then, the boundary function and internal loss function are established by using the principle of least squares. The working process of the PINNs is better explained below by taking the model we are asking for as an example.

When solving the Stokes–Darcy equation, we use the random Latin hypercube random point method to extract the data points and divide the data points into five parts according to the problem. After the input of the neural network is determined, we need to use the given boundary conditions and equation information to establish the loss function. Generally, the least square method is used, and the automatic differentiation technology [36] is also used in this process. Here, we divide the loss function into five parts: $\mathcal{L}(\mathbf{x}_{fs},\theta)$ represents the internal loss of the Stokes region; $\mathcal{L}(\mathbf{x}_{fd},\theta)$ represents the internal loss of the Darcy region; $\mathcal{L}(\mathbf{x}_{u\Gamma},\theta)$ represents the loss on the interface; and $\mathcal{L}(\mathbf{x}_{us},\theta)$ and $\mathcal{L}(\mathbf{x}_{ud},\theta)$ represent the loss on the boundary of the Stokes region and the Darcy region,respectively. Additionally, the specific expressions are as follows:

$$\mathcal{L}(\mathbf{x},\theta) = \mathcal{L}(\mathbf{x}_{fs},\theta) + \mathcal{L}(\mathbf{x}_{us},\theta) + \mathcal{L}(\mathbf{x}_{fd},\theta) + \mathcal{L}(\mathbf{x}_{ud},\theta) + \mathcal{L}(\mathbf{x}_{u\Gamma},\theta), \tag{7}$$

where,

$$\begin{aligned}
\mathcal{L}(\mathbf{x}_{fs},\theta) =& \frac{1}{N_{fs}}\sum_{i=1}^{i=N_{fs}} [|-2\nu\nabla\cdot D(\mathbf{u}_s(x^i_{f_s},y^i_{f_s})) + \nabla p_s(x^i_{f_s},y^i_{f_s}) - \mathbf{f}_s(x^i_{f_s},y^i_{f_s})|^2 + |\nabla\cdot\mathbf{u}_s(x^i_{f_s},y^i_{f_s})|^2],\\
\mathcal{L}(\mathbf{x}_{fd},\theta) =& \frac{1}{N_{fd}}\sum_{i=1}^{i=N_{fd}} [|\nu\mathbb{K}^{-1}\mathbf{u}_d(x^i_{f_d},y^i_{f_d}) + \nabla p_d(x^i_{f_d},y^i_{f_d}) - \mathbf{f}_d(x_i,y_i)|^2 + |\nabla\cdot\mathbf{u}_d(x^i_{f_d},y^i_{f_d}))|^2],\\
\mathcal{L}(\mathbf{x}_{u\Gamma},\theta) =& \frac{1}{N_{u_\Gamma}}\sum_{i=1}^{i=N_{u_\Gamma}} [|\mathbf{u}_s(x^i_{u_\Gamma},y^i_{u_\Gamma})\cdot\mathbf{n}_s + \mathbf{u}_d(x^i_{u_\Gamma},y^i_{u_\Gamma})\cdot\mathbf{n}_d|^2 + |2\nu\mathbf{n}_s\cdot D(\mathbf{u}_s(x^i_{u_\Gamma},y^i_{u_\Gamma}))\cdot\mathbf{n}_s\\
&- p_s(x^i_{u_\Gamma},y^i_{u_\Gamma}) + p_d(x^i_{u_\Gamma},y^i_{u_\Gamma})|^2 + |2\mathbf{n}_s\cdot D(\mathbf{u}_s(x^i_{u_\Gamma},y^i_{u_\Gamma}))\cdot\boldsymbol{\tau} + \alpha K^{-1/2}\mathbf{u}_s(x^i_{u_\Gamma},y^i_{u_\Gamma})\cdot\boldsymbol{\tau}|^2],\\
\mathcal{L}(\mathbf{x}_{us},\theta) =& \frac{1}{N_{us}}\sum_{i=1}^{i=N_{u_s}} |\mathbf{u}_s(x^i_{u_s},y^i_{u_s})|^2,\\
\mathcal{L}(\mathbf{x}_{ud},\theta) =& \frac{1}{N_{ud}}\sum_{i=1}^{i=N_{ud}} |\mathbf{u}_d(x^i_{u_d},y^i_{u_d})\cdot\mathbf{n}_s|^2.
\end{aligned}$$

Here, $\{x^i_{f_s},y^i_{f_s}\}_{i=1}^{N_{f_s}}$ represents the configuration points inside the Stokes region; $\{x^i_{f_d},y^i_{f_d}\}_{i=1}^{N_{f_d}}$ represents the configuration points inside the Darcy region; $\{x^i_{u_\Gamma},y^i_{u_\Gamma}\}_{i=1}^{N_{u_\Gamma}}$ represents the training data on the interface; and $\{x^i_{u_s},y^i_{u_s}\}_{i=1}^{N_{u_s}}$ and $\{x^i_{u_d},y^i_{u_d}\}_{i=1}^{N_{u_d}}$ represent the training data on the Ω_s and Ω_d boundaries, respectively. N_{fs}, N_{fd}, N_{u_Γ}, N_{us}, and N_{ud} represent the number of points in the Stokes region, the number of points in the Darcy region, the number of points in the interface, the number of points in the border of the Stokes region, and the number of points in the Darcy region. After establishing the loss function, we need to select the appropriate optimization algorithm to train the loss function and update the parameters in the neural network through training and back propagation. This process is repeated until the number of training sessions we set is reached or the loss function values converge. Then, we find an approximate solution of the partial differential equation. Common optimization algorithms include the stochastic gradient descent method, Newton method, and quasi-Newton method. This paper adopts the gradient based Adam algorithm [37], which has the advantages of adaptive learning rate and batch computing. In some calculation examples, the Adam algorithm is combined with the L-BFGS algorithm [38]. The working process of PINNs is given by Figure 1.

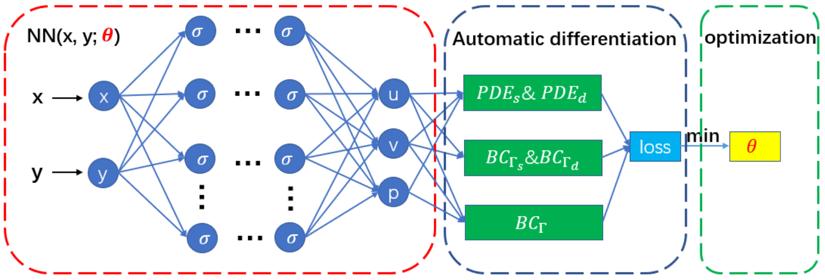


Figure 1. Physical-informed neural network structure diagram.

In Figure 1, x and y represent the input of the neural network; f_{act} in (5) and σ in the figure both represent the activation function in the neural network; and u, v, and p represent the output of the neural network.

3.3. Improving Strategy of Physical-Informed Neural Network

The PINNs has a strong approximation ability, can solve many physical problems, and can describe many physical phenomena, but it has certain limitations in solving small parameter problems, such as the fluid viscosity coefficient and permeability in the Stokes–Darcy system. If the viscosity coefficient of the problem to be solved is very small, it will increase the difficulty of solving. At the same time, there are some limitations in solving the interface discontinuity problem. In the Stokes–Darcy equation, if the analytical solution is discontinuous on the interface, the general PINNs cannot be well solved. Therefore, in order to solve the above limitations, we propose the following strategies.

