energies

MDPI

Article

On the Monotonicity and Positivity of Physics-Informed Neural Networks for Highly Anisotropic Diffusion Equations

Wenjuan Zhang and Mohammed Al Kobaisi *

Department of Petroleum Engineering, Khalifa University of Science and Technology, Abu Dhabi 127788, United Arab Emirates
* Correspondence: mohammed.alkobaisi@ku.ac.ae

Abstract: Physics-informed neural network (PINN) models are developed in this work for solving highly anisotropic diffusion equations. Compared to traditional numerical discretization schemes such as the finite volume method and finite element method, PINN models are meshless and, therefore, have the advantage of imposing no constraint on the orientations of the diffusion tensors or the grid orthogonality conditions. To impose solution positivity, we tested PINN models with positivity-preserving activation functions for the last layer and found that the accuracy of the corresponding PINN solutions is quite poor compared to the vanilla PINN model. Therefore, to improve the monotonicity properties of PINN models, we propose a new loss function that incorporates additional terms which penalize negative solutions, in addition to the usual partial differential equation (PDE) residuals and boundary mismatch. Various numerical experiments show that the PINN models can accurately capture the tensorial effect of the diffusion tensor, and the PINN model utilizing the new loss function can reduce the degree of violations of monotonicity and improve the accuracy of solutions compared to the vanilla PINN model, while the computational expenses remain comparable. Moreover, we further developed PINN models that are composed of multiple neural networks to deal with discontinuous diffusion tensors. Pressure and flux continuity conditions on the discontinuity line are used to stitch the multiple networks into a single model by adding another loss term in the loss function. The resulting PINN models were shown to successfully solve the diffusion equation when the principal directions of the diffusion tensor change abruptly across the discontinuity line. The results demonstrate that the PINN models represent an attractive option for solving difficult anisotropic diffusion problems compared to traditional numerical discretization methods.

Keywords: subsurface flow; porous media; permeability anisotropy; diffusion equation; physics informed neural networks; monotonicity

Citation: Zhang, W.; Al Kobaisi, M. On the Monotonicity and Positivity of Physics-Informed Neural Networks for Highly Anisotropic Diffusion Equations. *Energies* **2022**, *15*, 6823. https://doi.org/10.3390/en15186823

Academic Editor: Vasily Novozhilov

Received: 27 August 2022
Accepted: 15 September 2022
Published: 18 September 2022

1. Introduction

Modeling fluid flow in subsurface porous media often requires solving the following set of partial differential equations (PDE):

$$\begin{aligned} \nabla \cdot \mathbf{v}(\mathbf{x}) &= q(\mathbf{x}), \\ \mathbf{v}(\mathbf{x}) &= -\mathbf{K}(\mathbf{x})\nabla u(\mathbf{x}), \quad \mathbf{x} \in \Omega, \end{aligned} \tag{1}$$

with appropriate boundary conditions such as:

$$\mathcal{B}(u, \mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega, \tag{2}$$

where Ω is the computational domain with boundary $\partial\Omega$; $\mathbf{v}(\mathbf{x})$ is the fluid velocity; and $u(\mathbf{x})$ is the unknown function representing fluid pressure/potential that needs to be solved. Here, we have assumed the fluid viscosity to be unity. $q(\mathbf{x})$ is the fluid source/sink term. The diffusion coefficient $\mathbf{K}$ is the permeability of the porous media and it is, in general, a full tensor because of the anisotropic nature of the media, which poses great challenges to

many numerical methods. Cell-centered finite volume methods are the most widely used methods in the reservoir simulation community because of their property of being locally mass conservative. The linear two-point flux approximation (TPFA) method is the default choice for many commercial and academic reservoir simulators for its simplicity, efficiency and robustness [1,2], but it is well known to be inconsistent when the principal directions of the permeability tensor are not aligned with the mesh, and the numerical solution does not converge to the true solution when the mesh is refined. More specifically, the TPFA method is only applicable for the so-called **K**-orthogonal mesh, which can be thought of as an orthogonal mesh induced by the permeability tensor **K**. To amend the shortcomings of the TPFA method, a family of multi-point flux approximation (MPFA) methods have been proposed in the past two decades that can handle full permeability tensors on general non-orthogonal meshes [3–6]. However, the MPFA methods suffer from monotonicity issues and the solutions can contain spurious oscillations when the media is highly anisotropic, or the mesh has large aspect ratios [7–12]. Specifically, the numerical solution of MPFA can become negative, which is unphysical. To fulfill the requirements of both consistency and monotonicity, a class of nonlinear finite volume methods has been developed in the past decade [13–19]. The nonlinear methods such as the nonlinear two-point flux approximation (NTPFA) method are designed specifically to preserve the non-negativity of the numerical solutions. Therefore, the methods are monotone by design. However, the nonlinear methods lead to a system of nonlinear equations after discretization, even though the original equation to be solved is linear. Moreover, a nonlinear solver has to be utilized, which increases the computational expenses. In addition to finite volume methods, another popular consistent discretization method that is also locally mass conservative is the mixed finite element (MFE) method [20]. The MFE method treats both the fluid velocity, **v,** and fluid pressure/potential, u, as primary unknowns and solves for them simultaneously. Therefore, the number of unknowns is usually much larger than that of cell-centered finite volume methods. Similar to the classical finite element method (FEM), the solution quality of MFE is also closely tied to the quality of the mesh, which can become a problem for practical reservoir simulations because the simulation grids are often dictated by the complex geological constraints, and we are not free to choose the desired meshes for simulation. Moreover, finding suitable basis functions for arbitrary polyhedral elements is also a challenge. Another family of methods that share certain similarities to the MFE method but are much more flexible for general polygonal meshes in 2D and polyhedral meshes in 3D are the mimetic finite difference methods (MFD) [21]. The MFD methods are designed to not only provide consistent discretizations, but also to mimic the fundamental properties of the differential operators; they are currently used in a wide range of applications. The unknowns of MFD methods include fluxes across cell interfaces, fluid pressure/potential at cell centroids and at face centroids. Therefore, the final discretized system of equations is much larger than that of cell-centered finite volume methods.

Due to the tremendous advancements in both the hardware and algorithm/software designs in recent years, machine learning, and deep learning in particular, has made significant strides in solving many science and engineering problems. As a result, a growing body of literature has been devoted to leveraging the power of machine learning for scientific computing. Traditional machine learning applications require a large amount of labeled data for training the machine learning model. However, for many complex engineering problems such as modeling fluid flow in porous media, it is infeasible or computationally too expensive to obtain the amount of labeled data required for machine learning. To circumvent this problem, Raissi et al. [22] revisited the idea of approximating the solution to a partial differential equation by using a deep neural network. By taking advantage of automatic differentiation, the residual of the partial differential equation evaluated on a set of internal collocation points, in addition to any mismatch in boundary and initial conditions, are combined together to form a loss function that is minimized, thus, encoding the underlying physical laws into the model as prior information. The resulting machine

learning model is termed the physics-informed neural network (PINN) in their work. They demonstrated that the PINN can achieve excellent predictive accuracy for various computational problems and, thus, has the potential to serve as an alternative to traditional numerical discretization schemes such as the finite volume, finite element method, etc. Similar to some meshless methods such as the reproducing kernel particle method (RKPM) [23], the PINN method does not require the discretization of the computational domain by a mesh. Unlike the RKPM that approximates the unknown function by a linear combination of a set of reproducing kernel shape functions, the PINN method utilizes the neural network to approximate the unknown function by taking advantage of the fact that the neural networks are universal function approximators [24]. The task of solving a PDE is transformed into a minimization problem that can be solved effectively by employing powerful optimization algorithms and advanced computer hardware developed by the machine learning community. Moreover, PINN can be used to solve the inverse problems as easily as the forward problem. As a result, the scientific computing community has shown considerable interest in PINN [25–30], and it was applied to solve a wide variety of problems [31–34].

Regarding subsurface flow and transport modeling, Fuks and Tchelepi [35] applied the PINN to solve the standard Buckley–Leverett model governing incompressible and immiscible two-phase flow in porous media and found that, depending on the shape of the flux function, the PINN can struggle or even fail to find the correct solution, suggesting that there is still some way to go before PINN can be reliably used for reservoir simulation, despite its many successes in diverse fields. In our previous work [36], we investigated the performance of PINN for solving the anisotropic diffusion equation which governs incompressible single-phase flow in porous media and found that the PINN enjoys excellent performance for problems with smooth solutions but can struggle for more difficult problems, especially for highly anisotropic permeability tensors. Therefore, in this work, we advance our previous work by improving on the PINN model for solving the challenging anisotropic diffusion equation. Specifically, we focus on the monotonicity requirement for highly anisotropic full permeability tensors that have beset many discretization schemes, as mentioned previously. The contributions of this work are mainly twofold: firstly, given the problem at hand, we embed more physics into the neural network by adding to the loss function an additional loss term, which accounts for the monotonicity requirements explicitly; secondly, the PINN model is improved to deal with discontinuous permeability fields. The rest of the paper is organized as follows: a brief review of the PINN formulations as well as different loss functions are given in the next section; the results section presents the numerical experiments including homogeneous, rotating and discontinuous permeability tensor fields; and finally, the conclusions are given in Section 4.

