Journal of Computational Physics 470 (2022) 111539

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

NH-PINN: Neural homogenization-based physics-informed neural network for multiscale problems

Wing Tat Leung [a], Guang Lin [b,c], Zecheng Zhang [b]

[a] Department of Mathematics, City University of Hong Kong, Hong Kong, China
[b] Department of Mathematics, Purdue University, 610 Purdue Mall, West Lafayette, 47907, IN, USA
[c] Department of Mechanical Engineering, Purdue University, 610 Purdue Mall, West Lafayette, 47907, IN, USA

ARTICLE INFO

Article history:
Received 14 March 2022
Received in revised form 17 July 2022
Accepted 10 August 2022
Available online 27 August 2022

Keywords:
Multiscale partial differential equation
Physics-informed neural network
Homogenization

ABSTRACT

Physics-informed neural network (PINN) is a data-driven approach to solving equations. It is successful in many applications; however, the accuracy of the PINN is not satisfactory when it is used to solve multiscale equations. Homogenization approximates a multiscale equation by a homogenized equation without multiscale property; it includes solving cell problems and the homogenized equation. The cell problems are periodic, and we propose an oversampling strategy that significantly improves the PINN accuracy on periodic problems. The homogenized equation has a constant or slow dependency coefficient and can also be solved by PINN accurately. We hence proposed a 3-step method, neural homogenization based PINN (NH-PINN), to improve the PINN accuracy for solving multiscale problems with the help of homogenization.

1. Introduction

In the recent decade, the problems of having features at multiple scales have been of rising importance. For example, to obtain a more accurate model in biomedical applications, people need to incorporate cell-level information into the tissue-level model. In material application, people are interested in the macroscopic properties of some complex composite materials which are composited on a microscopic scale. In petroleum applications [20,18,9,14], the fluid flow in the reservoir is heavily dependent on the heterogeneous porous media. It is not easy to solve the equations by classical methods. To capture the multiscale properties, one needs to use a fine mesh [3,7,12,5,2]; alternatively, one can use the multiscale finite element methods which can save the computation cost, but it is still not easy for some problems. Hence, researchers also studied the data-driven approaches to solve multiscale problems.

Physics-informed neural network (PINN) is a neural network approach to solving partial differential equations (PDE) [22,17,16]. The idea of the PINN is to approximate the solution of the PDE by a network. The PINN has been widely used in solving both forward and inverse problems [25,16]. Compared to the classical numerical methods, PINN is a mesh-free method and hence can interpret the solution (predict solution at any point in the domain) without a mesh. Moreover, PINN has no CFL constraint and is easy to be used to solve time-dependent problems. For the equation with a convection term, PINN is also powerful and one does not need to consider the directions of the flow, which reduces the difficulty. Besides, PINN can be used to solve the equations with the mixed direction derivatives, such as paraxial approximation [5]. The traditional numerical methods are usually hard to implement and involve intense computation. Compared to other neural

E-mail addresses: sidnet123@gmail.com (W.T. Leung), guanglin@purdue.edu (G. Lin), zecheng.zhang.math@gmail.com (Z. Zhang).

https://doi.org/10.1016/j.jcp.2022.111539
0021-9991/

network methods of solving the PDE [26,6], PINN is a sample-free method that does not rely on labels, we hence can use PINN to solve problems not limited to UQ.

Although researchers have applied PINN to solve many mathematical problems, there are only a few works about multiscale problems. In [24], the authors point out that the classical PINN formulation is unable to capture the multiscale property of the solution; they later proposed a method that applies the Fourier transformation to prepossess the input to solve the problem. Their interest in the multiscale is in the solution; this is different from many existing real-life multiscale problems we have discussed before [15,10,26,8,4]. In this work, we mainly focus on the multiscale problems whose multiscale property comes from the equation. One typical example is the 2D elliptic equation, which models the porous media flow. The permeability of the equation has a multiscale nature and brings multiscale to the solution; for example, the permeability $\kappa(x) = \sin(x) + \sin(10x)$.

We hence seek help from the PINN method; however, we find that the classical PINN cannot give us an accurate prediction. The relative error is enormous, and the training is not robust to the hyperparameters and randomness. More precisely, the relative error varies for different runs even if we keep all hyperparameters the same. To alleviate the problems of the classical PINN and take advantage of the PINN, we propose a homogenization-based approach.

