Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 33(8): 4964–4983.
DOI: 10.3934/era.2025223
Received: 19 June 2025
Revised: 01 August 2025
Accepted: 06 August 2025
Published: 25 August 2025

Research article

Subregion–identified physics-informed neural networks for the Navier–Stokes/Darcy interface model

Mulin Wang[1], **Jinyi Luo**[2], **Xiangpeng Xin**[1,*] **and Changxin Qiu**[3]

[1] School of Mathematical Sciences, Liaocheng University, Liaocheng 252000, China
[2] School of Mathematical Sciences, Chengdu University of Technology, Chengdu 610059, China
[3] School of Mathematics and Statistics, Ningbo University, Ningbo 315211, China

* **Correspondence:** Email: xinxiangpeng@lcu.edu.cn.

Abstract: Physics-informed neural networks (PINNs), built upon general neural networks, integrate physical information into the loss function, achieving remarkable results in solving partial differential equations (PDEs). Meanwhile, PINNs face challenges in solving complex interface problems, such as the Navier–Stokes/Darcy interface model considered in this paper. To address these challenges, we extend the general PINNs to propose subregion–identified physics-informed neural networks (SI-PINNs), which consist of three different neural networks for different subregions (Navier–Stokes, Darcy, and interface) with separated datasets. By identifying and classifying the regional features of the neural networks and datasets, the complexity of the physical information that SI-PINNs need to process is effectively split, while the networks also take all physical information into account during the training. Additionally, this splitting feature substantially reduces the training costs while enhancing or maintaining the accuracy of the general PINNs for the target model. Numerical experiments are performed to validate the effectiveness of the SI-PINN method.

Keywords: SI-PINNs; subregion–identification; physics-informed neural networks; Navier–Stokes/Darcy interface model; Beavers–Joseph interface condition

1. Introduction

The Stokes–Darcy model and the more advanced Navier–Stokes/Darcy model, which govern the motion of fluid in a physical system where free flow and porous media flow coexist, are widely applied in various fields such as petroleum extraction [1, 2], dual-porosity flow [3–5], and groundwater systems in karst aquifers [6, 7]. Due to the complexity of interface conditions and interactions between multiple subregions, various mesh-based numerical methods were developed for these models, including Lagrange multiplier methods [8, 9], domain decomposition methods [10–13],

multi-physics domain decomposition methods [14–16], finite element methods [17–21], ensemble algorithms [22], boundary integral methods [23, 24], two-grid methods [25], least squares methods [26] and many other methods [27–30].

In the past few decades, due to the advantages arising from the technological advancements and the strong nonlinear approximation capabilities of deep neural networks (DNNs), DNNs have become a research focus in various scientific fields, as seen in [31–35] and the references therein. Related methods generally require the design of solution frameworks tailored to specific problems [36–40]. Later, Raissi et al. [41] proposed physics-informed neural networks (PINNs) by incorporating the information contained in the physical equations as constraint terms into the loss function of the network.

In fact, it is challenging to directly apply PINNs to complex partial differential equations (PDEs) [42, 43]. Therefore, a large body of research has been conducted in this area [44–48]. Fu et al. [49] proposed the physics-informed kernel function neural networks (PIKFNNs) to solve various linear and some specific nonlinear PDEs. Jagtap et al. [50] proposed conservative PINNs (cPINNs) on discrete domains for nonlinear conservation laws. Built upon the cPINNs, Jagtap and Karniadakis [51] proposed extended PINNs (XPINNs), a generalized space-time domain decomposition approach for the PINNs to solve nonlinear PDEs on arbitrary complex-geometry domains. The readers are referred to [52–54] and the references therein for more details about the PINNs, coupled with domain decomposition. Although these studies have proposed new numerical methods to improve PINNs, more efficient and accurate solutions are still worth exploring.

In this article, we propose subregion–identified physics-informed neural networks (SI-PINNs), consisting of three different neural networks for three different subregions with separated datasets, for solving the Navier–Stokes/Darcy interface model. These three subregions include the Navier–Stokes region, the Darcy region, and the interface. Similar to XPINNs, SI-PINNs have multiple subregions and assign a separate neural network to each subregion, while the training process of SI-PINNs is based on the model's interface between different physics to identify the subregions. In particular, the interface itself serves as one subregion. By integrating the regional identification scheme, we identify and classify the neural networks and datasets according to their regional features, enabling SI-PINNs to require a smaller size and simpler composition for the datasets when addressing complex interface problems. Therefore, the complexity of the information that the SI-PINNs need to process is effectively split. Compared with the general PINNs, this splitting feature substantially reduces the training costs while enhancing the accuracy.

The subsequent structure of this paper is as follows. In Section 2, we introduce the Navier–Stokes/Darcy model with the Beavers–Joseph (BJ) interface condition [55]. In Section 3, we review the basic principles of PINNs and elaborate on the specific applications of SI-PINNs. In Section 4, we conduct numerical experiments to validate the effectiveness of SI-PINNs. Conclusions and prospects are presented in Section 5.

2. The Navier–Stokes/Darcy interface model

We consider the Navier–Stokes/Darcy interface model in a simplified region Ω, where the interface Γ divides Ω into a free flow region Ω_f and a porous media flow region Ω_p. The fluid motion in Ω_f and Ω_p is governed by the Navier–Stokes and Darcy equations, respectively. Γ_f and Γ_p are the boundaries

of Ω_f and Ω_p, respectively, excluding the interface. On the interface Γ, $\vec{n}$ is the normal vector from the free flow region to the porous medium flow region, and $\vec{\tau}$ denotes the tangent vector.

The fluid motion within Ω_f is described by the Navier–Stokes equations

$$\begin{cases} \rho\left(\dfrac{\partial \vec{u}_f}{\partial t} + (\vec{u}_f \cdot \nabla)\vec{u}_f\right) - \nabla \cdot \mathbf{T}(\vec{u}_f, p_f) = \rho \vec{g}_f & \text{in } \Omega_f \times T \\ \nabla \cdot \vec{u}_f = 0 & \text{in } \Omega_f \times T \end{cases}, \tag{2.1}$$

where $\vec{u}_f$ is the fluid velocity in Ω_f, p_f is the fluid's pressure, and $\vec{g}_f$ is the resulting external acceleration field acting on the fluid. $\mathbf{T}(\vec{u}_f, p_f) = 2\mu\mathbf{D}(\vec{u}_f) - p_f\mathbf{I}$ is the stress tensor. Here, $\mathbf{D}(\vec{u}_f) = \frac{1}{2}(\nabla\vec{u}_f + \nabla\vec{u}_f^T)$ is the deformation tensor, μ is the dynamic viscosity of the fluid, and $\mathbf{I}$ is the identity matrix.

The fluid motion within Ω_p is described by the Darcy equations

$$\begin{cases} S_0\dfrac{\partial \phi}{\partial t} + \nabla \cdot \vec{u}_p = f_p & \text{in } \Omega_p \times T \\ \vec{u}_p = -\mathbf{K}\nabla\phi & \text{in } \Omega_p \times T \end{cases}, \tag{2.2}$$

where $\vec{u}_p$ is the fluid velocity in Ω_p, f_p is the external force source term acting on the fluid in this region, and S_0 is the specific mass storativity coefficient. Moreover, $\phi = z + \frac{p_p}{g\rho}$ is the height of the liquid column, known as the pressure head, a term often related to fluid pressure in fluid mechanics experiments. Here, p_p is the dynamic pressure of the fluid, z is a relative depth referred to as the elevation head, g is the gravitational acceleration, and ρ is the fluid density. $\mathbf{K} = \frac{\rho g \kappa \mathbf{I}}{\mu}$ is the hydraulic conductivity, which governs the difficulty of fluid flow through porous media. Here, μ is the dynamic viscosity, κ is the permeability, and $\mathbf{I}$ is the identity matrix.

On Γ, the fluid's motion also needs to satisfy the following three interface conditions:

$$\begin{aligned} \vec{u}_f \cdot \vec{n} &= \vec{u}_p \cdot \vec{n} && \text{on } \Gamma, \\ -(\mathbf{T}(\vec{u}_f, p_f)\vec{n}) \cdot \vec{n} &= g\rho(\phi - z) && \text{on } \Gamma, \\ -(\mathbf{T}(\vec{u}_f, p_f)\vec{n}) \cdot \vec{\tau} &= \alpha\sqrt{\frac{g\rho\mu}{(\mathbf{K}\vec{\tau}) \cdot \vec{\tau}}}(\vec{u}_f - \vec{u}_p) \cdot \vec{\tau} && \text{on } \Gamma. \end{aligned} \tag{2.3}$$

The first equation describes the law of mass conservation. The second equation describes the balance of the normal forces. The third equation is the well-known Beavers–Joseph (BJ) interface condition [55–59], where the parameter α is the well-known BJ constant.

3. The SI-PINN structure

This section introduces the structure and procedure of SI-PINNs step by step. By identifying the regional features of the neural networks and datasets and conducting the PINNs' training, we achieve more efficient and accurate solutions to complex interface problems. In the rest of the section, we first review the general PINN framework and then elaborate on the SI-PINN structure, especially the subregion–identification and loss function.