3.3.1. Add a Weight Function to the Loss Function

One way to improve the accuracy of PINNs is to add a weight function before various losses of the loss function. For small-parameter problems, the weight function can be increased to balance all kinds of losses, so that the network will not focus on training one item and ignore other items. For the problem we are trying to solve, according to the specificity of our loss function, we only add the weight function to $\mathcal{L}(\mathbf{x}_{fs},\theta)$ and $\mathcal{L}(\mathbf{x}_{fd},\theta)$, that is, we replace (7) with the following

$$\mathcal{L}(\mathbf{x},\theta) = \varphi(\nu)\mathcal{L}(\mathbf{x}_{fs},\theta) + \mathcal{L}(\mathbf{x}_{us},\theta) + \psi(\nu,K)\mathcal{L}(\mathbf{x}_{fd},\theta) + \mathcal{L}(\mathbf{x}_{ud},\theta) + \mathcal{L}(\mathbf{x}_{u\Gamma},\theta), \quad (8)$$

where $\varphi(\nu)$ and $\psi(\nu,K)$ take the reciprocal of the corresponding parameters in the equation, that is, $\varphi(\nu) = \frac{1}{\nu}$, $\psi(\nu,K) = \frac{K}{\nu}$. Here, we do not use some adaptive weighting strategies [39,40] because the purpose of adaptive weighting strategies is to accelerate the convergence of the loss function. By adjusting the weights of various losses in the loss function, the value of the loss function decreases rapidly, but this has little effect on the small parameter problem we want to study.

3.3.2. Parallel Network Architecture

Another way to improve the approximation ability of PINNs is to decompose the solution region, that is, to divide the solution region into several sub-regions and use independent networks within each sub-region, which is the parallel network structure strategy. Common parallel network algorithms are conservative PINNs (cPINNs) [41] and XPINNs [16], both of which have been given parallel implementations in [42]. cPINNs are required to satisfy nonlinear conservation laws, and the interface condition part of the loss function is relatively complex, but XPINNs are suitable for solving all differential equations, and the interface condition part is also relatively simple. In the model to be solved, we use the idea of XPINNs to divide the solution region into two regions and train the neural network in the two sub-regions respectively. The specific training process is shown in Figure 2. The parallel network architecture has a very good effect on the solution of discontinuous problems on the interface, which will be shown in the numerical examples that follow.

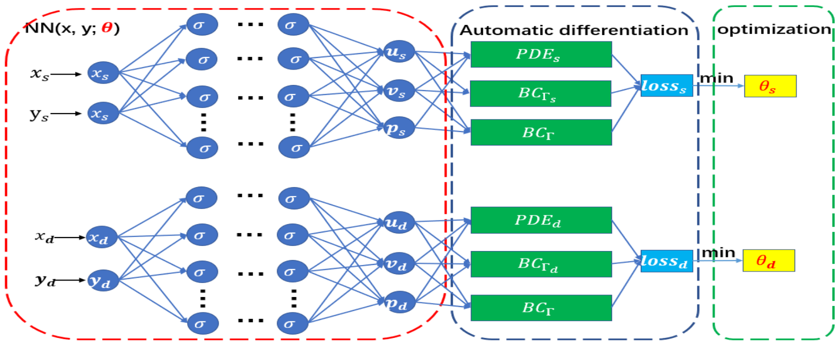


Figure 2. Parallel network architecture.

3.3.3. Local Adaptive Activation Function Strategy

The selection of activation function is very important for the training of neural networks. The use of a single activation function can no longer meet the needs of solving complex problems. Therefore, Jagtap et al. proposed Rowdy activation function [43] with good properties for solving partial differential equations with high-frequency composite components and proposed adaptive activation function. In [44], an additional scalable parameter na is introduced to the activation function, where $n \geq 1$ is a predefined scaling factor and parameter $a \in \mathbb{R}$ is the slope of the activation function. Since parameter a is defined for the whole network, we call this the global adaptive activation function (GAAF). The neural network expression of GAAF is shown by

$$\mathcal{U}_{\hat{\theta}}(\mathbf{z}) = (\mathcal{N}_D \circ \sigma \circ na\mathcal{N}_{D-1} \circ \sigma \circ na\mathcal{N}_{D-2} \circ \cdots \circ \sigma \circ na\mathcal{N}_1)(\mathbf{z}). \tag{9}$$

The optimization of these parameters will dynamically change the value of the loss function so as to accelerate the convergence of the loss function. But GAAF may fail on some complex issues. Therefore, a layer-wise locally defined activation function is proposed to extend this strategy, that is, add different slope a to the activation function of each hidden layer of the neural network. The neural network expression of the hierarchical local adaptive activation function is shown as follows:

$$\mathcal{U}_{\hat{\theta}}(\mathbf{z}) = (\mathcal{N}_D \circ \sigma \circ na^{D-1}\mathcal{N}_{D-1} \circ \sigma \circ na^{D-2}\mathcal{N}_{D-2} \circ \cdots \circ \sigma \circ na^1\mathcal{N}_1)(\mathbf{z}). \tag{10}$$

This provides additional D−1 parameters and optimizes the weight and bias, i.e., $\hat{\theta} = \{W_i, b_i, a_i\}_{i=1,2,\cdots,D-1} \in \hat{\Theta}$. Here, unlike the global adaptive activation function, each hidden layer has its own activation function slope.

4. Numerical Experiments

This section introduces several numerical experiments to solve the two-dimensional coupled Stokes–Darcy equation. Firstly, the accuracy and validity of the numerical method are verified by constructing numerical examples with analytical solutions, and the influence of weighted loss function on solving small parameter physical problems is demonstrated. Then, the analytical solution of interface discontinuity is constructed, and the numerical results of two different network structures are compared. Then, the more complicated interface curve problem is solved. Finally, a numerical example without analytical solution is designed to simulate the fluid movement under different permeabilities and viscosities, and the velocity flow diagram, in accordance with the physical law, is obtained.

In the following examples, we use the relative $\mathbb{L}_2$ norm to estimate our error by

$$E = \frac{\sqrt{\sum_{i=1}^{i=N} |U^i_{exact} - U^i_{pred}|^2}}{\sqrt{\sum_{i=1}^{i=N} |U^i_{exact}|^2}}. \tag{11}$$

Here, N represents the number of all points in the neural network training process, U_{pred} represents the predicted value at the corresponding coordinate point, and U_{exact} represents the analytical value at the corresponding coordinate point.