2. Methods, Models and Formulations

The details of PINN can be found in the literature, such as [22,30]. For completeness, we offer a brief review of the PINN model here. We consider a fully connected feed-forward neural network denoted by $\hat{u}(\mathbf{x};\theta)$ to approximate the solution $u(\mathbf{x})$. The neural network consists of L layers. The relationship between the input and output of layer l is given by:

$$x^l = h^l\left(x^{l-1}\right) = \sigma^l\left(W^l x^{l-1} + b^l\right),\ l = 1, 2, \ldots, L, \tag{3}$$

where x^{l-1} and x^l are the input and output of layer l, respectively; W^l is the weighting matrix and b^l is the bias term; σ^l is the nonlinear activation function of layer l. The input to the first layer is x^0, which is also the input to the whole network and the output of the last layer L is x^L that corresponds to the output of the whole neural network. The activation function σ^L for the last layer is often set as the identity mapping. The collection of all the weighting matrices and bias terms comprise the parameters θ of the neural network. That is, $\theta := \left\{W^l,\ b^l\right\}_{l=1}^{L}$.

Since $\hat{u}(\mathbf{x};\theta)$ is unlikely to be the true solution, substituting $\hat{u}(\mathbf{x};\theta)$ into Equation (1) leads to the following PDE residual:

$$R(\mathbf{x}) = -\nabla \cdot (\mathbf{K}(\mathbf{x})\nabla \hat{u}(\mathbf{x};\theta)) - q(\mathbf{x}),\ \mathbf{x} \in \Omega. \tag{4}$$

Similarly, inserting $\hat{u}(\mathbf{x};\theta)$ into the boundary condition (2) leads to a boundary mismatch. The idea of PINN is to find the best approximation by minimizing the PDE residual and the boundary mismatch. The PDE residual is calculated on a set of collocation points $\left\{\mathbf{x}_i^r\right\}_{i=1}^{N_r}$ that lies in Ω, while the boundary mismatch is calculated on a set of boundary points $\left\{\mathbf{x}_i^b\right\}_{i=1}^{N_b}$ that lies on $\partial\Omega$, where N_r and N_b denote the number of collocation points and boundary points, respectively. We will call both the collocation points and the boundary points our training points. The problem of solving the PDE is then transformed into a minimization problem with the following loss function to be minimized over the parameters θ of the neural network:

$$\mathcal{L}(\theta) = \omega_r \mathcal{L}_r(\theta) + \omega_b \mathcal{L}_b(\theta), \tag{5}$$

where ω_r and ω_b are the weighting coefficients for the residual and boundary components, respectively. $\mathcal{L}_r(\theta)$ and $\mathcal{L}_b(\theta)$ denote the loss from the PDE residual and the boundary mismatch, respectively. The mean square error (MSE) is usually used as a criterion to compute the loss:

$$\mathcal{L}_r(\theta) = \frac{1}{N_r}\sum_{i=1}^{N_r}\left|\mathcal{R}\left(\mathbf{x}_i^r\right)\right|^2 \tag{6}$$

$$\mathcal{L}_b(\theta) = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|\mathcal{B}\left(\hat{u}\left(\mathbf{x}_i^b;\theta\right),\ \mathbf{x}_i^b\right)\right|^2 \tag{7}$$

where N_r and N_b are the number of points internal to the domain Ω and on the boundary of Ω, respectively, for evaluating the residual and boundary mismatch. The set of training points $\mathbf{x}_i^r$ and $\mathbf{x}_i^b$ are usually sampled randomly in Ω and on $\partial\Omega$, respectively. When the training of the neural network finishes, all the parameters of the neural network are determined, and we are left with a continuous nonlinear function (the neural network with optimized parameters) that can be used to make predictions about the value of the unknown function at any point in the computational domain by a simple function evaluation.

One practical requirement for the development of any new numerical discretization scheme is the so-called monotonicity property; that is, the method should preserve the positivity (or non-negativity) of the differential solution [13,14]. Some commonly used numerical discretization schemes for solving the anisotropic diffusion equation, such as the linear muti-point flux approximation method and the mixed finite element method, are only conditionally monotone. Therefore, their solutions can become negative for highly anisotropic diffusion tensors. The nonlinear finite volume methods, on the other hand, are constructed by design to preserve the positivity of the numerical solutions at the expense of higher computational costs. Compared to those numerical discretization schemes, the high flexibility of neural networks makes it rather simple to enforce non-negativity requirements on the solutions. A seemingly straightforward way to achieve this is to use some positivity-preserving functions as our activation function σ^L for the last layer of the neural network. Obvious choices include the Rectified Linear Unit function (ReLU), the *Softplus* function, and the *Square* function (see Figure 1). Interestingly, however, we tested all three activation functions and found that the accuracy of the corresponding PINN solutions is quite poor compared to the vanilla PINN model, where the identity mapping is used as the activation function σ^L for the last layer (see results in Section 3.1) and the relative L^2 errors are almost two orders of magnitude higher than that of the vanilla PINN solution when the exact same configurations of neural networks are used. The reasons for this behavior are not yet

clear to us. Therefore, to improve the monotonicity properties of the PINN solutions, we propose the following loss function to enforce solution non-negativity in a soft manner:

$$\mathcal{L}(\theta) = \omega_r \mathcal{L}_r(\theta) + \omega_b \mathcal{L}_b(\theta) + \omega_m \mathcal{L}_m(\theta), \tag{8}$$

where $\mathcal{L}_r(\theta)$ and $\mathcal{L}_b(\theta)$ are still given by Equations (6) and (7), respectively. The last term is added to penalize solutions that are negative, and it is given by:

$$\mathcal{L}_m(\theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} ReLU\Big(-\hat{u}\Big(\mathbf{x}_i^b; \theta\Big)\Big), \tag{9}$$

where $ReLU(\cdot)$ is the ReLU activation function. To distinguish between the PINN model using the loss function (5) and the PINN model using the loss function (8), we denote them by PINN-MSE and PINN-Mono hereafter, respectively.

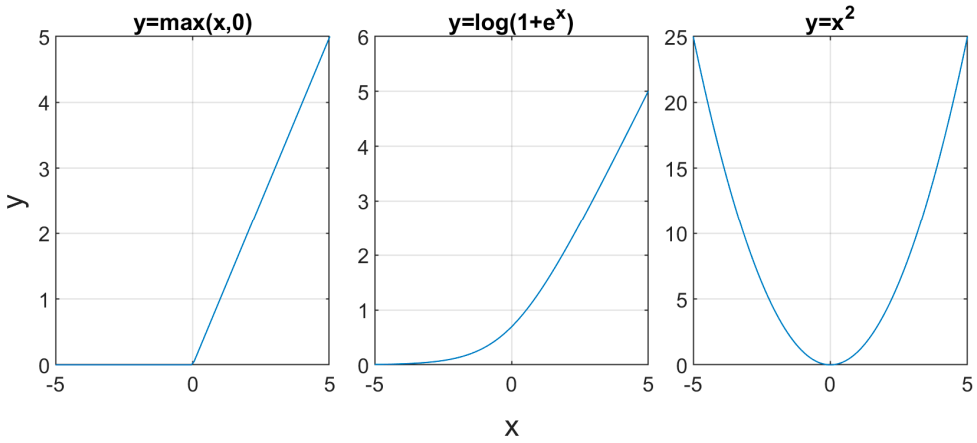

Figure 1. Activation functions preserving solution non-negativity. From left to right: ReLU, Softplus, and Square functions.

3. Results

This section presents the results of various numerical experiments. The first test case deals with a constant anisotropic permeability tensor, and a rotating heterogeneous permeability tensor is considered in the second case. In case 3, we extend the PINN models to account for discontinuous permeability fields. The neural network models are implemented in the open-source PyTorch [37] framework. Since the focus of this work is not to search for the optimal neural network architecture, all the PINN models used in the numerical experiments share the same network structure, a network with five hidden layers with each hidden layer containing 50 neurons. The input layer contains two neurons, and the output layer has one neuron. The *tanh* function is used as the activation function for all PINN models unless stated otherwise. All the experiments were run on a NVIDIA Quadro RTX 4000 GPU.

3.1. Case 1: Constant Permeability Tensor

We first solve the anisotropic diffusion Equation (1) on the unit square domain $\Omega = (0,1)^2$. The constant anisotropic permeability tensor is given by:

$$\mathbf{K} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}, \tag{10}$$

where k_1 and k_2 are the maximum and minimum principal value of the permeability tensor $\mathbf{K}$, respectively, and α is the rotation angle between the maximum principal direction and the positive x-axis. The permeability anisotropy ratio is defined as $k_r = \frac{k_1}{k_2}$. Homogeneous Dirichlet boundary conditions are applied on the boundaries of the domain and a concen-

trated source term is placed in the center of the domain to drive the flow. The radial basis function kernel is used as the source function, and it is given by:

$$q(\mathbf{x}) = 1000 \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_c\|^2}{2\sigma^2}\right) \tag{11}$$

where $\mathbf{x}$ is the position vector of any point belonging to the computational domain Ω, and $\mathbf{x}_c$ is the center point of Ω. In our case, $\mathbf{x}_c = (0.5,\ 0.5)$. σ is a parameter that controls the degree of concentration of the source function. A smaller σ means a more concentrated source function. The following values are used in this test case: $k_1 = 1$, $k_r = 1000$ and $\sigma = 0.02$. To evaluate the accuracy of PINN solutions, we first take $\alpha = 0$ and solve the equation numerically on a 256×256 Cartesian mesh using the linear TPFA method as our reference solution. The linear TPFA method is the golden standard for reservoir simulation, and it is well known to be both consistent and convergent for the so-called K-orthogonal meshes. The left of Figure 2 shows the source term, and the pressure solution using TPFA is shown on the right. Since the horizontal permeability (k_1) is 1000 times larger than the vertical permeability (k_2), the pressure diffusion mostly follows the horizontal direction.