3.1. General PINN structure

According to the universal approximation theorem of neural networks [60], the continuous differentiable function output by the neural network can accurately approximate any continuous function. Building upon DNNs, the physics-informed neural networks (PINNs) integrate physical information into the training process of neural networks, ultimately forming a machine learning framework that is better suited for numerical solutions to physical problems. We consider the general form of parameterized and nonlinear PDEs

$$u_t + \mathcal{N}[u;\lambda] = 0, x \in \Omega, t \in [0,T]. \tag{3.1}$$

Here, $u(x,t)$ is the potential solution of the PDE, $\mathcal{N}[\cdot;\lambda]$ is a nonlinear differential operator, θ is a subset of $\mathbb{R}^D$, and the left-hand side of Eq (3.1) is the governing equation f_{NN}, calculated as follows:

$$f_{NN} = (u_{NN}(x,t;\theta))_t + \mathcal{N}[u_{NN}(x,t;\theta);\lambda]. \tag{3.2}$$

Here, $u_{NN}(x,t;\theta)$ is the numerical solution of the PDE approximated by the neural network, where the time variable t and the spatial variable x serve as inputs to the PINN. The parameter θ is composed of the weight matrices and bias vectors in the neural network.

The task of PINNs is to continuously train the network using the existing data to ultimately obtain the optimal parameters θ^*, enabling the network to map spatio-temporal coordinates to a superior solution that closely approximates $u(x,t)$ as much as possible. PINNs incorporate known governing equations and boundary and initial conditions to compute the error of residuals for each component under a certain definition. Thus, the PINNs' training aims to find θ^* by optimizing the loss function $\mathcal{L}$, i.e.,

$$\theta^* \in \arg\min_{\theta} \mathcal{L}, \tag{3.3}$$

$$\mathcal{L} = \mathcal{L}_f + \mathcal{L}_b + \mathcal{L}_i, \tag{3.4}$$

where

$$\mathcal{L}_f = \frac{1}{N_f}\sum_{j=1}^{N_f}\left|f_{NN}\left(x_j^f, t_j^f\right)\right|^2, \tag{3.5}$$

$$\mathcal{L}_b = \frac{1}{N_b}\sum_{j=1}^{N_b}\left|u_{NN}\left(x_j^b, t_j^b\right) - u\left(x_j^b, t_j^b\right)\right|^2, \tag{3.6}$$

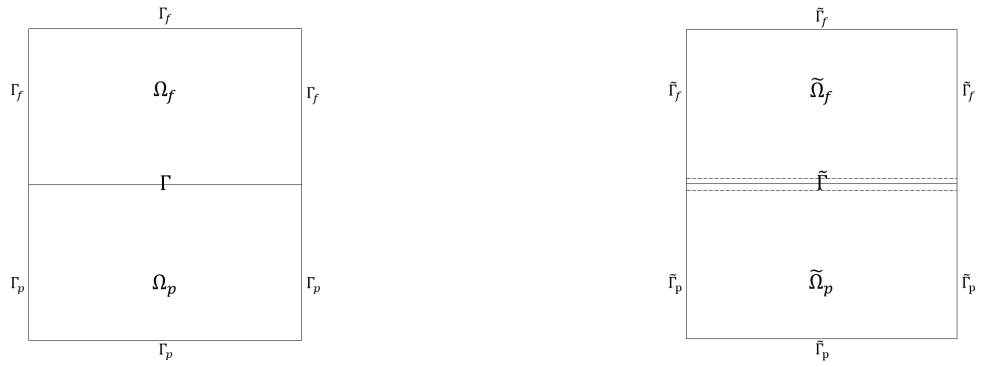
$$\mathcal{L}_i = \frac{1}{N_i}\sum_{j=1}^{N_i}\left|u_{NN}\left(x_j^{\mathrm{i}}, 0\right) - u\left(x_j^{\mathrm{i}}, 0\right)\right|^2. \tag{3.7}$$

Here, $\mathcal{L}_f$, $\mathcal{L}_b$, and $\mathcal{L}_i$ are the mean-squared error losses of the governing equations, boundary conditions, and initial conditions, respectively; $\left\{x_j^f, t_j^f\right\}_{j=1}^{N_f}$ represents the collocation points for f_{NN}; and $\left\{x_j^b, t_j^b\right\}_{j=1}^{N_b}$ and $\left\{x_j^i, t_j^i\right\}_{j=1}^{N_i}$ are the boundary and initial training data of $u(x,t)$, respectively. Through the use of optimizers such as Kingma and Adam [61], θ will be continuously optimized to minimize $\mathcal{L}$, thereby rendering $u_{NN}(x,t;\theta)$ close to $u(x,t)$. When $\mathcal{L}$ is less than a preset tolerance ϵ, we consider $u_{NN}(x,t;\theta)$ at this point to have satisfied the governing equations and boundary and initial conditions as much as possible.

3.2. Subregion–identification

When studying interface problems such as the one in Section 2, the loss function of the neural networks involves multiple subregions. Therefore, appropriate feeding of known regional information into the neural network is essential, which also affects how we identify and classify the neural networks and datasets for different subregions.

For the interface in this paper, we extend it to a narrow tubular neighborhood of the interface as a subregion. Let Γ be a smooth interface curve in R^2 . The tubular neighborhood of Γ with a half-width w is defined as the set $\tilde{\Gamma} = \left\{\mathbf{x} \in \mathbb{R}^2 \mid \text{dist}(\mathbf{x}, \Gamma) < w\right\}$, where $\text{dist}(\mathbf{x}, \Gamma) = \inf_{\mathbf{y}\in\Gamma} \|\mathbf{x} - \mathbf{y}\|$ denotes the shortest Euclidean distance from the point $\mathbf{x}$ to the curve Γ. The Navier–Stokes subregion is defined as $\tilde{\Omega}_f = \Omega_f/\tilde{\Gamma}$, with its boundary being $\tilde{\Gamma}_f = \Gamma_f/\tilde{\Gamma}$. Correspondingly, the Darcy subregion is defined as $\tilde{\Omega}_p = \Omega_p/\tilde{\Gamma}$, with its boundary being $\tilde{\Gamma}_p = \Gamma_p/\tilde{\Gamma}$. The subregion definition of SI-PINNs for the Navier–Stokes/Darcy interface model is shown in Figure 1.



(a) The original interface of the target model

(b) The definition of the subregions for SI-PINNs

Figure 1. The subregions' definitions.

For the general PINNs, we often feed datasets from all subregions into the neural network uniformly. For subregion–identified physics-informed neural networks (SI-PINNs), we first identify the three different subregions on the basis of the physical distribution of the interface problems, which will be used to identify the regional features corresponding to different datasets. Then, by integrating physical information and empirical knowledge, a masking strategy is employed to mark the identified dataset of each identified subregion, which is then delivered to SI-PINNs with smaller network sizes corresponding to different subregions. In the literature, different masking strategies are applicable to distinct problems and purposes. In accordance with the continuity of the mask, they can be broadly categorized into hard and soft masking strategies. The former, a method that completely retains or masks information from specific data during network training, denotes the process in which input data undergoes hard filtering via binarization operations and even multilevel quantization operations. The latter corresponds to the dynamic adjustment of data weights via continuous masking values, which provide partial information during the training.

Furthermore, on the basis of their own regional features and the distance relationship to Γ, the datasets provide SI-PINNs with different weights. Finally, after different SI-PINNs acquire the physical information from their datasets, they individually obtain θ corresponding to the different

regional features through the successive training. In general, through this approach, SI-PINNs effectively relax the requirements on the size and specific composition of the datasets, while splitting the complexity of the information that SI-PINNs need to process during the subsequent training.

The main steps of the subregion–identification of SI-PINNs are listed in **Procedure 1**.

Procedure 1 The regional identification of SI-PINNs

Step 1 Identify the three different subregions $\tilde{\Gamma}$, $\tilde{\Omega}_f$, and $\tilde{\Omega}_p$ according to the physical distribution of the interface problems.
Step 2 The masking strategy marks the identified dataset of each identified subregion.
Step 3 The dataset provides SI-PINNs with different weights corresponding to their own regional features and the distance relationship to Γ.
Step 4 Obtain network parameters θ corresponding to the different regional features.

3.3. Loss function of SI-PINNs

Built upon PINNs, SI-PINNs, consisting of three different neural networks for different subregions, integrate the general PINN framework with regional identification. By continuously optimizing θ, we ensure that $u_{NN}(x, t; \theta)$ ultimately adheres to all the governing equations and the corresponding boundary and initial conditions as closely as possible.

In the subsequent numerical examples, the loss function of SI-PINNs for the Navier–Stokes/Darcy model with the BJ interface condition is formulated as follows:

$$\mathcal{L} = \mathcal{L}_{ns} + \mathcal{L}_{d} + \mathcal{L}_{if}, \tag{3.8}$$

where

$$\begin{aligned}
\mathcal{L}_{ns} = \frac{1}{N_{f_{ns}}} \sum_{j=1}^{N_{f_{ns}}} & \left| \rho \left(\frac{\partial \vec{u}_f^{\mathcal{N}}}{\partial t} + (\vec{u}_f^{\mathcal{N}} \cdot \nabla) \vec{u}_f^{\mathcal{N}} \right) \right. \\
& \left. - \left(\nabla \cdot \mathbf{T}(\vec{u}_f^{\mathcal{N}}, p_f^{\mathcal{N}}) \right) \left(x_j^{f_{ns}}, t_j^{f_{ns}} \right) - \rho \vec{g} \left(x_j^{f_{ns}}, t_j^{f_{ns}} \right) \right|^2 \\
& + \frac{1}{N_{f_{ns}}} \sum_{j=1}^{N_{f_{ns}}} \left| (\nabla \cdot \vec{u}_f^{\mathcal{N}}) \left(x_j^{f_{ns}}, t_j^{f_{ns}} \right) - 0 \right|^2 \qquad \text{in } \Omega_p^c \times T, \\
& + \frac{1}{N_{b_{ns}}} \sum_{j=1}^{N_{b_{ns}}} \left| \vec{u}_f^{\mathcal{N}} \left(x_j^{b_{ns}}, t_j^{b_{ns}} \right) - \vec{u}_f \left(x_j^{b_{ns}}, t_j^{b_{ns}} \right) \right|^2 \\
& + \frac{1}{N_{i_{ns}}} \sum_{j=1}^{N_{i_{ns}}} \left| \vec{u}_f^{\mathcal{N}} \left(x_j^{i_{ns}}, t_j^{i_{ns}} \right) - \vec{u}_f \left(x_j^{i_{ns}}, t_j^{i_{ns}} \right) \right|^2
\end{aligned} \tag{3.9}$$