4.1. Interface Continuous Solution Problem

It is difficult to find a solution that meets the interface conditions (4b) and (4c). In this case, we simply extend the interface conditions to include an inhomogeneous term based on benchmark problem in [30,31]. In other words, we replace (4b) and (4c) with

$$2\nu \mathbf{n}_s \cdot D(\mathbf{u}_s) \cdot \mathbf{n}_s = p_s - p_d + g_1, \quad \text{on} \quad \Gamma, \tag{12a}$$

$$2\mathbf{n}_s \cdot D(\mathbf{u}_s) \cdot \boldsymbol{\tau} = -\alpha K^{-1/2}\mathbf{u}_s \cdot \boldsymbol{\tau} + g_2, \quad \text{on} \quad \Gamma. \tag{12b}$$

Then, we consider the coupled Stokes–Darcy equation in the region $\Omega = [0,1] \times [-1,1]$. The interface is $\Gamma = [0,1] \times \{0\}$, and set $\alpha = 1$; then, the analytical solution is given by

$$\begin{aligned} \mathbf{u}_s &= [-\sin(\pi x)^2 \sin(\pi y)\cos(\pi y), \quad \sin(\pi x)\cos(\pi x)\sin(\pi y)^2], \\ p_s &= \sin(\pi x)\cos(\pi y), \\ \mathbf{u}_d &= [-\sin(\pi x)^2 \sin(\pi y)\cos(\pi y), \quad \sin(\pi x)\cos(\pi x)\sin(\pi y)^2], \\ p_d &= \sin(\pi x)\cos(\pi y). \end{aligned} \tag{13}$$

Figure 3 shows the comparison between the analytical solution and the neural network prediction solution. Through (11), the relative $\mathbb{L}_2$ error of each physical quantity can be calculated as $E(\mathbf{u}_s)$ = 4.04 × 10^{-4}, $E(\mathbf{u}_d)$ = 1.46 × 10^{-3}, $E(p_s)$ = 3.48 × 10^{-3} and $E(p_d)$ = 4.22 × 10^{-5}, respectively. In the Table 1, we give the hyper-parameters in the neural network training process, where N_{us} and N_{ud} represent the number of points taken on the border of the Stokes region and Darcy region; $N_{fs} = N_{fd}$ and $N_{fs} = N_{fd}$ represent the number of internal points; $N_{u\Gamma}$ represents the number of interface points; L_N and N_N represent the number of neural network layers and the number of neurons in each hidden layer; and N_P represents the number of parameters of the neural network. The optimization algorithm, learning rate η, and activation function will continue to be used in the following examples unless otherwise specified. Next, in Tables 2 and 3, when we set the permeability as $\mathbb{K} = 10^{-2}\mathbb{I}$ and $10^{-4}\mathbb{I}$, respectively, and the fluid viscosity as $\nu = 10^{-1}$, 10^{-2}, 10^{-3} and 10^{-4} respectively, calculating the relative error of each physical quantity. The results show that the weighting of loss function is more helpful to calculate small parameter problems. At the same time, when permeability $\mathbb{K} = 10^{-2}\mathbb{I}$ and fluid viscosity $\nu = 10^{-3}$, the change of loss value and the change of relative $\mathbb{L}_2$ error of each physical quantity in the training process are shown in Figure 4. In the figure, each physical quantity with W in front represents the weighted result of the loss function, and the one without W in front represents the unweighted result of the loss function, which further shows that our measures are effective.

Table 1. Some hyper-parameters in the neural network training process.

$N_{us} = N_{ud}$	$N_{u\Gamma}$	$N_{fs} = N_{fd}$	L_N	N_N	*Opt Algorithm*	η	*Act Function*	N_P
500	125	15,000	5	100	Adam&L-BFGS	10^{-3}	$y = tanh(x)$	30,903

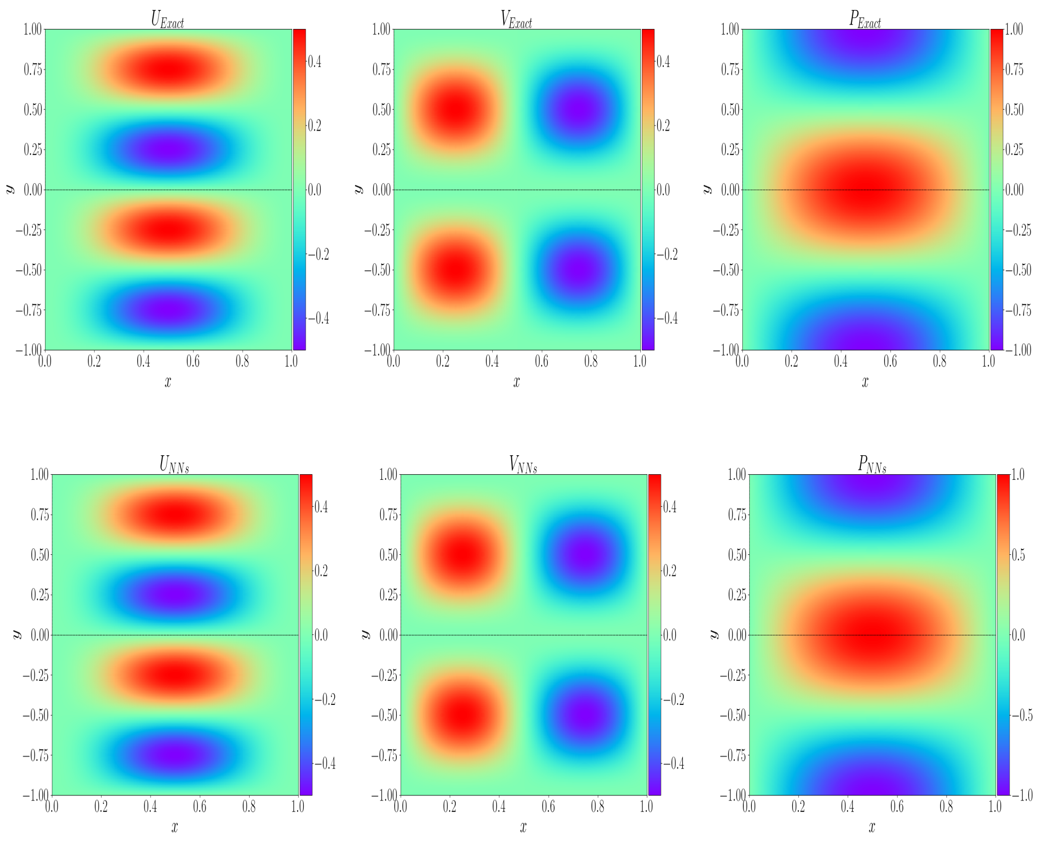


Figure 3. Top: Analytical solution of each solution variable. **Bottom**: The learning solution of each solution variable.

Table 2. When $\mathbb{K} = 10^{-2}\mathbb{I}$, the relative $\mathbb{L}_2$ error of velocity and pressure under different fluid viscosities.

	$\varphi(\nu) = 1, \psi(\nu, K) = 1$				$\varphi(\nu) = \frac{1}{\nu}, \psi(\nu, K) = \frac{K}{\nu}$			
ν	$E(\mathbf{u}_s)$	$E(\mathbf{u}_d)$	$E(p_s)$	$E(p_d)$	$E(\mathbf{u}_s)$	$E(\mathbf{u}_d)$	$E(p_s)$	$E(p_d)$
10^{-1}	7.31×10^{-4}	8.91×10^{-4}	1.12×10^{-3}	5.34×10^{-4}	3.60×10^{-4}	4.58×10^{-4}	7.32×10^{-4}	2.57×10^{-4}
10^{-2}	9.19×10^{-4}	2.99×10^{-3}	1.61×10^{-4}	9.07×10^{-5}	6.37×10^{-4}	1.98×10^{-3}	9.71×10^{-5}	8.43×10^{-5}
10^{-3}	6.35×10^{-2}	1.78×10^{-2}	2.37×10^{-4}	6.85×10^{-5}	2.39×10^{-3}	8.64×10^{-3}	3.26×10^{-5}	3.31×10^{-5}
10^{-4}	9.13×10^{-1}	2.62×10^{-1}	9.73×10^{-5}	3.32×10^{-5}	2.54×10^{-2}	5.52×10^{-2}	1.51×10^{-5}	1.68×10^{-5}

Table 3. When $\mathbb{K} = 10^{-4}\mathbb{I}$, the relative $\mathbb{L}_2$ error of velocity and pressure under different fluid viscosities.