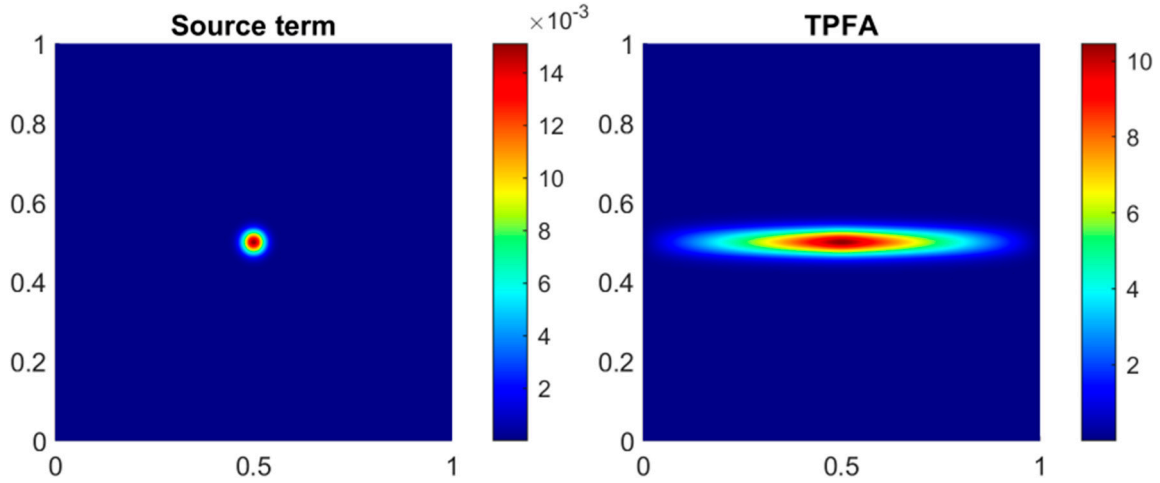

Figure 2. Source function (**left**) and pressure solution using TPFA (**right**) for case 1: $\theta = 0$.

We then solve the same problem using PINN with different loss functions. The Hammersley distribution is used to generate residual points for the training of PINN, since it has been demonstrated in the literature that this sampling distribution generally shows better performance than other distributions such as the random sampling [38,39]. A sample of residual points generated from the Hammersley distribution is shown in Figure 3. In our current implementation, 10,000 residual points are used for training and 800 evenly-spaced points are used on the boundary to enforce the boundary conditions. For the PINN-MSE model, the weighting parameters are given by $\omega_r = \omega_b = 1$ and for the PINN-Mono model, the weighting parameters are $\omega_r = \omega_b = 1$, $\omega_m = 10$. The left plot of Figure 4 shows the loss history of PINN solutions with the two different loss functions. We first train the models using the Adam optimizer for 10,000 epochs and then switch to the L-BFGS optimizer and continue the training for another 500 iterations. The L-BFGS optimizer is more expensive than the Adam optimizer for a single iteration, but it converges much faster, as can be seen clearly in the figure.

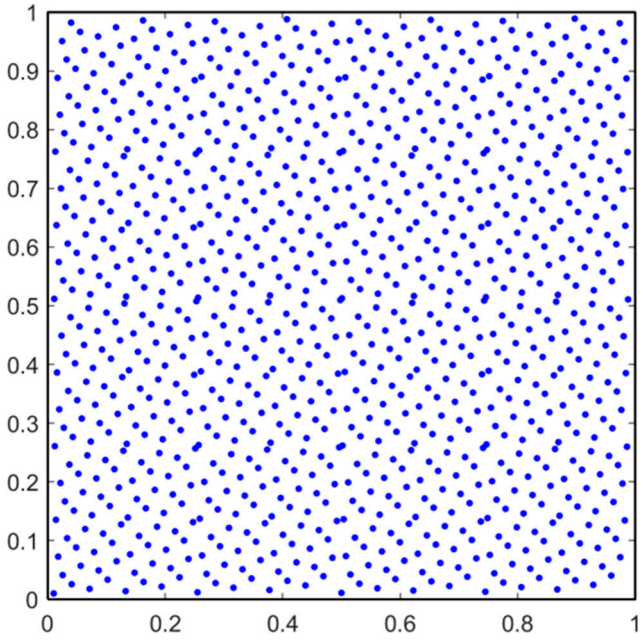

Figure 3. Hammersley distribution of residual points for the PINN model.

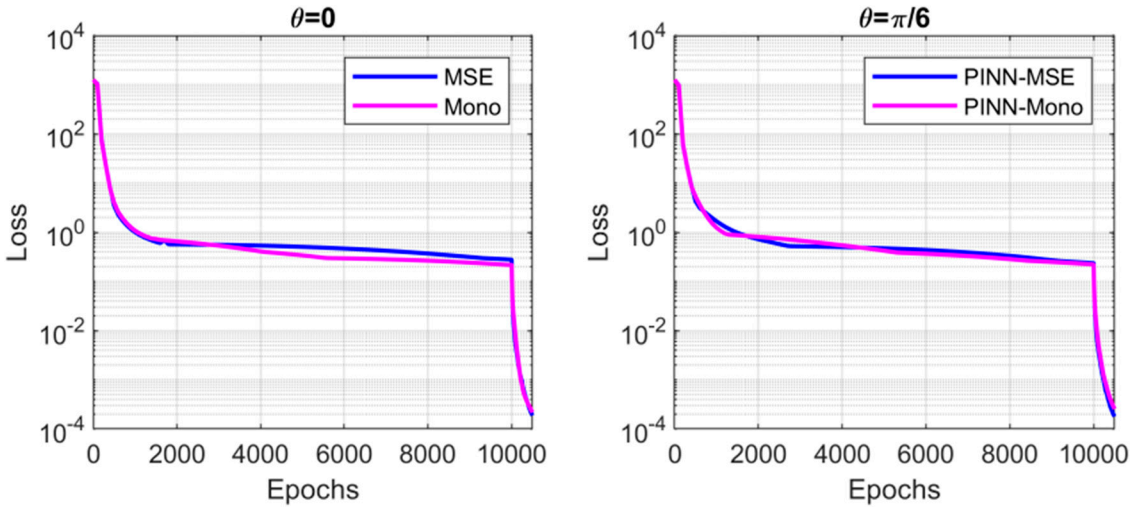

Figure 4. Loss history of PINN for case 1. **Left**: $\theta = 0$; **right**: $\theta = \pi/6$.

Figure 5 shows the solutions using the PINN models. The first row corresponds to the results of the PINN-MSE model and the second row the PINN-Mono model. The first column shows the pressure prediction, the second column the absolute difference between PINN and TPFA solutions, and the last column plots the absolute PDE residual distributions computed at the testing points at the end of training. We can see that, overall, both PINN models perform quite well compared to the TPFA method, with the PINN-Mono model being slightly more accurate than the PINN-MSE model. It is interesting to note that for the two PINN models, the largest PDE residuals are located close to the center of the computational domain where the absolute error of the solution is located; however, it is not the largest, suggesting that there is no direct connection between solution accuracy distribution and the PDE residual distribution. Table 1 lists the minimum of solution $u_{\min}$, the maximum of solution, $u_{\max}$ and the relative L^2 norm of solution errors for the two PINN models, as well as the training time. The training time can be thought of as the time spent on approximating the unknown function using neural network models, and it is mainly affected by the number of residual points used for computing the loss and the

number of iterations specified for training. The PINN models cannot guarantee the non-negativity of the pressure solutions. As expected, adding the monotonicity penalty term in the loss function can decrease the magnitude of violations of pressure non-negativity. Moreover, the relative L^2 norm of the PINN-Mono solution is also smaller compared to that of the PINN-MSE model, while the training time is comparable. The results show that incorporating more physics by adding the monotonicity penalty term in the loss function is indeed beneficial for the PINN solution with a negligible increase in computational expenses in terms of training. For comparison, we also show here the solution results of PINN models using the ReLU, Softplus and Square functions as the activation functions for the last layer in Figures 6 and 7 and Table 1. The corresponding models are denoted by PINN-ReLU, PINN-Softplus and PINN-Square, respectively. We can see that although the three PINN models can preserve solution positivity, their relative L^2 error is almost two orders of magnitude higher than the PINN-MSE and PINN-Mono models with the exact same configurations. Therefore, we will not use these positivity-preserving PINN models in the following.