$$
\begin{aligned}
\mathcal{L}_d = & \frac{1}{N_{f_d}} \sum_{j=1}^{N_{f_d}} \left| S_0 \frac{\partial \phi^{\mathcal{N}}(x_j^{f_d}, t_j^{f_d})}{\partial t} + (\nabla \cdot \vec{u}_p^{\mathcal{N}})(x_j^{f_d}, t_j^{f_d}) - f_p(x_j^{f_d}, t_j^{f_d}) \right|^2 \\
& + \frac{1}{N_{f_d}} \sum_{j=1}^{N_{f_d}} \left| \vec{u}_p^{\mathcal{N}}(x_j^{f_d}, t_j^{f_d}) + (\mathbf{K}\nabla\phi^{\mathcal{N}})(x_j^{f_d}, t_j^{f_d}) \right|^2 \\
& + \frac{1}{N_{b_d}} \sum_{j=1}^{N_{b_d}} \left| \phi^{\mathcal{N}}(x_j^{b_d}, t_j^{b_d}) - \phi(x_j^{b_d}, t_j^{b_d}) \right|^2 \\
& + \frac{1}{N_{i_d}} \sum_{j=1}^{N_{i_d}} \left| \phi^{\mathcal{N}}(x_j^{i_d}, t_j^{i_d}) - \phi(x_j^{i_d}, t_j^{i_d}) \right|^2
\end{aligned} \qquad \text{in } \Omega_f^c \times T, \tag{3.10}
$$

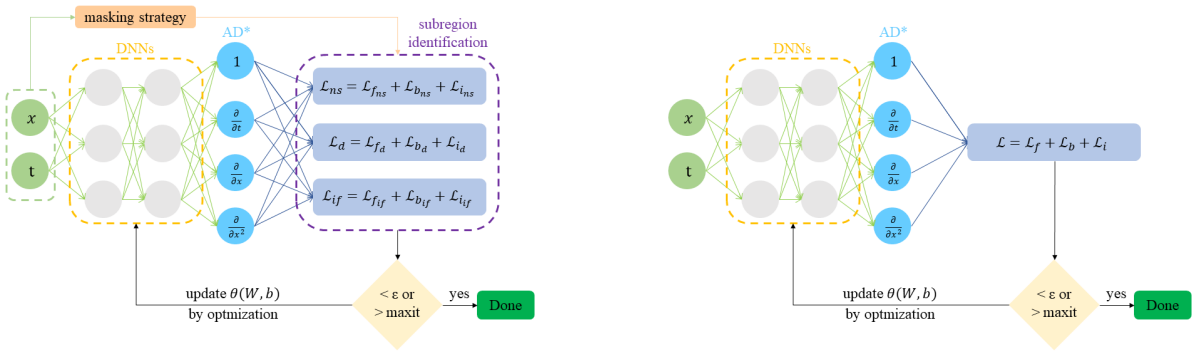
$$
\begin{aligned}
\mathcal{L}_{if} = & \frac{1}{N_{if}} \sum_{j=1}^{N_{if}} \left| (\vec{u}_f^{\mathcal{N}} \cdot \vec{n})(x_j^{if}, t_j^{if}) - (\vec{u}_p^{\mathcal{N}} \cdot \vec{n})(x_j^{if}, t_j^{if}) \right|^2 \\
& + \frac{1}{N_{if}} \sum_{j=1}^{N_{if}} \left| -\left((\mathbf{T}(\vec{u}_f^{\mathcal{N}}, p_f^{\mathcal{N}})\vec{n}) \cdot \vec{n} \right)(x_j^{if}, t_j^{if}) - g\rho(\phi^{\mathcal{N}}(x_j^{if}, t_j^{if}) - z) \right|^2 \\
& + \frac{1}{N_{if}} \sum_{j=1}^{N_{if}} \left| -\left((\mathbf{T}(\vec{u}_f^{\mathcal{N}}, p_f^{\mathcal{N}})\vec{n}) \cdot \vec{\tau} \right)(x_j^{if}, t_j^{if}) \right. \\
& \left. - \alpha \sqrt{\frac{g\rho\mu}{(\mathbf{K}\vec{\tau}) \cdot \vec{\tau}}} \left((\vec{u}_f^{\mathcal{N}} - \vec{u}_p^{\mathcal{N}}) \cdot \vec{\tau} \right)(x_j^{if}, t_j^{if}) \right|^2
\end{aligned} \qquad \text{in } \tilde{\Gamma} \times T, \tag{3.11}
$$

where $(\cdot)^{\mathcal{N}}$ is the numerical solution of the target model approximated by the neural network. Let $\left\{x_j^{f_{ns}}, t_j^{f_{ns}}\right\}$, $\left\{x_j^{b_{ns}}, t_j^{b_{ns}}\right\}$, and $\left\{x_j^{i_{ns}}, t_j^{i_{ns}}\right\}$ be the collocation points in $\tilde{\Omega}_p^c$, the training data on $\tilde{\Gamma}_p^c$, and the initial training data in $\tilde{\Omega}_p^c$, respectively. Let $\left\{x_j^{f_d}, t_j^{f_d}\right\}$, $\left\{x_j^{b_d}, t_j^{b_d}\right\}$, and $\left\{x_j^{i_d}, t_j^{i_d}\right\}$ be the collocation points in $\tilde{\Omega}_f^c$, the training data on $\tilde{\Gamma}_f^c$, and the initial training data in $\tilde{\Omega}_f^c$, respectively; $\left\{x_j^{if}, t_j^{if}\right\}$ represents the training data in $\tilde{\Gamma}$.

Similar to the general PINNs, by specifying parameters such as the network size and the different sampling points in different networks, setting the weights for each component of the loss function, and selecting the optimizer for the training, we ultimately train the network by continuously minimizing its loss to obtain $u_{NN}(x, t; \theta)$ close to $u(x, t)$. The differences in the frameworks between SI-PINNs and general PINNs can be clearly observed in Figure 2, and the complete training process of SI-PINNs can be found in **Procedure 2**.

Procedure 2 Training process of SI-PINNs

Step 1 Identify and classify the regional features of the training datasets per **Procedure 1**.
Step 2 Transfer the classified datasets into the subregion-specific PINN frameworks.
Step 3 Train the network to obtain the approximation $u_{NN}(x, t; \theta)$.
Step 4 Calculate the network loss $\mathcal{L}$ in the formulation (3.8).
Step 5 If $\mathcal{L} >$ the tolerance ϵ and the maximum number of iterations is not reached, update θ with the optimizer and return to **Step 3**.
Step 6 Obtain the predicted solution $u_{NN}(x, t; \theta)$.



(a) Framework of SI-PINNs

(b) Framework of general PINNs

Figure 2. Framework comparison between SI-PINNs and general PINNs. AD (marked with *) denotes automatic differentiation.

4. Numerical experiments

In this section, we conduct numerical experiments to validate the effectiveness of the proposed SI-PINN. The SI-PINN framework proposed in this paper is implemented using the DeepXDE Python library [62], a tool developed specifically for scientific machine learning and physics-informed learning. It supports diverse network designs, including PINNs and multifidelity neural networks (MFNNs), to solve various forward and inverse problems, such as partial differential equations (PDEs), integro-differential equations (IDEs), and fractional PDEs (fPDEs). Additionally, DeepXDE supports constructive solid geometry (CSG) technology, enabling the construction of complex geometric regions without generating meshes.

The swish activation function, a smooth approximation of the rectified linear unit (ReLU), is selected for the subsequent optimization [63–65], demonstrating better performance compared with ReLU and many of its improved variants on various tasks. Correspondingly, we selected He initialization as the network's initialization strategy. This is an improved version of the Xavier initialization strategy designed for ReLU-like functions [64, 66]. This improved version can address some of the challenges faced by these functions to a certain extent [67], thereby ensuring the smooth progress of network training. Additionally, the Adam optimizer is adopted to update θ on the basis of the training data [61]. By incorporating momentum information and adaptively adjusting the learning rates, the Adam optimizer dynamically tunes the learning rate for each parameter to achieve fast convergence. This generally makes it the preferred choice for problems involving large-scale datasets and nonconvex optimization.

To evaluate the accuracy of this method in $\tilde{\Gamma}$, we calculate the joint error, defined as follows:

$$\text{error}_u^{rel} = \frac{\sqrt{\left\|\vec{u}_f^{\mathcal{N}} - \vec{u}_f^*\right\|_{L^2_{\tilde{\Gamma}}}^2 + \left\|\vec{u}_p^{\mathcal{N}} - \vec{u}_p^*\right\|_{L^2_{\tilde{\Gamma}}}^2}}{\sqrt{\left\|\vec{u}_f^*\right\|_{L^2_{\tilde{\Gamma}}}^2 + \left\|\vec{u}_p^*\right\|_{L^2_{\tilde{\Gamma}}}^2}}, \tag{4.1}$$

$$\text{error}_p^{rel} = \frac{\sqrt{\left\|{p_f}^{\mathcal{N}} - {p_f}^*\right\|_{L^2_{\tilde{\Gamma}}}^2 + \left\|\phi^{\mathcal{N}} - \phi^*\right\|_{L^2_{\tilde{\Gamma}}}^2}}{\sqrt{\left\|{p_f}^*\right\|_{L^2_{\tilde{\Gamma}}}^2 + \|\phi^*\|_{L^2_{\tilde{\Gamma}}}^2}}. \tag{4.2}$$

Here, x^* denotes the exact solution in the corresponding region, and $x^{\mathcal{N}}$ denotes the approximation in the corresponding region, where $x = \vec{u_f}$, $\vec{u_p}$, p_f, and ϕ.