	$\varphi(\nu) = 1, \psi(\nu, K) = 1$				$\varphi(\nu) = \frac{1}{\nu}, \psi(\nu, K) = \frac{K}{\nu}$			
ν	$E(\mathbf{u}_s)$	$E(\mathbf{u}_d)$	$E(p_s)$	$E(p_d)$	$E(\mathbf{u}_s)$	$E(\mathbf{u}_d)$	$E(p_s)$	$E(p_d)$
10^{-1}	9.48×10^{-2}	3.07×10^{-3}	6.32×10^{-2}	6.86×10^{-1}	1.05×10^{-2}	2.51×10^{-3}	1.35×10^{-2}	5.13×10^{-1}
10^{-2}	1.62×10^{-2}	5.41×10^{-3}	1.95×10^{-3}	6.97×10^{-2}	4.81×10^{-3}	2.90×10^{-3}	7.65×10^{-2}	3.81×10^{-2}
10^{-3}	8.15×10^{-1}	7.53×10^{-3}	1.54×10^{-3}	8.25×10^{-3}	5.31×10^{-3}	5.62×10^{-3}	6.53×10^{-5}	5.59×10^{-3}
10^{-4}	8.06×10^{-1}	5.24×10^{-2}	2.17×10^{-3}	2.23×10^{-3}	2.25×10^{-2}	1.89×10^{-2}	1.92×10^{-5}	1.00×10^{-3}

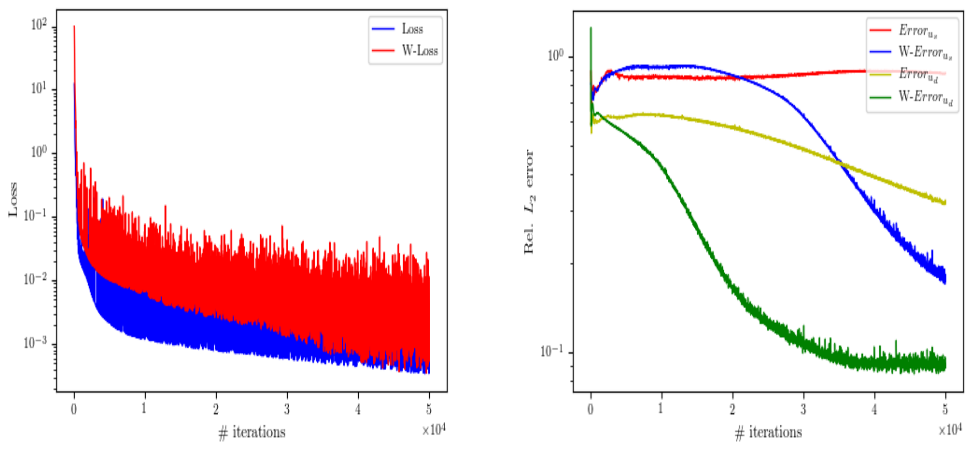


Figure 4. When permeability $\mathbb{K} = 10^{-2}\mathbb{I}$ and viscosity $\nu = 10^{-3}$, the change trend of weighted and unweighted loss function value (**left**) and the change trend of relative $\mathbb{L}_2$ error of velocity (**right**).

Next, we study the influence of the depth and width of the neural network on the prediction accuracy. In this study, we control other hyperparametric variables to remain unchanged. For different network depths and widths, the training times of Adam and L-BFGS algorithms are 10,000. As shown in Tables 4 and 5, we observe that the prediction accuracy of the model will increase with the increase of the width and depth of the neural network.

Table 4. The influence of neural network width on the prediction accuracy of each physical variable.

	$E(\mathbf{u}_s)$	$E(\mathbf{u}_d)$	$E(p_s)$	$E(p_d)$
[2] + 4 × [10] + [3]	2.71×10^{-2}	9.35×10^{-2}	2.52×10^{-1}	3.73×10^{-3}
[2] + 4 × [20] + [3]	6.88×10^{-3}	1.09×10^{-2}	3.58×10^{-2}	7.96×10^{-4}
[2] + 4 × [40] + [3]	3.23×10^{-3}	4.87×10^{-3}	2.01×10^{-2}	2.36×10^{-4}
[2] + 4 × [60] + [3]	1.21×10^{-3}	4.16×10^{-3}	1.58×10^{-2}	1.91×10^{-4}
[2] + 4 × [80] + [3]	1.00×10^{-3}	3.53×10^{-3}	1.03×10^{-2}	1.64×10^{-4}

Table 5. The influence of neural network depth on the prediction accuracy of each physical variable.

	$E(\mathbf{u}_s)$	$E(\mathbf{u}_d)$	$E(p_s)$	$E(p_d)$
[2] + 2 × [60] + [3]	3.50×10^{-3}	8.69×10^{-3}	1.23×10^{-2}	4.86×10^{-4}
[2] + 4 × [60] + [3]	1.45×10^{-3}	4.27×10^{-3}	2.19×10^{-2}	1.82×10^{-4}
[2] + 6 × [60] + [3]	1.46×10^{-3}	3.94×10^{-3}	2.16×10^{-2}	1.39×10^{-4}
[2] + 8 × [60] + [3]	1.09×10^{-3}	3.21×10^{-3}	1.06×10^{-2}	1.04×10^{-4}

4.2. Interface Discontinuity Solution Problem

In this example, we solve the Stokes–Darcy equation for discontinuous interfaces. Since the fluid is not continuous when passing through the interface, the solutions of the two regions will be very different, and it will be difficult to optimize, so it is difficult to simulate the fluid in the entire region with only one network. Therefore, we propose the parallel network architecture; one network is in the Stokes region, and the other network is in the Darcy region, and both networks play a role in the simulation at the interface. The solution region and parameter α are the same as in the previous example. The terms on the right-hand side of the equation and the inhomogeneous terms in the interface conditions are given by

$$\begin{aligned}
\mathbf{u}_s &= [-\sin(\pi x)^2 \sin(\pi y)\cos(\pi y), \quad \sin(\pi x)\cos(\pi x)\sin(\pi y)^2], \\
p_s &= \frac{1}{2}\sin(\pi x)\cos(\pi y), \\
\mathbf{u}_d &= [\frac{1}{2}\sin(2\pi x)\cos(2\pi y), \quad -\frac{1}{2}\cos(2\pi x)\sin(2\pi y)], \\
p_d &= \frac{1}{2}\sin(\pi x)\cos(\pi y).
\end{aligned} \tag{14}$$

Table 6 shows the comparison of the CPU-time used by the three algorithms to solve the model under the control of relevant variables, as well as the comparison of the relative $\mathbb{L}_2$ error of each physical variable. It can be observed that the parallel network architecture takes less time and has less error. Figure 5 shows the prediction results of velocity variables in the model by three algorithms, i.e., the single network structure, parallel network structure, and parallel network structure, with a local adaptive activation function strategy, as well as the absolute error comparison of the three algorithms. It can be observed from the absolute error diagram that the single network structure has a significant impact on the vicinity of the interface when solving the model. The simulation is not very good, but the parallel network architecture can be well simulated at the interface. It should be noted that the comparisons in Figure 5 and Table 6 are based on the same premise of controlling other training parameters.