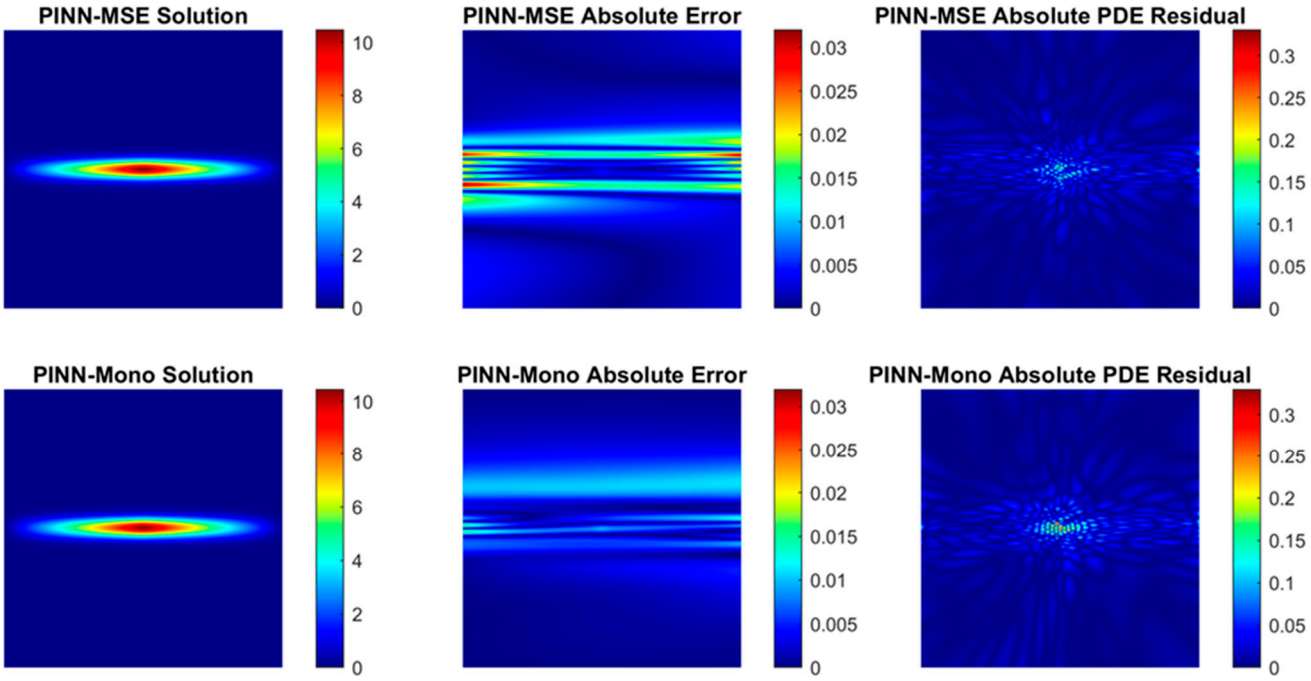

Figure 5. PINN solutions for case 1: $\theta = 0$. The first row corresponds to the results of the PINN-MSE model and the second row the PINN-Mono model.

Table 1. Summary of computational results for case 1: $\theta = 0$.

	$u_{\min}$	$u_{\max}$	Relative L^2	Training Time (s)
TPFA	1.5075e-22	1.0457e+01	\	\
PINN-MSE	−1.8819e-02	1.0454e+01	4.7659e-03	1051
PINN-Mono	−1.4453e-03	1.0449e+01	3.2576e-03	1059
PINN-ReLU	0	9.6432e+00	2.4135e-01	1127
PINN-Softplus	6.5235e-10	1.0820e+01	7.0749e-02	1173
PINN-Square	8.1448e-16	1.1019e+01	1.0677e-01	1164

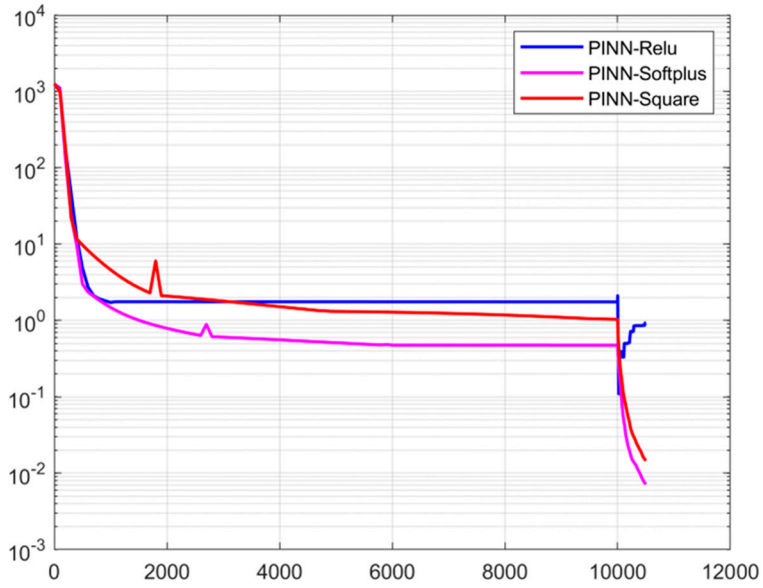

Figure 6. Loss history of the three PINN models that are positivity-preserving.

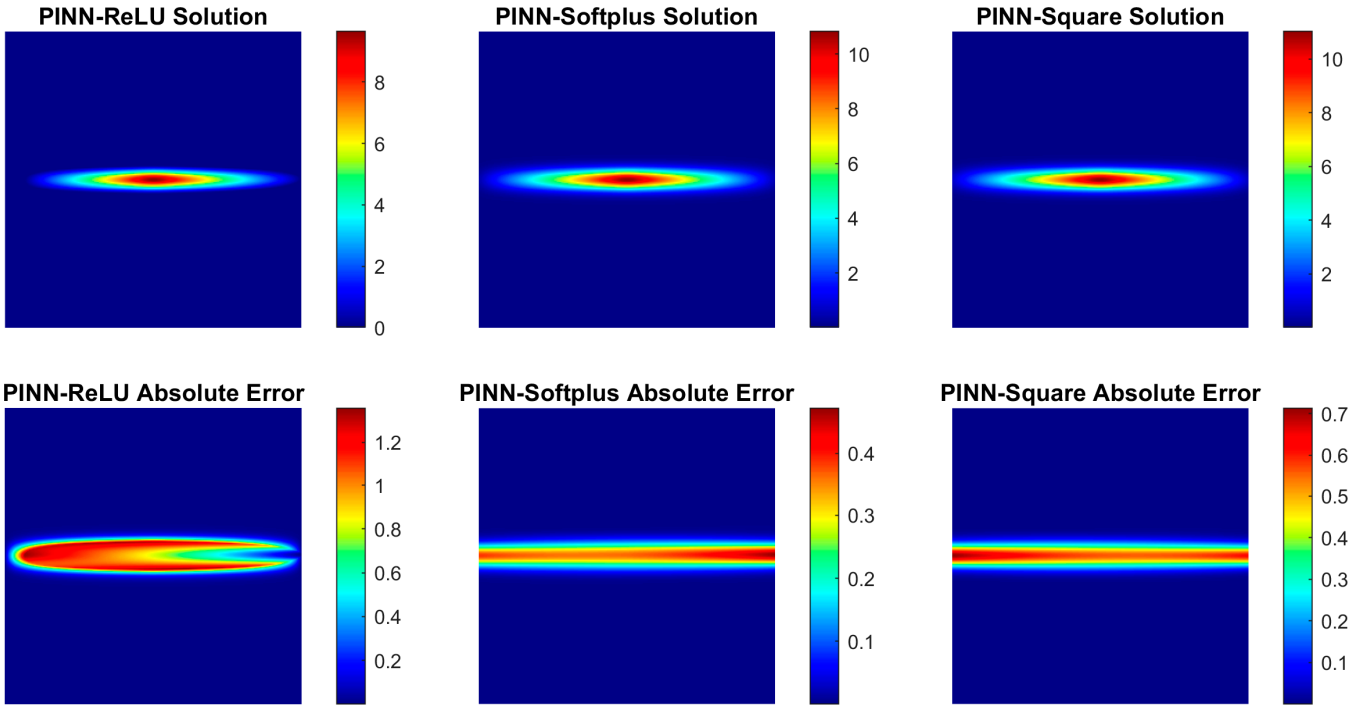

Figure 7. Solution results of the three PINN models that are positivity-preserving.

Next, we change the value of α from 0 to $\frac{\pi}{6}$. Since the principal directions of the permeability tensor are not aligned with the axes, the TPFA method is no longer applicable. The PINN models, on the other hand, are not restricted since they are meshless. The right plot of Figure 4 shows the loss history of the two PINN models and the corresponding solution results are shown in Figure 8. As a comparison, we also include in the figure the pressure solutions obtained from the MFD method and NTPFA method that were implemented in the open-source MATLAB Reservoir Simulation Toolbox (MRST) [40]. The results show that both the MFD, NTPFA method and the PINN models can capture the tensorial effect of the permeability tensor with the pressure solutions dictated by the principal directions of the permeability tensor, and the two PINN model solutions are in close agreement with that of the MFD method, while the solution of the NTPFA method has

a more diffused profile around the source term. Moreover, the minimum and maximum of all the pressure solutions and the training time for the PINN models are listed in Table 2, and we can see that the maximum values of the MFD and PINN solutions are also very close and quite far from that of the NPTFA solution, suggesting that the MFD and PINN solutions may be more reliable compared to the NTFA solution. On the other hand, both the MFD and PINN solutions violate the solution non-negativity condition and are, therefore, non-monotone. As expected, the minimum of the PINN-Mono solution is closer to 0 than that of the PINN-MSE model.