In the following numerical experiments, we will compare the proposed SI-PINNs with general PINNs for the Navier–Stokes/Darcy model. For SI-PINNs, a soft masking strategy based on the standard logistic function with an adjustment coefficient α_0 for network inputs is employed, enabling efficient tuning of the function's slope. This logistic-type function $\mathrm{S}(\mathbf{x})$ is defined as follows:

$$\mathrm{S}(\mathbf{x}) = \frac{1}{1 + e^{-\alpha_0 \,\mathrm{SD}(\mathbf{x})}}, \tag{4.3}$$

and the function $f(\mathbf{x})$ for the soft masking strategy used herein is defined as follows:

$$f(\mathbf{x}) = \begin{cases} \mathrm{S}(\mathbf{x}) & \text{in } \tilde{\Omega}_f \\ 4\,\mathrm{S}(\mathbf{x})(1 - \mathrm{S}(\mathbf{x})) & \text{in } \tilde{\Gamma} \\ \mathrm{S}(-\mathbf{x}) & \text{in } \tilde{\Omega}_p \end{cases}. \tag{4.4}$$

Here, $\mathrm{SD}(\mathbf{x}) = (\mathbf{x} - \mathbf{p}) \cdot \mathrm{n}(\mathbf{p})$ is the signed distance function in this paper (with Γ taken as a reference, the signed distance in Ω_f is positive, while that in Ω_p is negative), $\mathbf{p} \in \Pi_\Gamma(\mathbf{x})$ denotes the point on Γ closest to $\mathbf{x}$, and $\Pi_\Gamma(\mathbf{x}) = \{\mathbf{p} \in \Gamma \mid \|\mathbf{x} - \mathbf{p}\| = \mathrm{dist}(\mathbf{x}, \Gamma)\}$ denotes the set of all points on Γ closest to $\mathbf{x}$; $\mathrm{n}(\mathbf{p})$ is $\vec{n}$ at $\mathbf{p}$. Due to the strategy, the SI-PINNs will mostly accept the information of the data when the data match the regional features. Otherwise, the $f(\mathbf{x})$-based masking strategy will smoothly decrease the weights of the physical information provided by the datasets according to their own regional features and the distance to Γ.

The Navier–Stokes/Darcy model is considered in the domain Ω $[0, 0] \times [1, 2]$, where $[0, 1] \times [1, 2]$ is the Navier–Stokes region Ω_f and $[0, 0] \times [1, 1]$ is the Darcy region Ω_p. The interface Γ is defined as $[0, 1] \times \{1\}$, with the normal vector $\vec{n} = (0, -1)^T$ and the tangential vector $\vec{\tau} = (1, 0)^T$. The time domain for this target model is $T = [0, 2]$. The parameters are chosen as $z = 0$, $\rho = \mu = S_0 = g = k = \alpha = 1$, $\mathbf{K} = \frac{\rho g k \mathbf{I}}{\mu} = \mathbf{I}$, and $w = 1.0 \times 10^{-2}$. The source terms, boundary conditions, and initial conditions are set up such that the analytical solutions are given as follows:

$$\begin{cases} u^*_{f_{x_1}} = (x^2(y-1)^2 + y)\dfrac{\cos(t)}{3} & \text{in } \tilde{\Omega}_p^c \times T \\ u^*_{f_{x_2}} = \left(-\dfrac{2}{3}x(y-1)^3 + 2 - \pi\sin(\pi x)\right)\dfrac{\cos(t)}{3} & \text{in } \tilde{\Omega}_p^c \times T \\ p^*_f = (2 - \pi\sin(\pi x))\sin\left(\dfrac{\pi y}{2}\right)\dfrac{\cos(t)}{3} & \text{in } \tilde{\Omega}_p^c \times T \\ u^*_{p_{x_1}} = \pi^2(-y - \cos(\pi y) + 1)\dfrac{\cos(t)}{3}\cos(\pi x) & \text{in } \tilde{\Omega}_f^c \times T \\ u^*_{p_{x_2}} = (\pi\sin(\pi x) - 2)(\pi\sin(\pi y) - 1)\dfrac{\cos(t)}{3} & \text{in } \tilde{\Omega}_f^c \times T \\ \phi^* = (2 - \pi\sin(\pi x))(1 - y - \cos(\pi y))\dfrac{\cos(t)}{3} & \text{in } \tilde{\Omega}_f^c \times T \end{cases}. \tag{4.5}$$

In the general PINNs, the sampling points for training the PINNs are all sampled on the basis of the Latin hypercube sampling (LHS) method [68]. By first performing equal-probability stratification on the entire parameter space and then conducting random sampling, it can better cover each parameter subspace, ensuring more comprehensive information for the training process. Specifically, the number of sampling points in Ω is 9000, including those externally generated via LHS to address the difficulties in automatic sampling on Γ. These points are fed into general PINNs through DeepXDE's anchor interface. The number of boundary condition sampling points is 680 and the number of initial condition sampling points is 340. We set 14 hidden layers, each having 20 neurons. The swish function is selected as the activation function for all hidden layers to approximate $u(x, y, t)$. The weights of each component in the loss function are set to be equal by default. After 1.0×10^4 training epochs, Figures 3–6 show the exact and approximate solutions at $t = 2$, respectively.

In SI-PINNs, the number of sampling points in both $\tilde{\Omega}_f$ and $\tilde{\Omega}_p$ is 4000. For each subregion, the boundary conditions involve 300 sampling points and the initial conditions involve 150 sampling points. In $\tilde{\Gamma}$, the number of sampling points for the governing equations is 1000, the number of sampling points for the boundary conditions is 80, and the number of sampling points for the initial conditions is 40. All of these points are still sampled using the LHS method. We set four hidden layers with 20 neurons each for the individual subregions of $\tilde{\Omega}_f$ and $\tilde{\Omega}_p$, respectively. For $\tilde{\Gamma}$, we set six hidden layers with 20 neurons each. An excessively shallow or deep neural network may increase the errors, and the same applies to the width of the neural network. The selection of activation functions and the assignment of loss weights are kept consistent with the general PINNs. Additionally, we set the logistic function coefficient $a_0 = 2.5$. When α_0 changes significantly, it may also amplify errors. We then train SI-PINNs according to **Procedure 2**.

After 1.0×10^4 training epochs, Figures 7–10 show the exact and approximate solutions at $t = 2$. Compared with the general PINNs, the error of SI-PINNs in each subregion is lower under the same network size and dataset composition. Table 1 presents the time consumption and the joint relative error in $\tilde{\Gamma}$ corresponding to SI-PINNs and the general PINNs. Table 2 shows the L^2 norm errors of both of these PINNs in $\tilde{\Omega}_f$ and $\tilde{\Omega}_p$, respectively. All these tables illustrate that under the same number of training epochs, SI-PINNs exhibit shorter computational times while achieving a lower L^2 norm error. Our results show that SI-PINNs achieve higher accuracy in each subregion at a significantly lower training cost.

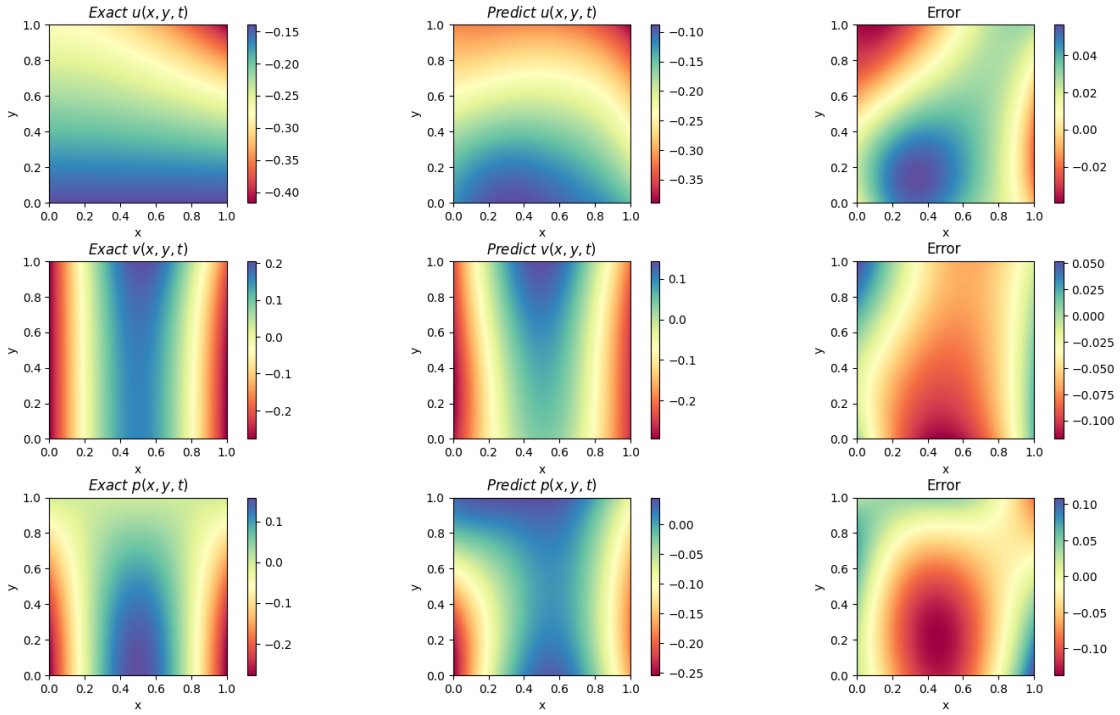


Figure 3. PINNs: Comparison of exact solutions and approximations for u, v, and p in $\tilde{\Omega}_f$ at $t = 2$, with their errors.