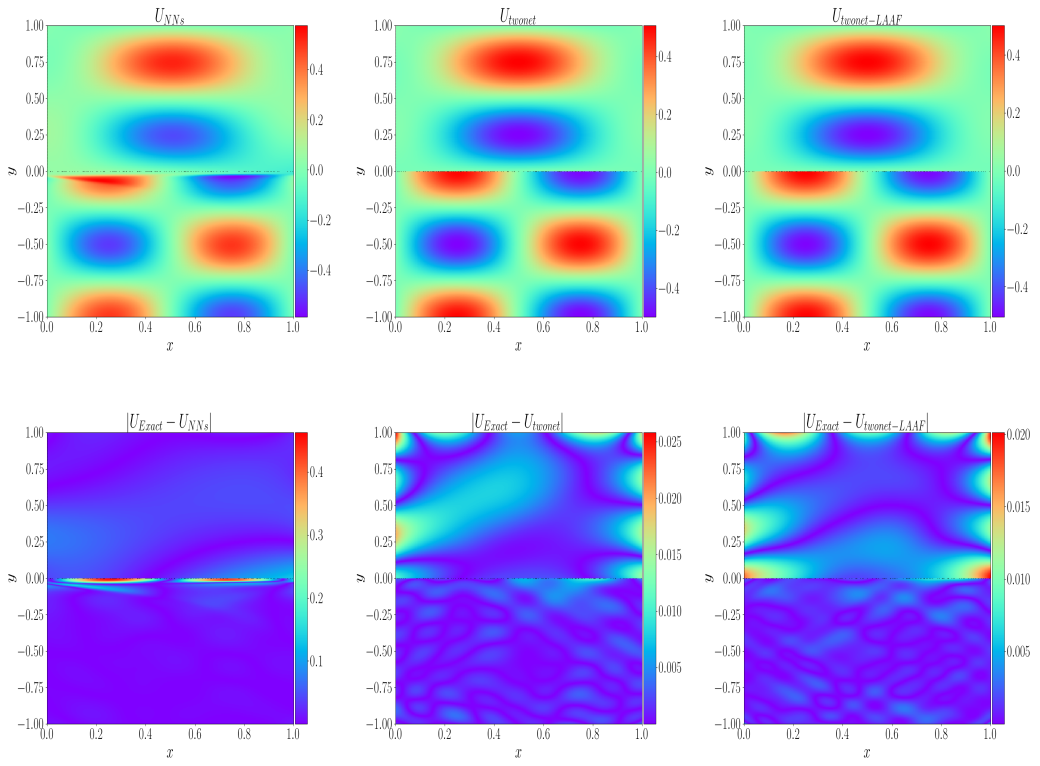


Figure 5. *Cont.*

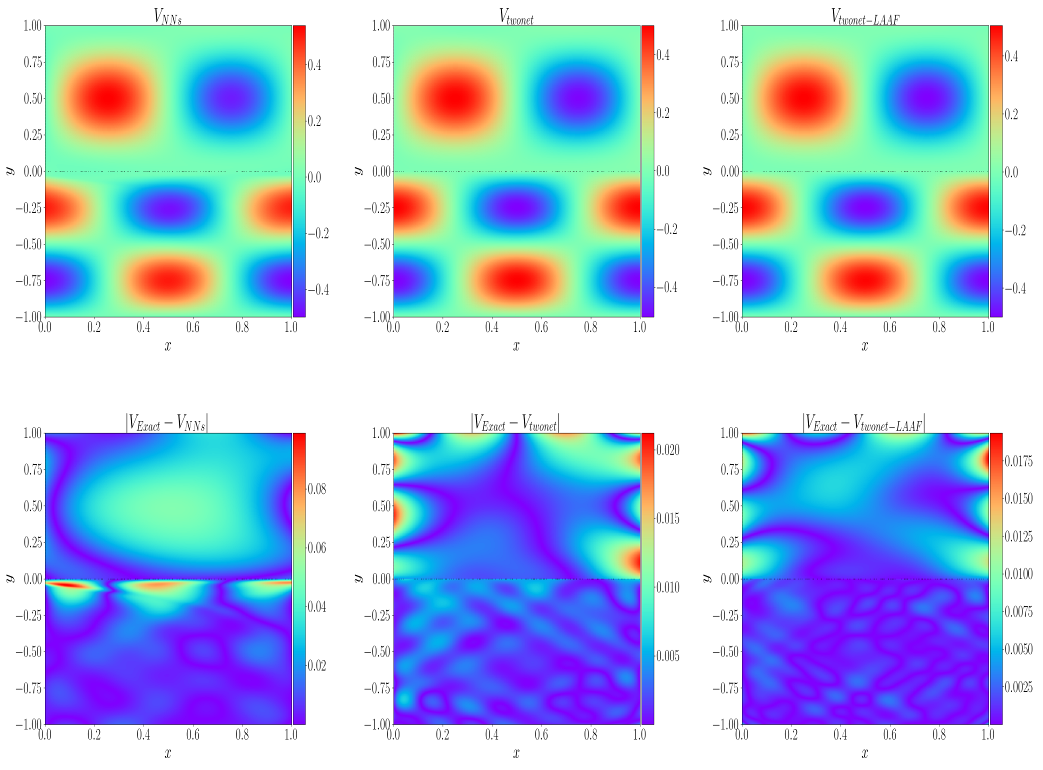


Figure 5. When the solution at the interface is discontinuous, the absolute errors of single network structure, parallel network structure, and local adaptive activation function parallel network structure are compared.

Table 6. The parameters of single network structure, parallel network structure, and parallel network structure using local adaptive activation function are compared.

	Single Network	**Parallel Network, a = 1**	**Variable a, (n = 20)**
network architecture	[2] + 4 × [100] + [3]	[2] + 4 × [70] + [3] (double)	[2] + 4 × [70] + [3] (double)
N_P	30,903	30,666	30,666
Training times	50,000	50,000	50,000
N	31,000	31,000	31,000
CPU-time(s)	11,482.7207	8482.6347	10,607.3143
$E(\mathbf{u}_s)$	2.25×10^{-1}	4.28×10^{-2}	1.48×10^{-2}
$E(\mathbf{u}_d)$	3.41×10^{-1}	1.05×10^{-2}	3.66×10^{-3}
$E(p_s)$	9.41×10^{-1}	1.75×10^{-1}	1.51×10^{-1}
$E(p_d)$	1.51×10^{-2}	2.37×10^{-3}	1.17×10^{-3}

4.3. Curved Interface Problem

In this example, we solve the curve interface problem, solve the region $\Omega = [0,1] \times [-1,1]$, interface $\Gamma : y = 0.0625\sin(4\pi x)$, and make the parameter $\alpha = 1$. Since the interface is a curve, the outer normal vector $\mathbf{n}$ and tangent vector $\boldsymbol{\tau}$ at each point on the interface are changed, so we make $\varphi(x,y) = 0.0625\sin(4\pi x) - y$; therefore, the outer normal vector at the interface is

$$\mathbf{n} = \frac{\nabla\varphi}{|\nabla\varphi|},$$

here, we get $\mathbf{n}_s = \frac{1}{|\nabla\varphi|}(\frac{d\varphi}{dx}, \frac{d\varphi}{dy})$, $\mathbf{n}_d = \frac{1}{|\nabla\varphi|}(-\frac{d\varphi}{dx}, -\frac{d\varphi}{dy})$.