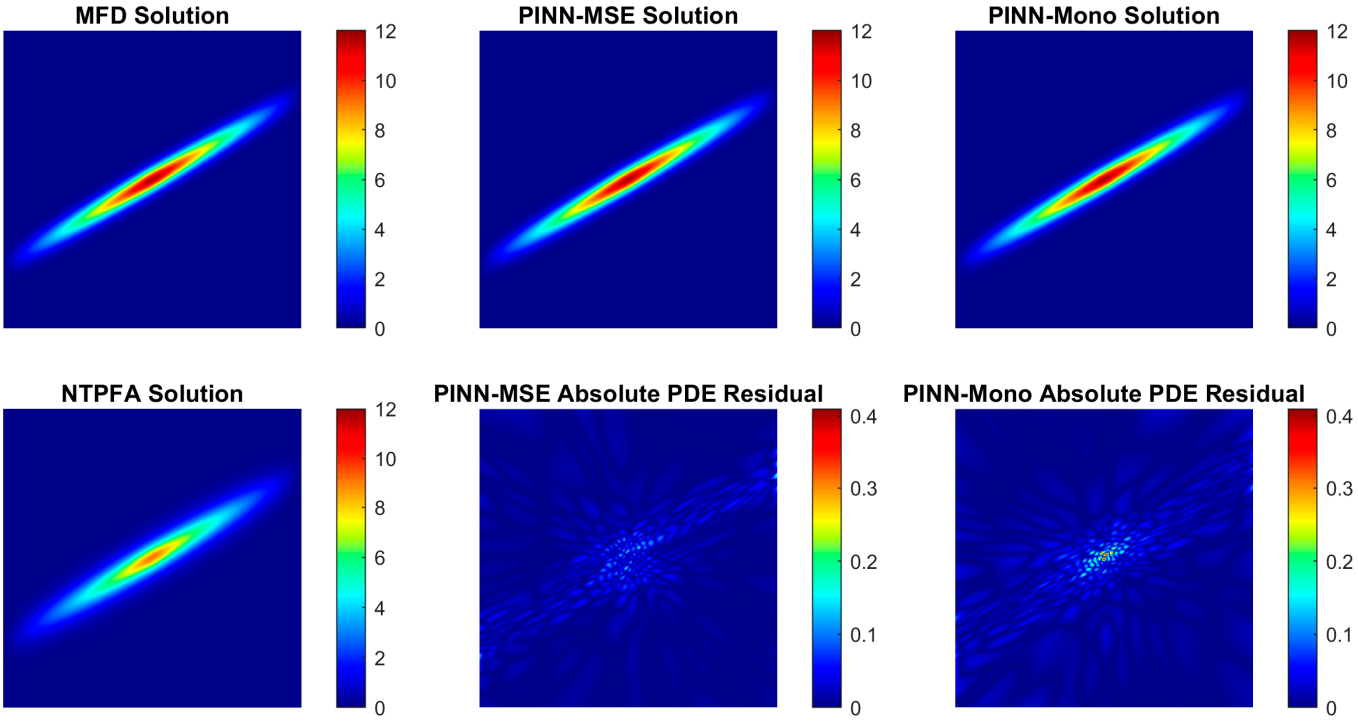

Figure 8. Results of case 1: $\theta = \frac{\pi}{6}$. The first column shows the solution of MFD and NTPFA. The second and third columns show the solution and absolute PDE residual of PINN-MSE and PINN-Mono, respectively.

Table 2. Summary of computational results for case 1: $\theta = \frac{\pi}{6}$.

	$u_{\min}$	$u_{\max}$	Training Time (s)
NTPFA	1.4180e-19	9.0828e+00	\
MFD	−9.2764e-33	1.1730e+01	\
PINN-MSE	−1.2959e-02	1.1768e+01	1059
PINN-Mono	−8.6398e-05	1.1770e+01	1063

3.2. Case 2: Rotating Anisotropy Tensor

This second test case is similar to the previous one, except that the permeability tensor is a rotating anisotropic field that is adapted from the benchmark study of discretization schemes for anisotropic diffusion problems [41]:

$$\mathbf{K} = \begin{bmatrix} \varepsilon x^2 + y^2 & (\varepsilon - 1)xy \\ (\varepsilon - 1)xy & x^2 + \varepsilon y^2 \end{bmatrix} \tag{12}$$

where ε is a parameter that controls the strength of the permeability anisotropy. Figure 9 shows an example of the permeability field on an 8×8 Cartesian grid where the permeability tensor evaluated at the centroid of each cell is represented by an ellipse whose semi-axes

are scaled by the square root of the maximum and minimum principal values, respectively, and the rotation of the semi-axes follows the corresponding principal directions of the permeability tensor. The permeability anisotropy ratio k_r is a constant and it is related to the parameter ε by $k_r = \frac{1}{\varepsilon}$. We test two levels of anisotropy by letting the values of k_r be 10 (mild anisotropy) and 1000 (strong anisotropy), respectively. Figure 10 shows the loss history of the training for the two PINN models, and the corresponding results for the two levels of permeability anisotropy are shown in Figures 11 and 12, Tables 3 and 4, respectively. Note that the absolute PDE residuals in Figure 12 are in log-scale. The MFD and NTPFA methods are still used to solve the diffusion equation on a 256×256 Cartesian mesh for the purpose of comparison. Note that the MFD and NTPFA methods require that the permeability be piece-wise constant. In our implementation, we compute the permeability tensor for each cell by evaluating Equation (12) at the cell centroid. Therefore, the variation of the permeability tensor within each cell is ignored. The PINN models, on the other hand, can handle the permeability variation without any difficulty. For the sub-case of mild anisotropy ($k_r = 10$), the solutions of the MFD and NTPFA methods are quite close to that of both PINN models. Mild anisotropy generally poses less of a challenge to numerical discretization schemes than high anisotropy problems. Therefore, the numerical solutions are reliable when the anisotropy is not too strong. For the case of strong anisotropy ($k_r = 1000$), the solutions of the PINN models are again much closer to the MFD solution than the NTPFA solution. The PINN models cannot enforce absolute non-negative pressure solutions, but the PINN-Mono model does a better job in reducing the degree of violations as opposed to the PINN-MSE model, especially for the case of strong anisotropy.

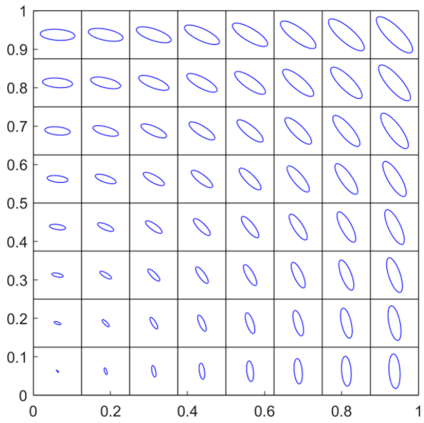

Figure 9. Rotating anisotropic permeability tensor field.

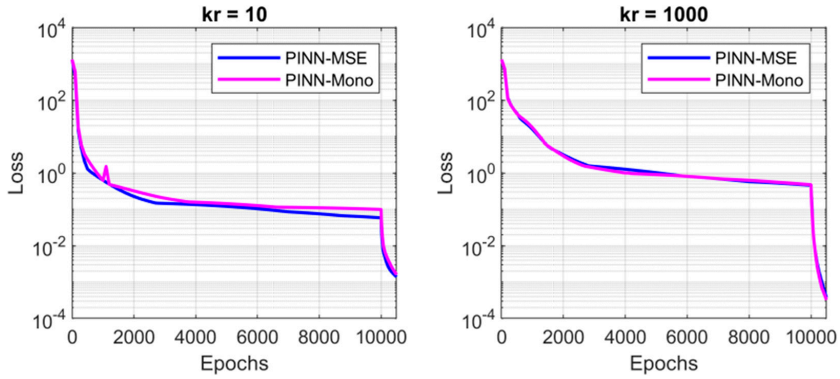

Figure 10. Loss history of PINN for case 2. **Left**: $k_r = 10$; **right**: $k_r = 1000$.