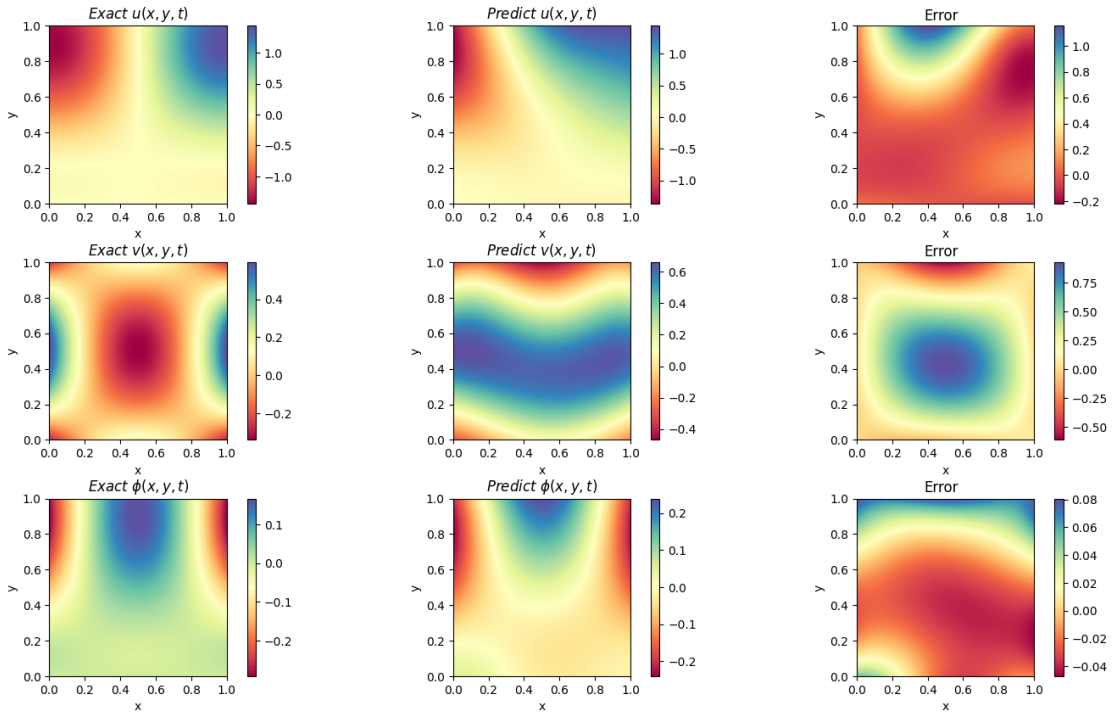


Figure 4. PINNs: Comparison of exact solutions and approximations for u, v, and ϕ in $\tilde{\Omega}_p$ at $t = 2$, with their errors.

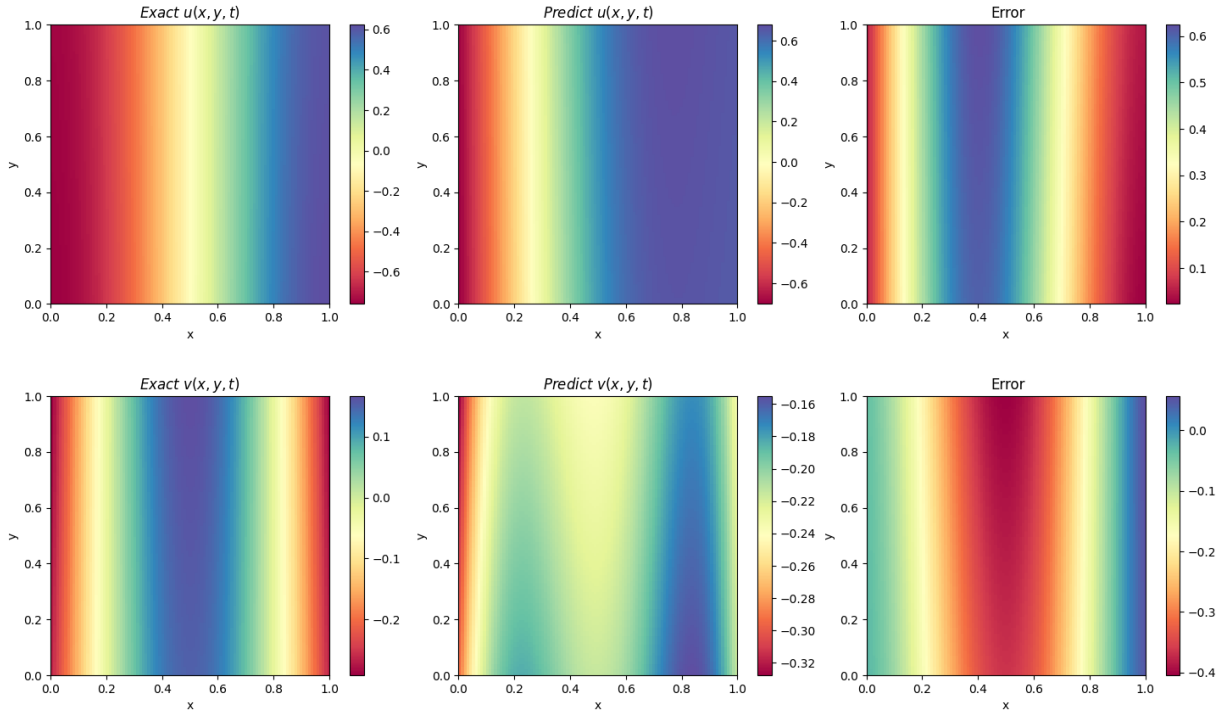


Figure 5. PINNs: Comparison of exact solutions and approximations for u and v in $\tilde{\Gamma}$ at $t = 2$, with their errors.

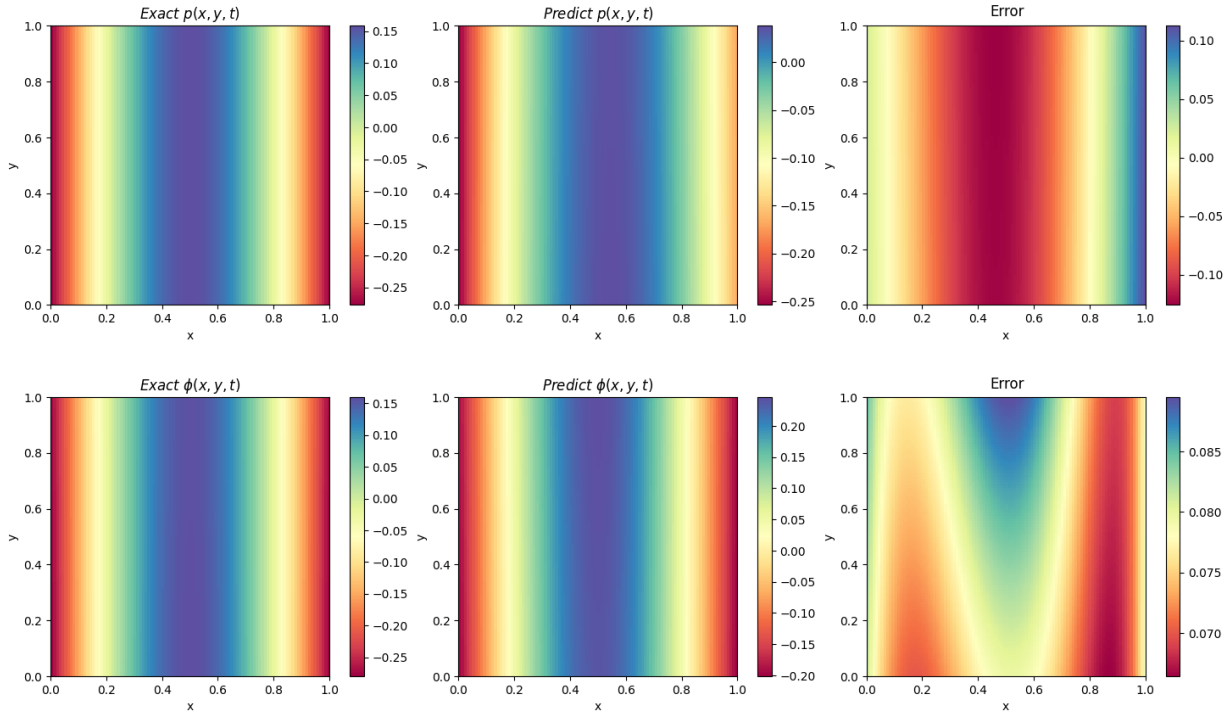


Figure 6. PINNs: Comparison of exact solutions and approximations for p and ϕ in $\tilde{\Gamma}$ at $t = 2$, with their errors.