The right end term of the equation and the non-homogeneous term in the interface condition are given by

$$
\begin{aligned}
\mathbf{u}_s &= [\frac{2}{\pi}\sin\pi y\cos\pi y\cos(x),\ \ (-2+\frac{1}{\pi^2}\sin(\pi y)^2)\sin(x)],\\
p_s &= (e^y - e^{-y})\sin(x),\\
\mathbf{u}_d &= [\frac{2}{\pi}\sin\pi y\cos\pi y\cos(x),\ \ (-2+\frac{1}{\pi^2}\sin(\pi y)^2)\sin(x)],\\
p_d &= (e^y - e^{-y})\sin(x).
\end{aligned}
\tag{15}
$$

Table 7 shows the hyper-parameters in the neural network training process. Figure 6 shows the comparison between the simulated fluid velocity and the fluid velocity given by the analytical solution and shows our calculation effect through the absolute error diagram. It can be seen that the error is relatively large only near the curve interface, and the simulation in other places is very good. Figure 7 represents the distribution of data points in each region used in the training process and the change of relative error of each physical quantity. Through (11), the relative $\mathbb{L}_2$ error of each physical quantity can be calculated as $E(\mathbf{u}_s)$=3.81 $\times$ 10^{-3} and $E(\mathbf{u}_d)$=3.74 $\times$ 10^{-3}, respectively.

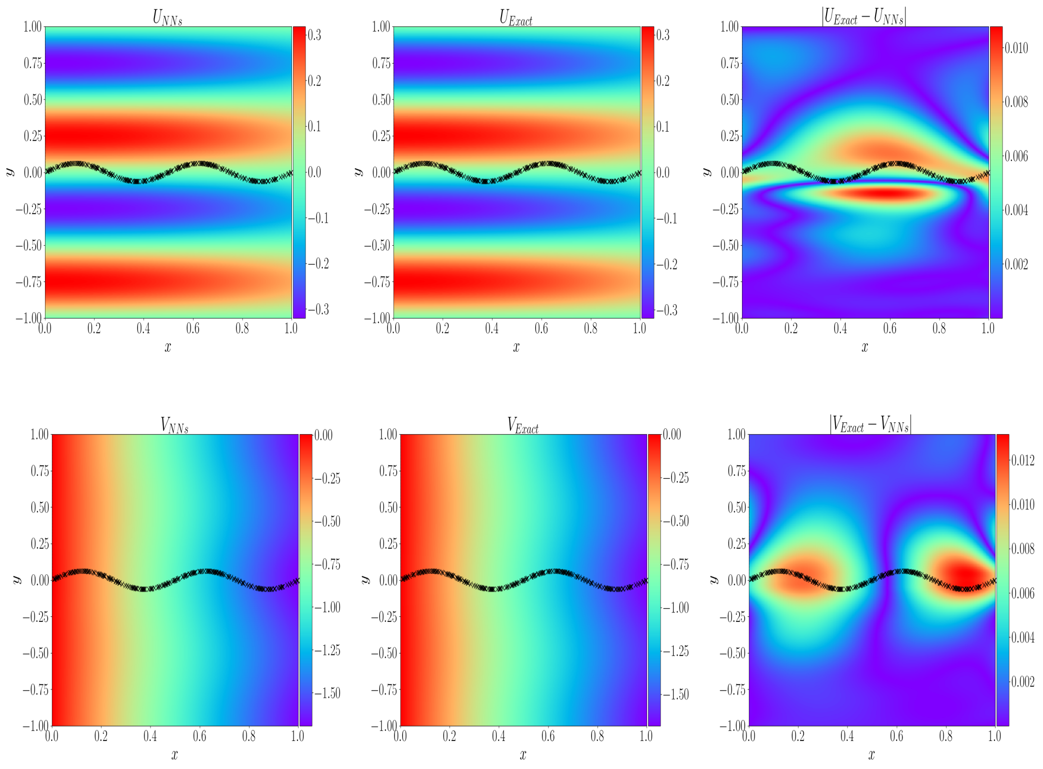


Figure 6. **Top**: Prediction solution, analytical solution, and absolute error of fluid velocity in x direction. **Bottom**: prediction solution, analytical solution, and absolute error of fluid velocity in y direction.

Table 7. Some hyper-parameters in the neural network training process.

$N_{us} = N_{ud}$	$N_{u\Gamma}$	$N_{fs} = N_{fd}$	L_N	N_N	N_P
500	200	15,000	5	100	30,903

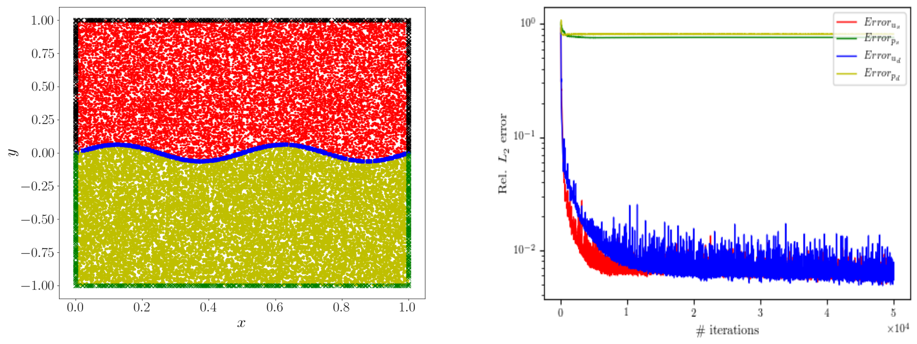
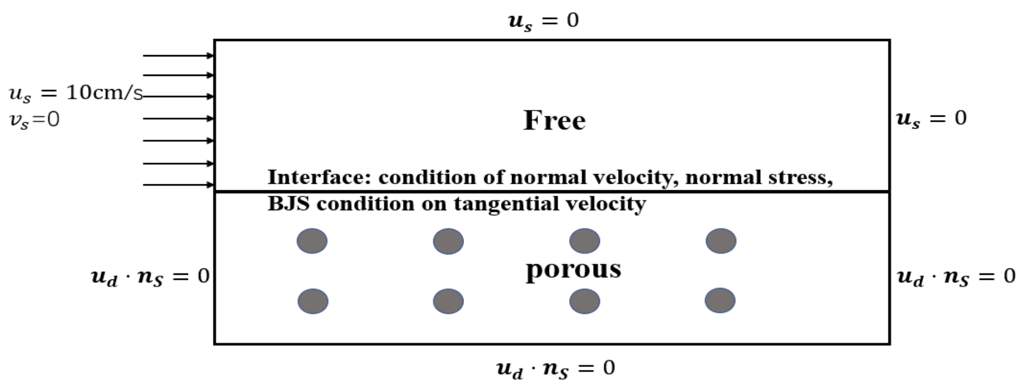


Figure 7. Left: the data points of each region are represented by different colors and symbols; **right**: the change of relative $\mathbb{L}_2$ error of each physical quantity during the solution process.