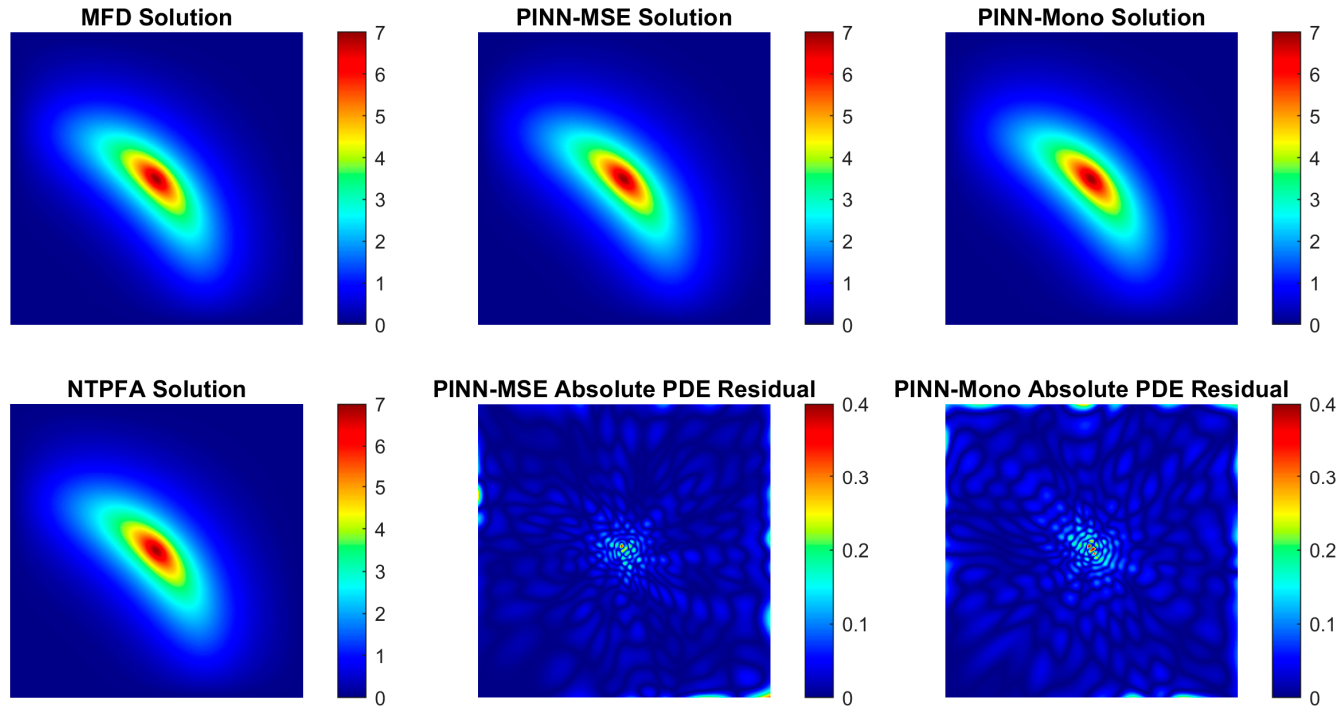

Figure 11. Results of case 2: $k_r = 10$.

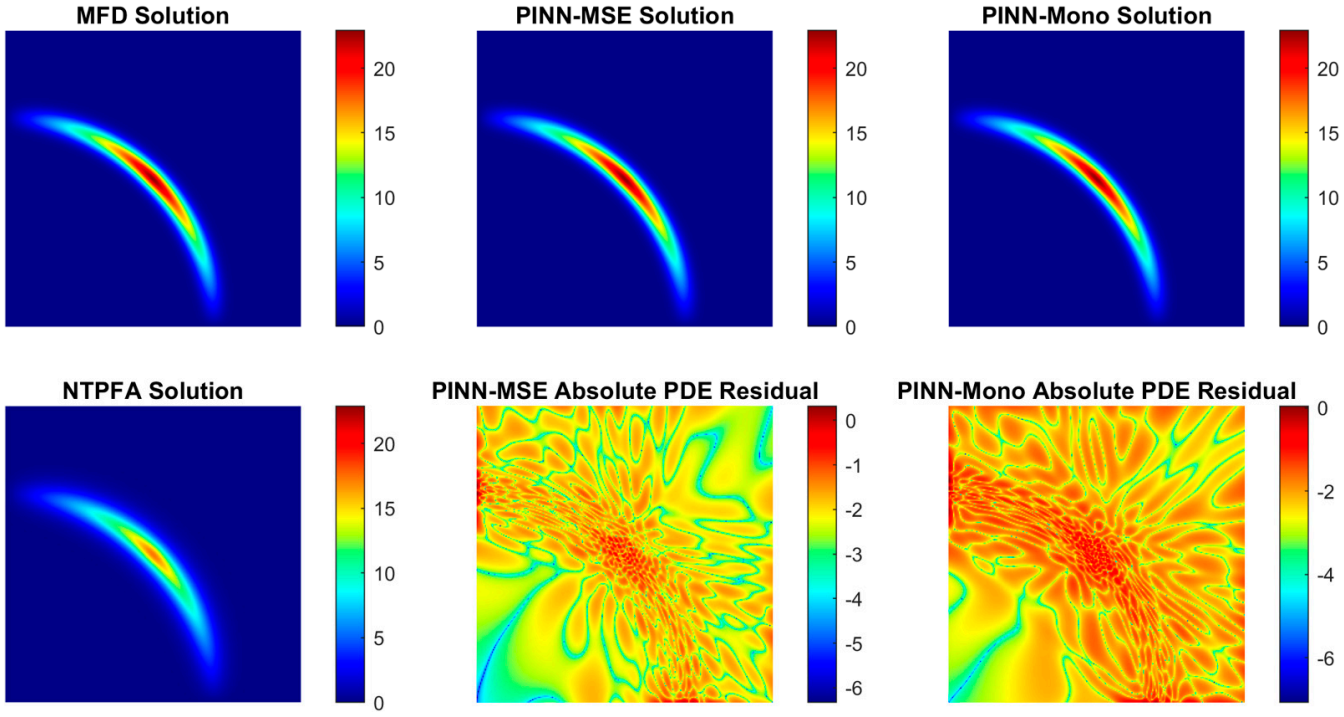

Figure 12. Results of case 2: $k_r = 1000$. The absolute PDE residuals shown in the second row are in log-scale.

Table 3. Summary of computational results for case 2: $k_r = 10$.

	$u_{\min}$	$u_{\max}$	Training Time (s)
NTPFA	1.1647e-10	6.9191e+00	\
MFD	1.1663e-12	6.9438e+00	\
PINN-MSE	−8.7351e-02	6.9560e+00	1075
PINN-Mono	−3.6969e-02	6.9587e+00	1090

Table 4. Summary of computational results for case 2: $k_r = 1000$.

	$u_{\min}$	$u_{\max}$	Training Time (s)
NTPFA	1.4398e-23	1.7269e+01	\
MFD	−1.1511e-18	2.2738e+01	\
PINN-MSE	−3.5002e-02	2.2832e+01	1101
PINN-Mono	−6.5714e-03	2.28333e+01	1095

3.3. Case 3: Discontinuous Permeability Tensor

This final test case is adapted from [13] and deals with a discontinuous permeability field. As depicted in Figure 13, the computational domain Ω is divided into four quadrants. The permeability tensor of the north-east and south-west quadrants is $\mathbf{K}_1$, and the other two quadrants $\mathbf{K}_2$. The principal directions of $\mathbf{K}_1$ and $\mathbf{K}_2$ are rotated by different angles α_1 and α_2, respectively. Therefore, the permeability field is discontinuous across the line $x = 0.5$ and $y = 0.5$. To deal with this discontinuous permeability field, we adopt similar ideas from [26] and use two different neural networks for the two different permeability tensors. More specifically, our model consists of two neural networks, the first one covering the north-east and south-west quadrants and a second one covering the remaining two quadrants. Pressure and flux continuity conditions on the discontinuity lines are used to stitch the two distinct neural networks into a single model. More specifically, a loss term resulting from the pressure and flux continuity conditions is added to the loss function. The two neural networks are denoted by $\hat{u}_1(\mathbf{x};\theta_1)$ and $\hat{u}_2(\mathbf{x};\theta_2)$. The single loss function for the model then takes the following form:

$$\mathcal{L}(\theta) = \omega_r \mathcal{L}_r(\theta) + \omega_b \mathcal{L}_b(\theta) + \omega_d \mathcal{L}_d(\theta) \tag{13}$$

where the loss terms $\mathcal{L}_r(\theta)$ and $\mathcal{L}_b(\theta)$ are calculated in a similar fashion to Equations (6) and (7), respectively, with the difference being that the residual loss and boundary loss are calculated using $\hat{u}_1(\mathbf{x};\theta_1)$ if the training points lie in the north-east and south-west quadrants. Otherwise, the network $\hat{u}_2(\mathbf{x};\theta_2)$ is used instead. The last term in the loss function accounts for the pressure and flux continuity conditions on the permeability discontinuity lines, and it is given by:

$$\mathcal{L}_d(\theta) = \frac{1}{N_d} \begin{pmatrix} \sum_{i=1}^{N_d} \left| \hat{u}_1\left(\mathbf{x}_i^d;\theta_1\right) - \hat{u}_2\left(\mathbf{x}_i^d;\theta_1\right) \right|^2 \\ + \\ \sum_{i=1}^{N_d} \left| \mathbf{K}_1 \nabla \hat{u}_1\left(\mathbf{x}_i^d;\theta_1\right) \cdot \mathbf{n}_i - \mathbf{K}_2 \nabla \hat{u}_2\left(\mathbf{x}_i^d;\theta_2\right) \cdot \mathbf{n}_i \right|^2 \end{pmatrix} \tag{14}$$

where $\mathbf{x}_i^d$ is the training point on the discontinuity line and N_d is the total number of these training points; $\mathbf{n}_i$ is the normal vector to the discontinuity line at point $\mathbf{x}_i^d$. ω_d is the weighting parameter.

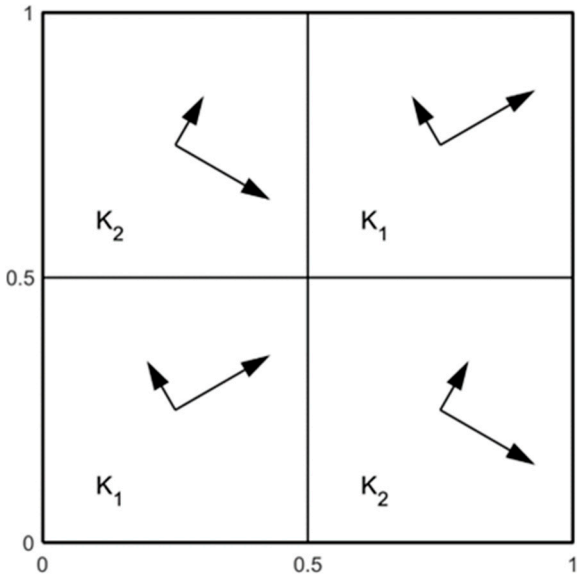

Figure 13. Discontinuous permeability field.