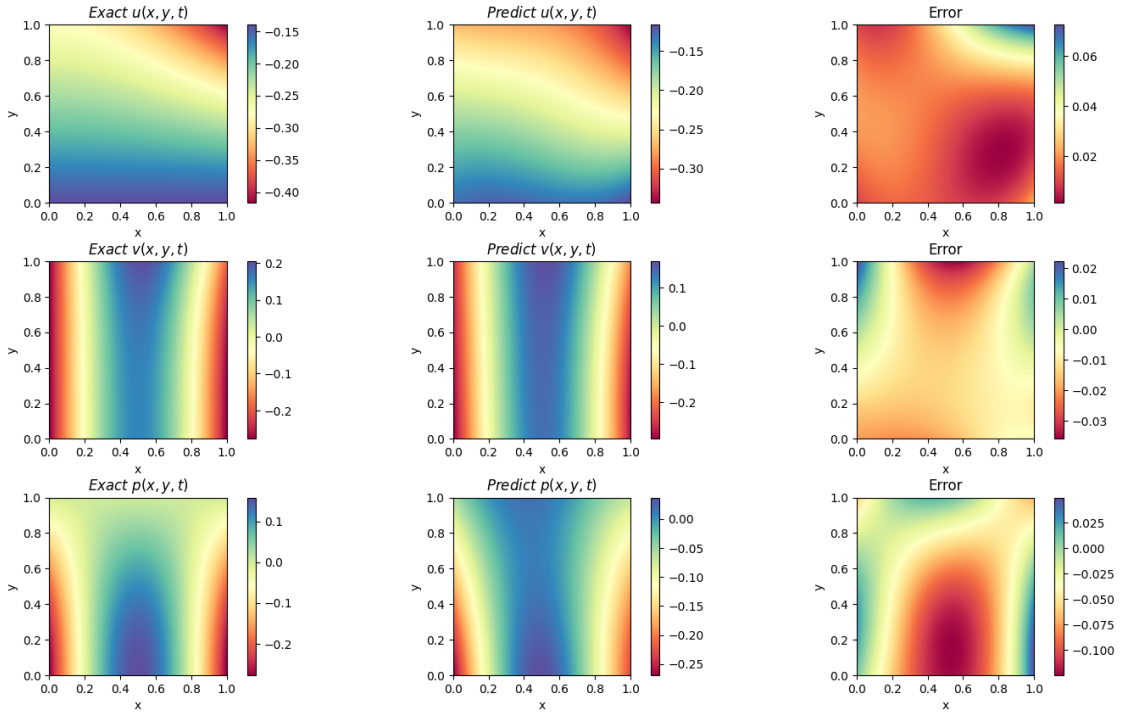


Figure 7. SI-PINNs: Comparison of exact solutions and approximations for u, v, and p in $\tilde{\Omega}_f$ at $t = 2$, with their errors.

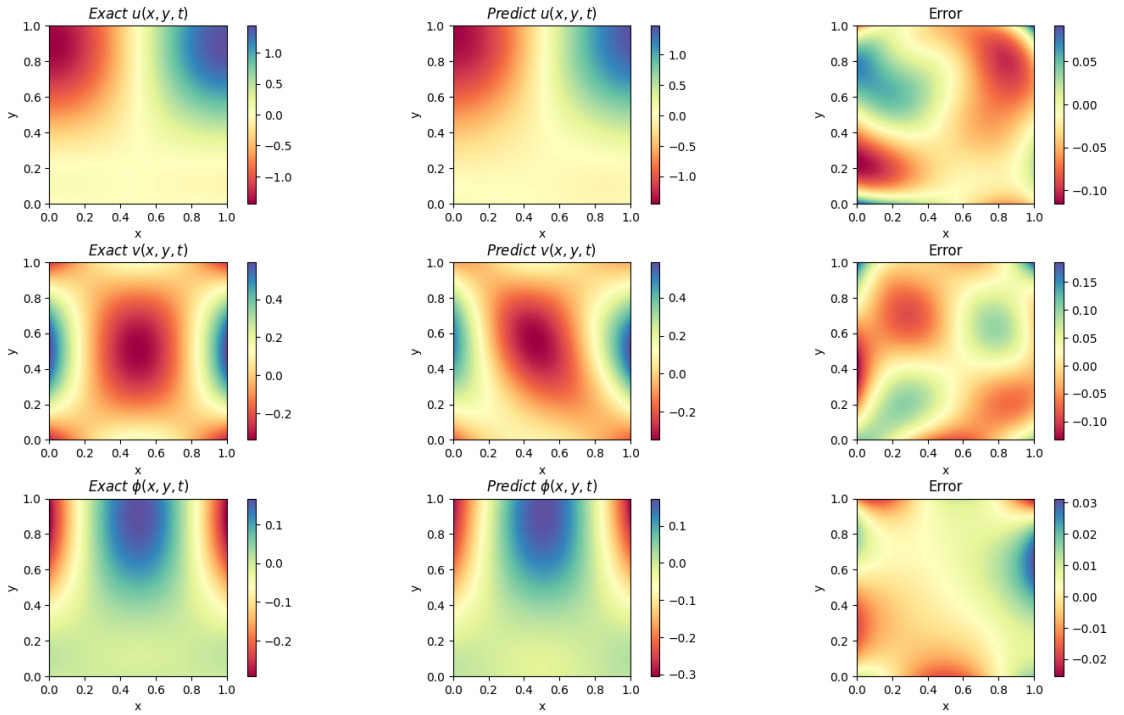


Figure 8. SI-PINNs: Comparison of exact solutions and approximations for u, v, and ϕ in $\tilde{\Omega}_p$ at $t = 2$, with their errors.

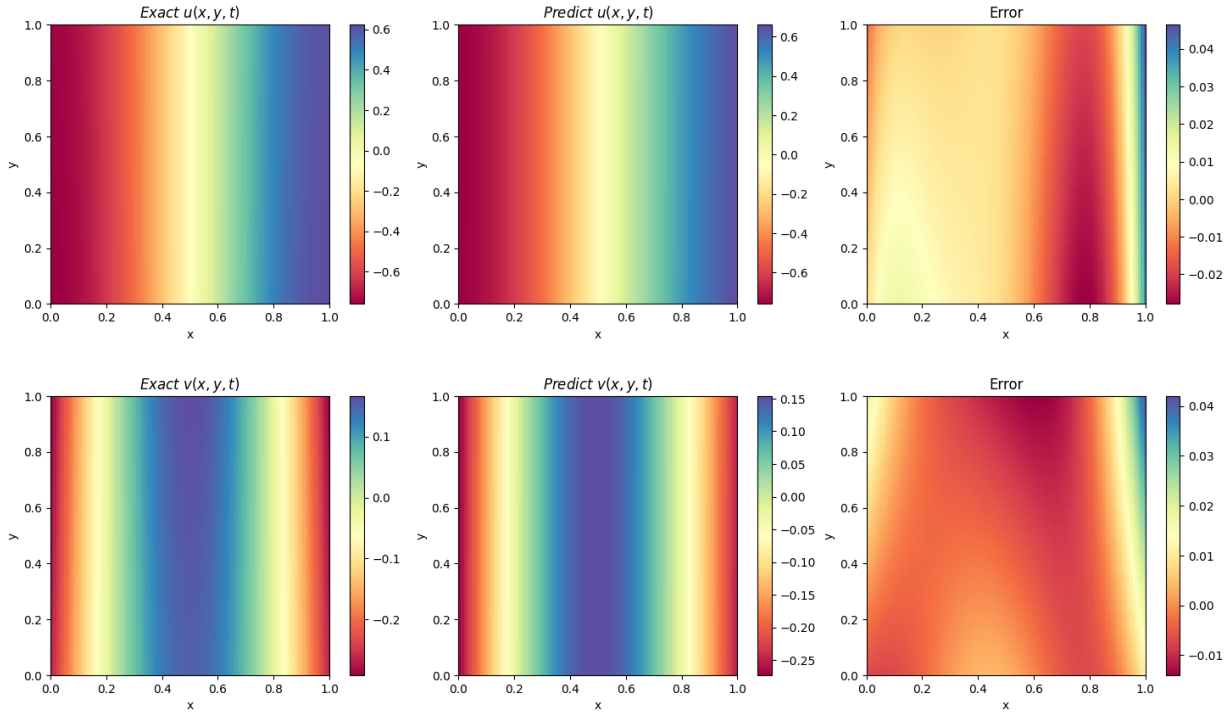


Figure 9. SI-PINNs: Comparison of exact solutions and approximations for u and v in $\tilde{\Gamma}$ at $t = 2$, with their errors.

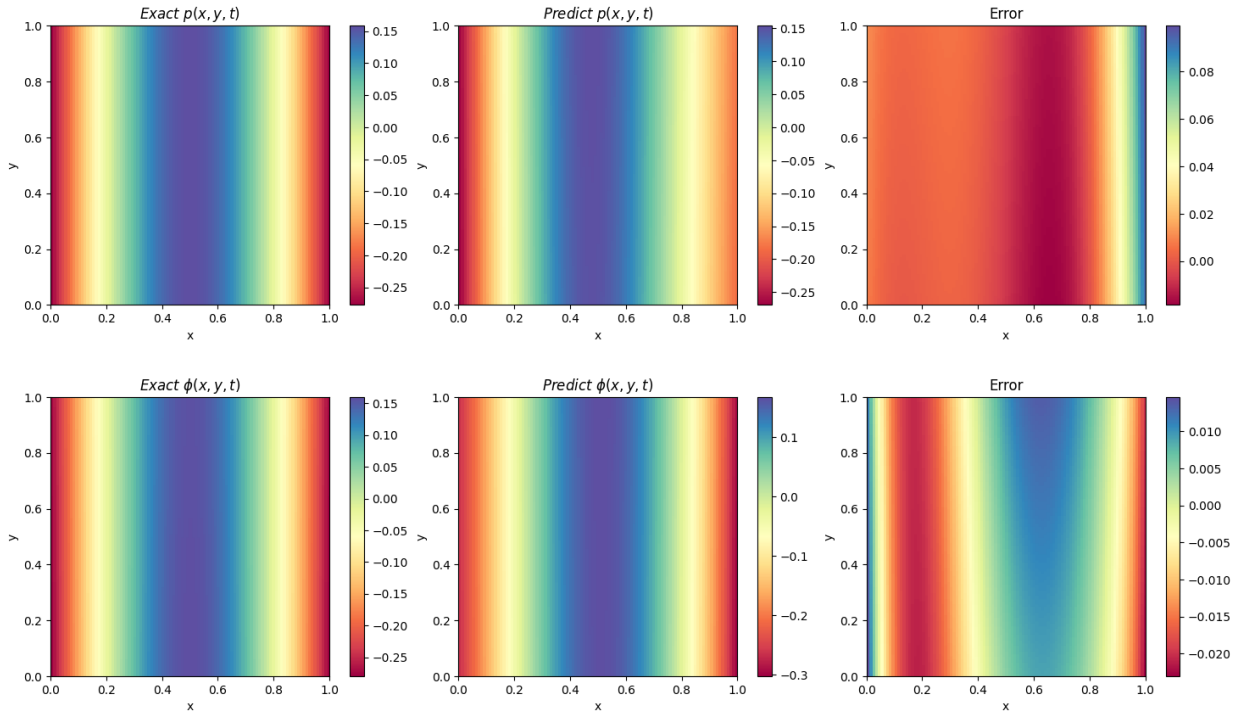


Figure 10. SI-PINNs: Comparison of exact solutions and approximations for p and ϕ in $\tilde{\Gamma}$ at $t = 2$, with their errors.