4.4. No-True Solution Problem

In this example, the physical phenomenon described by the Stokes–Darcy system is examined. Let $\mathbf{f}_s = 0$ and $\mathbf{f}_d = 0$ in (1) and (2), the fluid viscosity $\nu = 1$, and the solution region $\Omega = [0,1] \times [-1,1]$. The boundary conditions of the two regions are shown in Figure 8.



$\boldsymbol{u_s} = 0$
u_s = 10cm/s
v_s=0
Free
$\boldsymbol{u_s} = 0$
Interface: condition of normal velocity, normal stress,
BJS condition on tangential velocity
$\boldsymbol{u_d} \cdot \boldsymbol{n_S} = 0$
porous
$\boldsymbol{u_d} \cdot \boldsymbol{n_S} = 0$
$\boldsymbol{u_d} \cdot \boldsymbol{n_S} = 0$

Figure 8. Computational domain and boundary conditions of Stokes–Darcy coupling model.

Table 8 shows the hyper-parameters we used during training. We simulated different physical phenomena exhibited when a fixed permeability changes the viscosity of the fluid, as shown in Figure 9. From the simulation results, it can be seen that as the viscosity of the fluid decreases, the motion of the fluid becomes more intense, and the amount of fluid flowing through the interface and the speed will increase. At the same time, the physical phenomenon exhibited when the viscosity of the fixed fluid changes the permeability is simulated, as shown in Figure 10. From the simulation results, it can be observed that as the permeability decreases, the amount of fluid passing through the interface will decrease, and the velocity of the fluid that has passed through the interface will also decrease. The simulation effects shown in Figures 9 and 10 conform to certain physical laws.

Table 8. Hyper-parameters in neural network training.

$N_{us} = N_{ud}$	$N_{u\Gamma}$	$N_{fs} = N_{fd}$	L_N	N_N	N_P
375	125	15,000	5	100	30,903

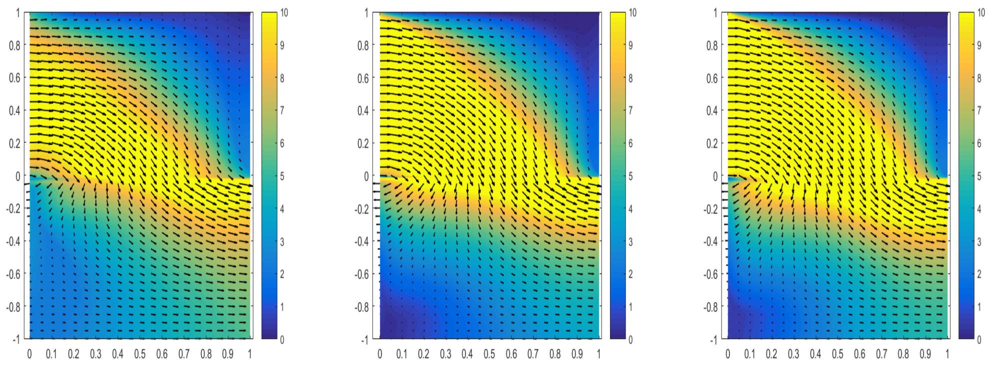


Figure 9. The figure shows that when the permeability of porous media is $\mathbb{K} = \mathbb{I}$, the change of fluid velocity is observed by changing fluid viscosity. **Left**: diagram of fluid velocity change when $\nu = 1$. **Middle**: diagram of flow velocity change when $\nu = 10^{-1}$. **Right**: diagram of flow velocity change when $\nu = 10^{-2}$.

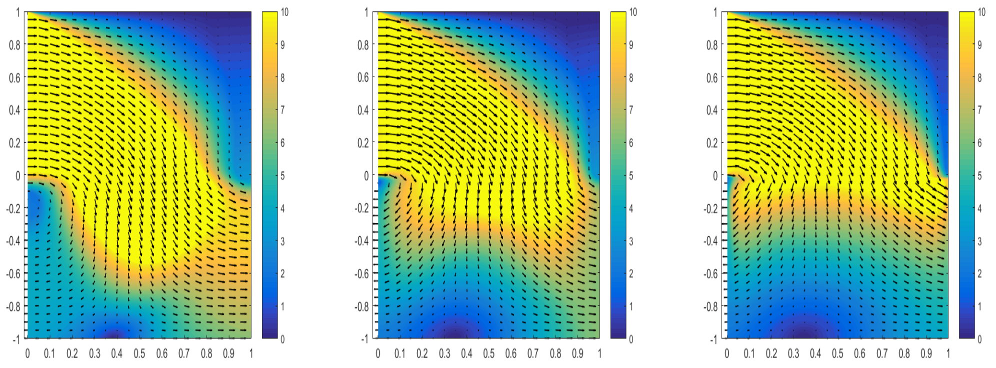


Figure 10. The figure shows that when the fluid viscosity is fixed at 10^{-4}, the permeability of the porous medium is changed to observe the change of fluid velocity. **Left**: Graph of fluid velocity change when porous media permeability $\mathbb{K} = \mathbb{I}$. **Middle**: the change of fluid velocity when the permeability of porous medium $\mathbb{K} = 10^{-2}\mathbb{I}$. **Right**: Variation diagram of fluid flow rate when porous media permeability $\mathbb{K} = 10^{-4}\mathbb{I}$.

5. Conclusions

In this paper, based on the PINN algorithm, we propose several strategies to improve the accuracy to solve the more complex Stokes–Darcy model, and the effectiveness of our proposed strategy is well verified in the example of small parameters and discontinuous interface. These strategies are not only applicable to the Stokes–Darcy system but also have a certain reference for the solution of other more complex multiphysics coupled models. However, in the process of solving the training network, we did not obtain the convergence speed of the algorithm, and the research on the minimum and saddle point problems in the optimization problem is also very meaningful.

Author Contributions: Conceptualization, X.F.; methodology, R.P.; software, R.P.; validation, X.F.; formal analysis, R.P.; investigation, R.P.; resources, X.F.; data curation, R.P.; writing—original draft preparation, R.P.; writing—review and editing, X.F. and R.P.; visualization, R.P.; supervision, X.F.; project administration, X.F.; and funding acquisition, X.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work received support by the Research Fund from Key Laboratory of Xinjiang Province (No. 2020D04002).