Similarly, as before, we impose the monotonicity constraint by adding one additional loss term:

$$\mathcal{L}(\theta) = \omega_r \mathcal{L}_r(\theta) + \omega_b \mathcal{L}_b(\theta) + \omega_d \mathcal{L}_d(\theta) + \omega_m \mathcal{L}_m(\theta) \tag{15}$$

where the loss term $\mathcal{L}_m(\theta)$ is computed analogous to Equation (9), with the single neural network replaced by two neural networks. For ease of notation, we again call the PINN model using loss function (13) PINN-MSE and the model using loss function (15) PINN-Mono in the following.

A homogeneous Dirichlet boundary condition is applied on the boundary and the source term is given by Equation (11) with $\sigma = 0.1$. To evaluate the accuracy of our PINN models, we first set $\alpha_1 = 0$ and $\alpha_2 = \frac{\pi}{2}$ so that we can solve the problem using the linear TPFA method on a 256×256 Cartesian mesh as our reference solution. Next, we set $\alpha_1 = \frac{\pi}{6}$ and $\alpha_2 = -\frac{\pi}{6}$ and solve the problem numerically using the MFD and NTPFA methods for comparison. For the PINN models, we set $N_r = 10,000$, $N_b = 1000$, and $N_d = 2000$. The residual points follow the Hammersley distribution, while the training points on the boundary and permeability discontinuity line are evenly spaced. The weighting parameters in the loss function are set as $\omega_r = \omega_b = 1$ and $\omega_d = \omega_m = 50$. The PINN models are trained using the Adam optimizer for 10,000 iterations and then trained using the L-BFGS optimizer for 2500 iterations. The loss history of the two PINN models is shown in Figure 14 and the corresponding solution results are shown in Figures 15 and 16 and Tables 5 and 6, respectively. Compared to the solution of TPFA, the two PINN models have similar accuracy in terms of the relative L^2 error, although a visual inspection seem to suggest that the solution of the PINN-Mono model is slightly better than that of the PINN-MSE model. The minimum values for both PINN model solutions are negative and violate the monotonicity requirement, with the degree of violations much smaller in the PINN-Mono model thanks to the penalty term $\mathcal{L}_m(\theta)$ in its loss function. For the sub-case of the $\alpha_1 = \frac{\pi}{6}$ and $\alpha_2 = -\frac{\pi}{6}$, solutions of the MFD, the NTPFA methods and the two PINN models reflect the effect of the rotation of the permeability principal directions. The major difference in the solutions lies in the center of the domain where the solutions of the PINN models are higher than that of the MFD and NTPFA methods.

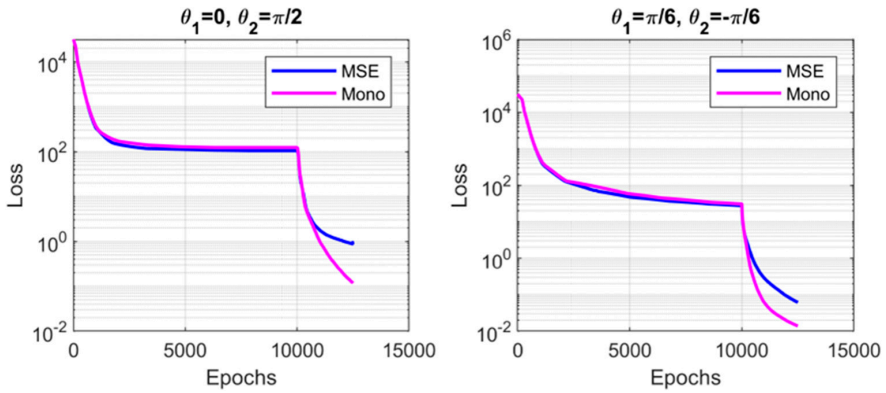

Figure 14. Loss history of PINN for case 3.

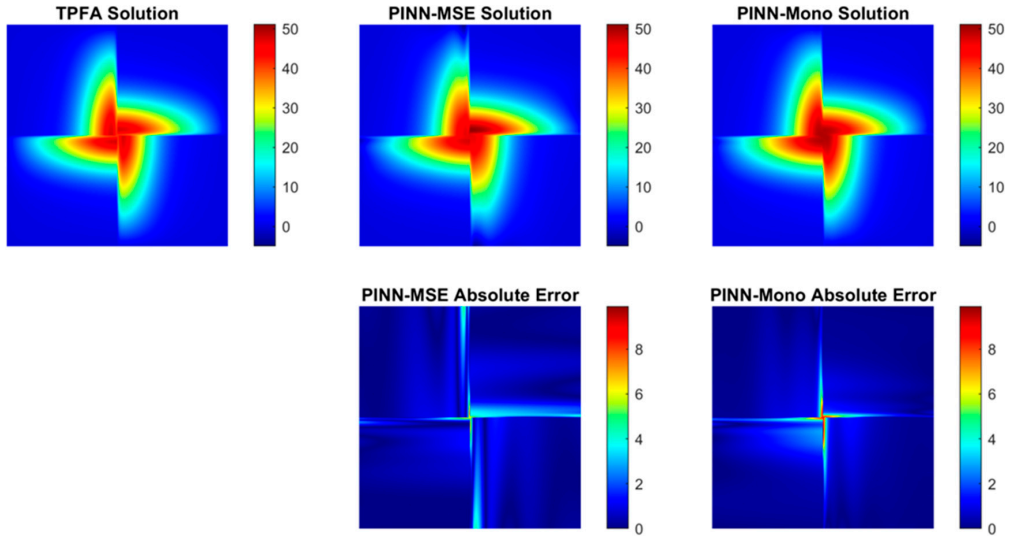

Figure 15. Results for case 3: $\alpha_1 = 0,\ \alpha_2 = \frac{\pi}{2}$.

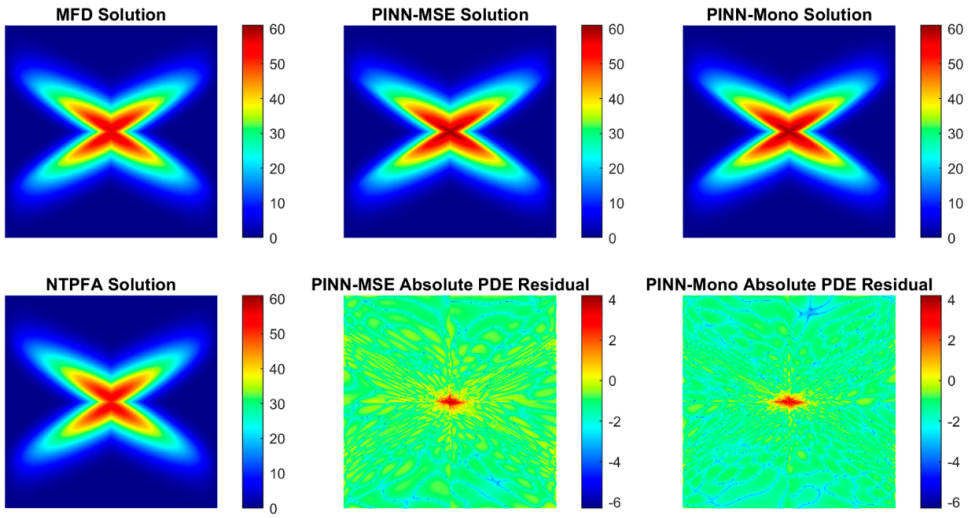

Figure 16. Results for case 3: $\alpha_1 = \frac{\pi}{6},\ \alpha_2 = -\frac{\pi}{6}$. Absolute PDE residuals in the second row are in log-scale.

Table 5. Summary of computational results for case 3: $\alpha_1 = 0$, $\alpha_2 = \frac{\pi}{2}$.

	$u_{\min}$	$u_{\max}$	Relative L^2	Training Time (s)
TPFA	5.4846e-5	4.7339e+01	\	\
PINN-MSE	−4.5730e0	5.0947e+01	6.1622e-02	7186
PINN-Mono	−7.9776e-01	4.9551e+01	6.1676e-02	7535

Table 6. Summary of computational results for case 3: $\alpha_1 = \frac{\pi}{6}$, $\alpha_2 = -\frac{\pi}{6}$.

	$u_{\min}$	$u_{\max}$	Training Time (s)
NTPFA	1.1138e-07	5.4074e+01	\
MFD	6.7068e-08	5.6624e+01	\
PINN-MSE	−2.3159e-01	6.0775e+01	6025
PINN-Mono	−9.2147e-02	6.0198e+01	6214

4. Conclusions

Physics-informed neural network models are developed in this work to solve the highly anisotropic diffusion equation with a focus on improving the monotonicity properties of the solutions. An additional loss term is included in the loss function to account for the monotonicity requirement. The results of the numerical experiments show that the resulting PINN model can reduce the degree of monotonicity violations significantly. When the mesh is K-orthogonal and the linear TPFA method is applicable, we can evaluate the accuracy of the PINN solutions against the TPFA solutions. The results show that the new PINN model has comparable or slightly better accuracy compared to the vanilla PINN model. The PINN model can accommodate both homogeneous and heterogeneous permeability tensors, and we further extended the model to account for permeability discontinuity.