Table 1. Loss comparison of the SI-PINNs and the general PINNs in $\tilde{\Gamma}$ at $t = 2$.

	Total training time	ψ joint error	p and ϕ joint error
SI-PINNs	803.208 s	2.24×10^{-2}	1.53×10^{-2}
PINNs	8097.952 s	6.77×10^{-1}	1.27×10^{-1}

Table 2. Loss comparison of the SI-PINNs and the general PINNs in L^2 norm at $t = 2$.

	u_f	v_f	p	u_p	v_p	ϕ
SI-PINNs	1.30×10^{-2}	2.33×10^{-2}	6.71×10^{-2}	1.65×10^{-2}	6.67×10^{-2}	1.75×10^{-2}
PINNs	5.79×10^{-2}	1.06×10^{-1}	1.72×10^{-1}	1.05×10^{-1}	4.71×10^{-1}	1.09×10^{-1}

5. Conclusions

This paper proposes the subregion–identified physics-informed neural networks (SI-PINNs) to solve complex interface problems, such as the Navier–Stokes/Darcy model. By identifying the regional features of the neural networks and datasets for classification, SI-PINNs reduce the requirements on the size and composition of the training data, splitting the complexity of physical information that the networks need to process. All the three subregion-specific low-complexity neural networks together can incorporate all the physical information while effectively reducing the training cost of the neural networks. The key issue of the proposed method is its ability to clearly identify and classify different subregions, which may pose certain difficulties and limitations in some complex problems, such as the identification of the original interface in the model and the interface subregion constructed in the algorithm.

Use of AI tools declaration

The authors declare they have not used artificial intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors would like to extend special thanks to all the reviewers for their guidance and assistance.

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. T. Arbogast, D. S. Brunson, A computational method for approximating a Darcy-Stokes system governing a vuggy porous medium, *Comput. Geosci.*, **11** (2007), 207–218. https://doi.org/10.1007/s10596-007-9043-0

2. T. Arbogast, H. L. Lehr, Homogenization of a Darcy-Stokes system modeling vuggy porous media, *Comput. Geosci.*, **10** (2006), 291–302. https://doi.org/10.1007/s10596-006-9024-8

3. J. Hou, D. Hu, X. M. He, C. Qiu, Modeling and a Robin-type decoupled finite element method for dual-porosity-Navier-Stokes system with application to flows around multistage fractured horizontal wellbore, *Comput. Methods Appl. Mech. Eng.*, **388** (2022), 114248. https://doi.org/10.1016/j.cma.2021.114248

4. J. Hou, M. Qiu, X. M. He, C. Guo, M. Wei, B. Bai, A dual-porosity-Stokes model and finite element method for coupling dual-porosity flow and free flow, *SIAM J. Sci. Comput.*, **38** (2016), B710–B739. https://doi.org/10.1137/15M1044072

5. Y. Zhang, C. Zhou, C. Qu, M. Wei, X. M. He, B. Bai, Fabrication and verification of a glass-silicon-glass micro-nanofluidic model for investigating multi-phase flow in unconventional dual-porosity porous media, *Lab Chip*, **19** (2019), 4071–4082. https://doi.org/10.1039/C9LC00847K

6. Y. Gao, D. Han, X. M. He, U. Rüde, Unconditionally stable numerical methods for Cahn-Hilliard-Navier-Stokes-Darcy system with different densities and viscosities, *J. Comput. Phys.*, **454** (2022), 110968. https://doi.org/10.1016/j.jcp.2022.110968

7. Y. Gao, X. M. He, L. Mei, X. Yang, Decoupled, linear, and energy stable finite element method for the Cahn-Hilliard-Navier-Stokes-Darcy phase field model, *SIAM J. Sci. Comput.*, **40** (2018), B110–B137. https://doi.org/10.1137/16M1100885

8. M. Discacciati, R. Oyarzúa, A conforming mixed finite element method for the Navier-Stokes/Darcy coupled problem, *Numer. Math.*, **135** (2017), 571–606. https://doi.org/10.1007/s00211-016-0811-4

9. W. J. Layton, F. Schieweck, I. Yotov, Coupling fluid flow with porous media flow, *SIAM J. Numer. Anal.*, **40** (2002), 2195–2218. https://doi.org/10.1137/S0036142901392766

10. M. Discacciati, E. Miglio, A. Quarteroni, Mathematical and numerical models for coupling surface and groundwater flows, *Appl. Numer. Math.*, **43** (2002), 57–74. https://doi.org/10.1016/S0168-9274(02)00125-3

11. D. Vassilev, C. Wang, I. Yotov, Domain decomposition for coupled Stokes and Darcy flows, *Comput. Methods Appl. Mech. Eng.*, **268** (2014), 264–283. https://doi.org/10.1016/j.cma.2013.09.009

12. M. Discacciati, A. Quarteroni, A. Valli, Robin-Robin domain decomposition methods for the Stokes-Darcy coupling, *SIAM J. Numer. Anal.*, **45** (2007), 1246–1268. https://doi.org/10.1137/06065091X

13. M. Discacciati, P. Gervasio, A. Giacomini, A. Quarteroni, The interface control domain decomposition method for Stokes-Darcy coupling, *SIAM J. Numer. Anal.*, **54** (2016), 1039–1068. https://doi.org/10.1137/15M101854X

14. M. Gunzburger, X. M. He, B. Li, On Stokes-Ritz projection and multistep backward differentiation schemes in decoupling the Stokes-Darcy model, *SIAM J. Numer. Anal.*, **56** (2018), 397–427. https://doi.org/10.1137/16M1099601

15. Y. Cao, M. Gunzburger, X. M. He, X. Wang, Parallel, non-iterative, multi-physics domain decomposition methods for time-dependent Stokes-Darcy systems, *Math. Comput.*, **83** (2014), 1617–1644. https://doi.org/10.1090/S0025-5718-2014-02779-8

16. Y. Liu, Y. Boubendir, X. M. He, Y. He, New optimized Robin-Robin domain decomposition methods using Krylov solvers for the Stokes-Darcy system, *SIAM J. Sci. Comput.*, **44** (2022), B1068–B1095. https://doi.org/10.1137/21M1417223

17. P. Chidyagwai, B. Rivière, On the solution of the coupled Navier-Stokes and Darcy equations, *Comput. Methods Appl. Mech. Eng.*, **198** (2009), 3806–3820. https://doi.org/10.1016/j.cma.2009.08.012

18. B. Rivière, I. Yotov, Locally conservative coupling of Stokes and Darcy flows, *SIAM J. Numer. Anal.*, **42** (2005), 1959–1977. https://doi.org/10.1137/S0036142903427640

19. G. Kanschat, B. Riviere, A strongly conservative finite element method for the coupling of Stokes and Darcy flow, *J. Comput. Phys.*, **229** (2010), 5933–5943. https://doi.org/10.1016/j.jcp.2010.04.021

20. J. Camaño, G. N. Gatica, R. Oyarzúa, R. Ruiz-Baier, P. Venegas, New fully-mixed finite element methods for the Stokes-Darcy coupling, *Comput. Methods Appl. Mech. Eng.*, **295** (2015), 362–395. https://doi.org/10.1016/j.cma.2015.07.007

21. K. Lipnikov, D. Vassilev, I. Yotov, Discontinuous Galerkin and mimetic finite difference methods for coupled Stokes-Darcy flows on polygonal and polyhedral grids, *Numer. Math.*, **126** (2014), 321–360. https://doi.org/10.1007/s00211-013-0563-3

22. N. Jiang, H. Yang, Highly efficient ensemble algorithms for computing the Stokes-Darcy equations, *Comput. Methods Appl. Mech. Eng.*, **418** (2024), 116562. https://doi.org/10.1016/j.cma.2023.116562

23. Y. Boubendir, S. Tlupova, Domain decomposition methods for solving Stokes-Darcy problems with boundary integrals, *SIAM J. Sci. Comput.*, **35** (2013), B82–B106. https://doi.org/10.1137/110838376

24. Y. Boubendir, S. Tlupova, Stokes-Darcy boundary integral solutions using preconditioners, *J. Comput. Phys.*, **228** (2009), 8627–8641. https://doi.org/10.1016/j.jcp.2009.08.014

25. M. Mu, J. Xu, A two-grid method of a mixed Stokes-Darcy model for coupling fluid flow with porous media flow, *SIAM J. Numer. Anal.*, **45** (2007), 1801–1813. https://doi.org/10.1137/050637820