Acknowledgments: The authors would like to extend special thanks to Hui Xu for his guidance and assistance.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yarotsky, D. Error bounds for approximations with deep ReLU networks. *Neural Netw.* **2017**, *94*, 103–114. [CrossRef] [PubMed]
2. Chen, W.B.; Wang, F.; Wang, Y.Q. Weak Galerkin method for the coupled Darcy-Stokes flow. *IMA J. Numer. Anal.* **2016**, *36*, 897–921. [CrossRef]
3. Chen, W.B.; Gunzburger, M.; Hua, F.; Wang, X. A parallel robin-robin domain decomposition method for the Stokes–Darcy system. *IMA J. Numer. Anal.* **2011**, *49*, 1064–1084. [CrossRef]
4. Discacciati, M.; Quarteroni, A. Convergence analysis of a subdomain iterative method for the finite element approximation of the coupling of Stokes and Darcy equations. *Comput. Vis. Sci.* **2004**, *6*, 93–103. [CrossRef]
5. Discacciati, M.; Miglio, E.; Quarteroni, A. Mathematical and numerical models for coupling surface and groundwater flows. *Appl. Numer. Math.* **2002**, *43*, 57–74. [CrossRef]
6. Jiang, B. A parallel domain decomposition method for coupling of surface and groundwater flows. *Comput. Method Appl. Mech. Eng.* **2009**, *198*, 947–957. [CrossRef]
7. Kanschat, G.; Rivière, B. A strongly conservative finite element method for the coupling of Stokes and Darcy flow. *J. Comput. Phys.* **2010**, *229*, 5933–5943. [CrossRef]
8. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning;* MIT Press: Cambrigde, MA, USA, 2016.
9. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
10. Dockhorn, T. A discussion on solving partial differential equations using neural networks. *arXiv* **2022**, arXiv:1904.07200.
11. Berg, J.; Nyström, K. Data-driven discovery of PDEs in complex datasets. *J. Comput. Phys.* **2019**, *384*, 239–252. [CrossRef]
12. Sirignano, J.; Spiliopoulos, K. A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **2018**, *375*, 1339–1364. [CrossRef]
13. Yu, B. The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.* **2018**, *6*, 1–12.
14. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
15. Dwivedi, V.; Parashar, N.; Srinivasan, B. Distributed physics informed neural network for data-efficient solution to partial differential equations. *arXiv* **2019**, arXiv:1907.08967.
16. Jagtap, A.D.; Karniadakis, G.E. Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Commun. Comput. Phys.* **2020**, *28*, 2002–2041.
17. Jin, X.; Cai, S.; Li, H.; Karniadakis, G.E. NSFnets (Navier–Stokes Flow nets): Physics-informed neural networks for the incompressible Navier–Stokes equations. *J. Comput. Phys.* **2021**, *426*, 109951. [CrossRef]
18. Kharazmi, E.; Zhang, Z.; Karniadakis, G.E. Variational physics-informed neural networks for solving partial differential equations. *arXiv* **2019**, arXiv:1912.00873.
19. Meng, X.; Li, Z.; Zhang, D.; Karniadakis, G.E. PPINN: Parareal physics-informed neural network for time-dependent PDEs. *Comput. Method Appl. Mech. Eng.* **2020**, *370*, 113250. [CrossRef]
20. Shukla, K.; Jagtap, A.D.; Blackshire, J.L.; Sparkman, D.; Karniadakis, G.E. A Physics-Informed Neural Network for Quantifying the Microstructural Properties of Polycrystalline Nickel Using Ultrasound Data: A promising approach for solving inverse problems. *IEEE. Signal. Proc. Mag.* **2021**, *39*, 68–77. [CrossRef]
21. Jagtap, A.D.; Mitsotakis, D.; Karniadakis, G.E. Deep learning of inverse water waves problems using multi-fidelity data: Application to Serre–Green–Naghdi equations. *Ocean. Eng.* **2022**, *248*, 110775. [CrossRef]
22. Jagtap, A.D.; Mao, Z.; Adams, N.; Karniadakis, G.E. Physics-informed neural networks for inverse problems in supersonic flows. *J. Comput. Phys.* **2022**, *466*, 111402. [CrossRef]
23. Wang, S.; Yu, X.; Perdikaris, P. When and why pinns fail to train: A neural tangent kernel perspective. *J. Comput. Phys.* **2022**, *499*, 110768. [CrossRef]
24. Mao, Z.; Jagtap, A.D.; Karniadakis, G.E. Physics-informed neural networks for high-speed flows. *Comput. Method Appl. Mech. Eng.* **2020**, *360*, 112789. [CrossRef]
25. Lu, L.; Meng, X.; Mao, Z.; Karniadakis, G.E. DeepXDE: A deep learning library for solving differential equations. *SIAM Rev.* **2021**, *63*, 208–228. [CrossRef]
26. Mishra, S.; Molinaro, R. Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs. *IMA J. Numer. Anal.* **2022**, *42*, 981–1022. [CrossRef]
27. De Ryck, T.; Jagtap, A.D.; Mishra, S. Error estimates for physics informed neural networks approximating the Navier–Stokes equations. *arXiv* **2022**, arXiv:2203.09346.
28. Hu, Z.; Jagtap, A.D.; Karniadakis, G.E.; Kawaguchi, K. When do extended physics-informed neural networks (XPINNs) improve generalization? *arXiv* **2021**, arXiv:2109.09444.

29. Wang, Z.; Zhang, Z. A mesh-free method for interface problems using the deep learning approach. *J. Comput. Phys.* **2020**, *400*, 108963. [CrossRef]
30. Rui, H.; Zhang, R. A unified stabilized mixed finite element method for coupling Stokes and Darcy flows. *Comput. Method Appl. Mech. Eng.* **2009**, *198*, 2692–2699. [CrossRef]
31. Arbogast, T.; Brunson, D.S. A computational method for approximating a Darcy-Stokes system governing a vuggy porous medium. *Computat. Geosci.* **2007**, *11*, 207–218. [CrossRef]
32. Vassilev, D.; Yotov, I. Coupling Stokes–Darcy flow with transport. *SIAM J. Sci. Comput.* **2009**, *31*, 3661–3684. [CrossRef]
33. Jäger, W.; Mikelic, A. On the interface boundary condition of Beavers, Joseph, and Saffman. *SIAM J. Appl. Math.* **2000**, *60*, 1111–1127.
34. Beavers, G.S.; Joseph, D.D. Boundary conditions at a naturally permeable wall. *J. Fluid Mech.* **1967**, *30*, 197–207. [CrossRef]
35. Saffman, P.G. On the boundary condition at the surface of a porous medium. *Stud. Appl. Math.* **1971**, *50*, 93–101. [CrossRef]
36. Baydin, A.G.; Pearlmutter, B.A.; Radul, A.A.; Siskind, J.M. Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.* **2017**, *18*, 5595–5637.
37. Kingma, D.P.; Ba, J.L. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
38. Zhu, C.; Byrd, R.H.; Lu, P.; Nocedal, J. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM. Trans. Math. Softw.* **1997**, *23*, 550–560. [CrossRef]
39. Wang, S.; Teng, Y.; Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM J. Sci. Comput.* **2021**, *43*, A3055–A3081. [CrossRef]
40. McClenny, L.; Braga-Neto, U. Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism. *arXiv* **2020**, arXiv:2009.04544.
41. Jagtap, A.D.; Kharazmi, E.; Karniadakis, G.E. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Comput. Method Appl. Mech. Eng.* **2020**, *365*, 113028. [CrossRef]
42. Shukla, K.; Jagtap, A.D.; Karniadakis, G.E. Parallel physics-informed neural networks via domain decomposition. *J. Comput. Phys.* **2021**, *447*, 110683. [CrossRef]
43. Jagtap, A.D.; Shin, Y.; Kawaguchi, K. Deep Kronecker neural networks: A general framework for neural networks with adaptive activation functions. *Neurocomputing* **2022**, *468*, 165–180. [CrossRef]
44. Jagtap, A.D.; Kawaguchi, K.; Karniadakis, G.E. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *J. Sci. Comput.* **2020**, *404*, 109136. [CrossRef]