A seemingly obvious option to impose solution monotonicity for the PINN model is to use some positivity-preserving activation function in the last layer of the neural network. However, our numerical testing shows that the solution accuracy is rather poor compared to the PINN models developed in this work. The reason is hitherto unclear to us and needs further investigation. Our work here is a prelude to a pressing front that we are actively working on, which is how to develop PINN models for permeability tensors that are discontinuous cell-wise—that is, the permeability tensor is constant in each cell but discontinuous across cell interfaces, as is commonly encountered in the geoscience industry.

Author Contributions: Conceptualization, W.Z. and M.A.K.; methodology, W.Z. and M.A.K.; software, W.Z.; validation, W.Z.; formal analysis, W.Z. and M.A.K.; investigation, W.Z.; data curation, W.Z.; writing—original draft preparation, W.Z.; writing—review and editing, W.Z. and M.A.K.; visualization, W.Z.; supervision, M.A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

Nomenclature

v fluid velocity [LT^{-1}]
u fluid pressure/potential [$ML^{-1}T^{-2}$]
K permeability tensor [L^2]
q fluid source/sink [M^3T^{-1}]

References

1. Aziz, K.; Aziz, K.; Settari, A. *Petroleum Reservoir Simulation*; Applied Science Publishers: London, UK, 1979.
2. Aarnes, J.E.; Gimse, T.; Lie, K.-A. An Introduction to the Numerics of Flow in Porous Media using Matlab. In *Geometric Modelling, Numerical Simulation, and Optimization: Applied Mathematics at SINTEF*; Hasle, G., Lie, K.-A., Quak, E., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 265–306.
3. Aavatsmark, I.; Barkve, T.; Bøe, Ø.; Mannseth, T. Discretization on Non-Orthogonal, Quadrilateral Grids for Inhomogeneous, Anisotropic Media. *J. Comput. Phys.* **1996**, *127*, 2–14. [CrossRef]
4. Lee, S.H.; Durlofsky, L.J.; Lough, M.F.; Chen, W.H. Finite Difference Simulation of Geologically Complex Reservoirs with Tensor Permeabilities. *SPE Reserv. Eval. Eng.* **1998**, *1*, 567–574. [CrossRef]
5. Edwards, M.G.; Rogers, C.F. Finite volume discretization with imposed flux continuity for the general tensor pressure equation. *Comput. Geosci.* **1998**, *2*, 259–290. [CrossRef]
6. Aavatsmark, I. An Introduction to Multipoint Flux Approximations for Quadrilateral Grids. *Comput. Geosci.* **2002**, *6*, 405–432. [CrossRef]
7. Nordbotten, J.M.; Aavatsmark, I. Monotonicity conditions for control volume methods on uniform parallelogram grids in homogeneous media. *Comput. Geosci.* **2005**, *9*, 61–72. [CrossRef]
8. Mlacnik, M.J.; Durlofsky, L.J. Unstructured grid optimization for improved monotonicity of discrete solutions of elliptic equations with highly anisotropic coefficients. *J. Comput. Phys.* **2006**, *216*, 337–361. [CrossRef]
9. Chen, Q.-Y.; Wan, J.; Yang, Y.; Mifflin, R.T. Enriched multi-point flux approximation for general grids. *J. Comput. Phys.* **2008**, *227*, 1701–1721. [CrossRef]
10. Edwards, M.G.; Zheng, H. A quasi-positive family of continuous Darcy-flux finite-volume schemes with full pressure support. *J. Comput. Phys.* **2008**, *227*, 9333–9364. [CrossRef]
11. Keilegavlen, E.; Aavatsmark, I. Monotonicity for MPFA methods on triangular grids. *Comput. Geosci.* **2011**, *15*, 3–16. [CrossRef]
12. Zhang, W.; Al Kobaisi, M. A simplified enhanced MPFA formulation for the elliptic equation on general grids. *Comput. Geosci.* **2017**, *21*, 621–643. [CrossRef]
13. Lipnikov, K.; Shashkov, M.; Svyatskiy, D.; Vassilevski, Y. Monotone finite volume schemes for diffusion equations on unstructured triangular and shape-regular polygonal meshes. *J. Comput. Phys.* **2007**, *227*, 492–512. [CrossRef]
14. Yuan, G.; Sheng, Z. Monotone finite volume schemes for diffusion equations on polygonal meshes. *J. Comput. Phys.* **2008**, *227*, 6288–6312. [CrossRef]
15. Sheng, Z.; Yuan, G. The finite volume scheme preserving extremum principle for diffusion equations on polygonal meshes. *J. Comput. Phys.* **2011**, *230*, 2588–2604. [CrossRef]
16. Gao, Z.; Wu, J. A small stencil and extremum-preserving scheme for anisotropic diffusion problems on arbitrary 2D and 3D meshes. *J. Comput. Phys.* **2013**, *250*, 308–331. [CrossRef]
17. Wu, J.; Gao, Z. Interpolation-based second-order monotone finite volume schemes for anisotropic diffusion equations on general grids. *J. Comput. Phys.* **2014**, *275*, 569–588. [CrossRef]
18. Schneider, M.; Flemisch, B.; Helmig, R.; Terekhov, K.; Tchelepi, H. Monotone nonlinear finite-volume method for challenging grids. *Comput. Geosci.* **2018**, *22*, 565–586. [CrossRef]
19. Zhang, W.; Al Kobaisi, M. Cell-Centered Nonlinear Finite-Volume Methods with Improved Robustness. *SPE J.* **2020**, *25*, 288–309. [CrossRef]
20. Brezzi, F.; Fortin, M. *Mixed and Hybrid Finite Element Methods*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; Volume 15.
21. da Veiga, L.B.; Lipnikov, K.; Manzini, G. *The Mimetic Finite Difference method for Elliptic Problems*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 11.
22. 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]
23. Huang, T.-H.; Wei, H.; Chen, J.-S.; Hillman, M.C. RKPM2D: An open-source implementation of nodally integrated reproducing kernel particle method for solving partial differential equations. *Comput. Part. Mech.* **2020**, *7*, 393–433. [CrossRef]
24. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
25. Yang, Y.; Perdikaris, P. Adversarial uncertainty quantification in physics-informed neural networks. *J. Comput. Phys.* **2019**, *394*, 136–152. [CrossRef]
26. Jagtap, A.D.; Kawaguchi, K.; Karniadakis, G.E. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *J. Comput. Phys.* **2020**, *404*, 109136. [CrossRef]
27. Krishnapriyan, A.; Gholami, A.; Zhe, S.; Kirby, R.; Mahoney, M.W. Characterizing possible failure modes in physics-informed neural networks. *Adv. Neural Inf. Processing Syst.* **2021**, *34*, 26548–26560.
28. 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]
29. Yang, L.; Meng, X.; Karniadakis, G.E. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *J. Comput. Phys.* **2021**, *425*, 109913. [CrossRef]

30. 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]
31. Mao, Z.; Jagtap, A.D.; Karniadakis, G.E. Physics-informed neural networks for high-speed flows. *Comput. Methods Appl. Mech. Eng.* **2020**, *360*, 112789. [CrossRef]
32. Misyris, G.S.; Venzke, A.; Chatzivasileiadis, S. Physics-Informed Neural Networks for Power Systems. In Proceedings of the 2020 IEEE Power & Energy Society General Meeting (PESGM), Montreal, QC, Canada, 2–6 August 2020; pp. 1–5.
33. Yang, X.; Zafar, S.; Wang, J.-X.; Xiao, H. Predictive large-eddy-simulation wall modeling via physics-informed neural networks. *Phys. Rev. Fluids* **2019**, *4*, 034602. [CrossRef]
34. Chen, Y.; Lu, L.; Karniadakis, G.E.; Dal Negro, L. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Opt. Express* **2020**, *28*, 11618–11633. [CrossRef] [PubMed]
35. Fuks, O.; Tchelepi, H.A. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *J. Mach. Learn. Modeling Comput.* **2020**, *1*. [CrossRef]
36. Zhang, W.; Diab, W.; Al Kobaisi, M. *Physics Informed Neural Networks for Solving Highly Anisotropic Diffusion Equations*; European Association of Geoscientists & Engineers: Houten, The Netherlands, 2022; pp. 1–15. [CrossRef]
37. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS* **2019**, *32*, 8026–8037.
38. Das, S.; Tesfamariam, S. State-of-the-Art Review of Design of Experiments for Physics-Informed Deep Learning. *arXiv* **2022**, arXiv:2202.06416.
39. Peng, W.; Zhou, W.; Zhang, X.; Yao, W.; Liu, Z. A Residual-based Adaptive Node Generation Method for Physics-Informed Neural Networks. *arXiv* **2022**, arXiv:2205.01051.
40. Lie, K.-A. *An Introduction to Reservoir Simulation Using MATLAB/GNU Octave: User Guide for the MATLAB Reservoir Simulation Toolbox (MRST)*; Cambridge University Press: Cambridge, UK, 2019. [CrossRef]
41. Herbin, R.; Hubert, F. Benchmark on Discretization Schemes for Anisotropic Diffusion Problems on General Grids. In *Finite Volumes for Complex Applications V*; Wiley: Hoboken, NJ, USA, 2008.