26. S. Münzenmaier, G. Starke, First-order system least squares for coupled Stokes-Darcy flow, *SIAM J. Numer. Anal.*, **49** (2011), 387–404. https://doi.org/10.1137/100805108

27. S. Zeng, Z. Xie, X. Yang, J. Wang, Fully discrete, decoupled and energy-stable Fourier-spectral numerical scheme for the nonlocal Cahn-Hilliard equation coupled with Navier-Stokes/Darcy flow regime of two-phase incompressible flows, *Comput. Methods Appl. Mech. Eng.*, **415** (2023), 116289. https://doi.org/10.1016/j.cma.2023.116289

28. W. Chen, M. Gunzburger, D. Sun, X. Wang, An efficient and long-time accurate third-order algorithm for the Stokes-Darcy system, *Numer. Math.*, **134** (2016), 857–879. https://doi.org/10.1007/s00211-015-0789-3

29. V. Girault, D. Vassilev, I. Yotov, Mortar multiscale finite element methods for Stokes-Darcy flows, *Numer. Math.*, **127** (2014), 93–165. https://doi.org/10.1007/s00211-013-0583-z
30. X. Li, W. Gong, X. M. He, T. Lin, Variational data assimilation and its decoupled iterative numerical algorithms for Stokes-Darcy model, *SIAM J. Sci. Comput.*, **46** (2024), S142–S175. https://doi.org/10.1137/22M1492994
31. C. Mingard, H. Rees, G. V. Pérez, A. A. Louis, Deep neural networks have an inbuilt Occam's razor, *Nat. Commun.*, **16** (2025), 220. https://doi.org/10.1038/s41467-024-54813-x
32. Q. Guan, How can deep neural networks fail even with global optima?, *Int. J. Numer. Anal. Model.*, **21** (2024), 674–696. https://doi.org/10.4208/ijnam2024-1027
33. L. Storm, H. Linander, J. Bec, K. Gustavsson, B. Mehlig, Finite-time Lyapunov exponents of deep neural networks, *Phys. Rev. Lett.*, **132** (2024), 057301. https://doi.org/10.1103/PhysRevLett.132.057301
34. J. Cheng, Q. Li, T. Lin, Z. Shen, Interpolation, approximation, and controllability of deep neural networks, *SIAM J. Control Optim.*, **63** (2025), 625–649. https://doi.org/10.1137/23M1599744
35. Z. Zhang, F. Bao, G. Zhang, Improving the expressive power of deep neural networks through integral activation transform, *Int. J. Numer. Anal. Model.*, **21** (2024), 739–763. https://doi.org/10.4208/ijnam2024-1030
36. D. Frerichs-Mihov, L. Henning, V. John, Using deep neural networks for detecting spurious oscillations in discontinuous Galerkin solutions of convection-dominated convection–diffusion equations, *J. Sci. Comput.*, **97** (2023), 36. https://doi.org/10.1007/s10915-023-02335-x
37. S. Ji, S. Peng, Y. Peng, X. Zhang, A novel control method for solving high-dimensional Hamiltonian systems through deep neural networks, *SIAM J. Sci. Comput.*, **47** (2025), C873–C898. https://doi.org/10.1137/24M1647084
38. B. Chudomelka, Y. Hong, J. Morgan, H. Kim, J. Park, Deep neural network for solving differential equations motivated by Legendre-Galerkin approximation , *Int. J. Numer. Anal. Model.*, **21** (2024), 652–673. https://doi.org/10.4208/ijnam2024-1026
39. J. Y. Lee, J. W. Jang, H. J. Wang, The model reduction of the Vlasov–Poisson–Fokker–Planck system to the Poisson–Nernst–Planck system via the deep neural network approach, *ESAIM: M2AN*, **55** (2021), 1803–1846. https://doi.org/10.1051/m2an/2021038
40. C. Fan, M. A. Ali, Z. Zhang, Decoupling numerical method based on deep neural network for nonlinear degenerate interface problems, *Comput. Phys. Commun.*, **303** (2024), 109275. https://doi.org/10.1016/j.cpc.2024.109275
41. M. Raissi, P. Perdikaris, 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** (2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045
42. S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, *SIAM J. Sci. Comput.*, **43** (2021), A3055–A3081. https://doi.org/10.1137/20M1318043
43. W. Cao, W. Zhang, An analysis and solution of ill-conditioning in physics-informed neural networks, *J. Comput. Phys.*, **520** (2025), 113494. https://doi.org/10.1016/j.jcp.2024.113494

44. X. Jin, S. Cai, H. Li, G. E. Karniadakis, NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations, *J. Comput. Phys.*, **426** (2021), 109951. https://doi.org/10.1016/j.jcp.2020.109951

45. A. F. Psaros, K. Kawaguchi, G. E. Karniadakis, Meta-learning PINN loss functions, *J. Comput. Phys.*, **458** (2022), 111121. https://doi.org/10.1016/j.jcp.2022.111121

46. Q. Zhang, C. Qiu, J. Hou, W. Yan, Advanced physics-informed neural networks for numerical approximation of the coupled Schrödinger–KdV equation, *Commun. Nonlinear Sci. Numer. Simul.*, **138** (2024), 108229. https://doi.org/10.1016/j.cnsns.2024.108229

47. Z. Gao, Z. Fu, M. Wen, Y. Guo, Y. Zhang, Physical informed neural network for thermo-hydral analysis of fire-loaded concrete, *Eng. Anal. Boundary Elem.*, **158** (2024), 252–261. https://doi.org/10.1016/j.enganabound.2023.10.027

48. S. Xu, Y. Dai, C. Yan, Z. Sun, R. Huang, D. Guo, et al., On the preprocessing of physics-informed neural networks: How to better utilize data in fluid mechanics, *J. Comput. Phys.*, **528** (2025), 113837. https://doi.org/10.1016/j.jcp.2025.113837

49. Z. Fu, W. Xu, S. Liu, Physics-informed kernel function neural networks for solving partial differential equations, *Neural Networks*, **172** (2024), 106098. https://doi.org/10.1016/j.neunet.2024.106098

50. A. D. Jagtap, E. Kharazmi, G. E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, *Comput. Methods Appl. Mech. Eng.*, **365** (2020), 113028. https://doi.org/10.1016/j.cma.2020.113028

51. A. D. Jagtap, G. E. Karniadakis, Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, *Commun. Comput. Phys.*, **28** (2020), 2002–2041. https://doi.org/10.4208/cicp.OA-2020-0164

52. A. Kopaničáková, H. Kothari, G. E. Karniadakis, R. Krause, Enhancing training of physics-informed neural networks using domain decomposition–based preconditioning strategies, *SIAM J. Sci. Comput.*, **46** (2024), S46–S67. https://doi.org/10.1137/23M1583375

53. V. Dolean, A. Heinlein, S. Mishra, B. Moseley, Multilevel domain decomposition-based architectures for physics-informed neural networks, *Comput. Methods Appl. Mech. Eng.*, **429** (2024), 117116. https://doi.org/10.1016/j.cma.2024.117116

54. C. Si, M. Yan, Initialization-enhanced physics-informed neural network with domain decomposition (IDPINN), *J. Comput. Phys.*, **530** (2025), 113914. https://doi.org/10.1016/j.jcp.2025.113914

55. G. S. Beavers, D. D. Joseph, Boundary conditions at a naturally permeable wall, *J. Fluid Mech.*, **30** (1967), 197–207. https://doi.org/10.1017/S0022112067001375

56. Y. Cao, M. Gunzburger, F. Hua, X. Wang, Coupled Stokes-Darcy model with Beavers-Joseph interface boundary condition, *Commun. Math. Sci.*, **8** (2010), 1–25. https://doi.org/10.4310/CMS.2010.v8.n1.a2

57. Y. Cao, M. Gunzburger, X. Hu, F. Hua, X. Wang, W. Zhao, Finite element approximations for Stokes-Darcy flow with Beavers-Joseph interface conditions, *SIAM J. Numer. Anal.*, **47** (2010), 4239–4256. https://doi.org/10.1137/080731542
58. Y. Cao, M. Gunzburger, X. M. He, X. Wang, Robin-Robin domain decomposition methods for the steady-state Stokes-Darcy system with the Beaver-Joseph interface condition, *Numer. Math.*, **117** (2011), 601–629. https://doi.org/10.1007/s00211-011-0361-8
59. X. M. He, J. Li, Y. Lin, J. Ming, A domain decomposition method for the steady-state Navier-Stokes-Darcy model with Beavers-Joseph interface condition, *SIAM J. Sci. Comput.*, **37** (2015), S264–S290. https://doi.org/10.1137/140965776
60. K. Hornik, Approximation capabilities of multilayer feedforward networks, *Neural Networks*, **4** (1991), 251–257. https://doi.org/10.1016/0893-6080(91)90009-T
61. D. Kingma, J. Ba, Adam: A method for stochastic optimization, preprint, arXiv:1412.6980.
62. L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, *SIAM Rev.*, **63** (2021), 208–228. https://doi.org/10.1137/19M1274067
63. V. Nair, G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Omnipress, (2010), 807–814.
64. K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on imagenet classification, in *Proceedings of the IEEE International Conference on Computer Vision*, IEEE, (2015), 1026–1034.
65. P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, preprint, arXiv:1710.05941.
66. X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, PMLR, (2010), 249–256.
67. L. Lu, Y. Shin, Y. Su, G. E. Karniadakis, Dying ReLU and initialization: Theory and numerical examples, *Commun. Comput. Phys.*, **28** (2020), 1671–1706. https://doi.org/10.4208/cicp.OA-2020-0165
68. M. Stein, Large sample properties of simulations using Latin hypercube sampling, *Technometrics*, **29** (1987), 143–151. https://doi.org/10.1080/00401706.1987.10488205

AIMS Press