Homogenization is one of the classic ways of solving an equation with a highly heterogeneous medium [1,13,19,23]. Basically, throughout the homogenization process, we will obtain a homogenized equation of the original equation such that the solution of this homogenized equation can capture the macroscopic behavior of the solution of the original equation. The parameters of the homogenized equation are normally homogeneous or smooth, such that the equation can be solved numerically in a coarse mesh. For example, we consider a diffusion equation with a periodic media κ:

$$-\nabla \cdot (\kappa(\frac{x}{\epsilon})\nabla u_\epsilon) = f. \tag{1}$$

By homogenization theory, we can obtain a homogenized equation:

$$-\nabla \cdot (\kappa^* \nabla u_0) = f, \tag{2}$$

where κ^* is the homogenized parameter which is a constant tensor, and we have u_0 is converging to u_ϵ in L_2-norm as $\epsilon \to 0$. Thus, one can solve the homogenized equation in a coarse grid and use the homogenized solution u_0 to approximate the solution of the original equation u_ϵ.

We now propose to solve the multiscale problems by PINN with the help of homogenization [11]. We find that PINN can implement homogenization very naturally. Classical homogenization consists of three steps. The first step is to solve the cell problems on a unit cube. The cell problems are equipped with a periodic boundary condition, and the permeability has no fast dependency. It should be noted that we find the periodic PDE is not easily solved by PINN when the dimension is high; however, we propose an oversampling technique that significantly improves the performance of solving periodic problems. In all, cell problems can be solved by the PINN method accurately and efficiently.

The second step is to evaluate the homogenized permeability with no multiscale property and then obtain the homogenized equations. This step usually involves finding the derivatives of the cell problem solutions. If the solutions are approximated by a network, the derivatives can be easily derived by the auto differentiation of the software. Once the cell problems are solved by the network accurately, we can anticipate getting an accurate homogenized equation.

Lastly, we can solve the homogenized equation. The permeability associated with this equation is usually constant or depends on the slow variable only, the PINN has shown its power in solving such kinds of equations. We conclude that the PINN can be used to implement the entire homogenization process; one can solve the multiscale PDEs by PINN with the help of homogenization. Compared to numerical homogenization, we can benefit from the data-driven method, and we review some advantages here.

1. Data-driven method is very accurate and efficient in solving equations without multiscale features. We find that the cell problems and the homogenized equations can be solved with only a few training points (around 400 in most $2D$ examples we tried).
2. One can utilize the auto-differentiation tool, which is an easier way to obtain the homogenized equations once the cell problems are solved.
3. Our NH-PINN data-driven method is a mesh-free method, which gives us more flexibility to interpret the solutions and conduct other research based on the learned solution. One example is learning an operator that maps the coarse model to a fine model. This is a popular topic and to learn the operator, solutions at arbitrary locations are required, hence we need a mesh-free solver.

Finally, let us summarize the contributions of this work as follows:

1. We observe that the accuracy of the PINN degenerates when solving the multiscale PDE; we also explain the reason for the failure by applying the transfer learning. Please note that in this work, we focus on the PDE whose multiscale property originates from the equation coefficients.

2. We propose a 3-step data-free approach, neural homogenization-based PINN (NH-PINN). The classical PINN accuracy for solving multiscale problems is greatly improved.
3. We propose an oversampling strategy to solve the periodic PDE by PINN; this method greatly improves the accuracy of the PINN when solving the high dimensional periodic problems.
4. We conduct four numerical experiments which represent three different types of homogenization. The predictions are very accurate and are much better than the classical PINN, particularly when the scaling is small.
5. In addition to the improved accuracy of the PINN, we also observe that NH-PINN can improve homogenization accuracy. That is, if we apply PINN to implement homogenization, the solution may be more accurate than the traditional numerical methods (e.g., finite element methods) driven homogenization. We hence suggest that PINN may be a potential alternative to implementing homogenization.

The rest of the paper is organized as follows. In Section 2, we review the basics of the PINN; the performance of applying classical PINN to solve multiscale problems is also presented in this Section. Next, we will review the homogenization method in Section 3. We will give the details of one homogenization of the elliptic equations. Our numerical examples are not limited to the elliptic operator; we hence also present the homogenization of the reaction-diffusion equation in the Appendix A. We conduct four experiments, and they are shown in Section 5.

2. Physics-informed neural network (PINN)

In this section, we briefly review the physics-informed neural network (PINN) for solving the PDE; for the inversion problems, please refer to it [25,16] for details. PINN is a data-driven method to solve PDE. The network approximates the solution of the PDE, and the target is to minimize the error of this approximation. The PINN has been applied to solve various problems; however, it is rarely used in solving multiscale problems. Suppose we are solving the following system in the domain Ω,

$$\mathcal{L}(u) = f \text{ in } \Omega$$
$$\mathcal{B}(u) = b \text{ on } \partial\Omega$$

where $\mathcal{L}$ is a differential operator and $\mathcal{B}$ is the boundary condition operator. f is the given source term, and b is the given boundary condition. The idea of the PINN is to build a map from Ω to the solution by a network $\mathcal{F}_\beta(\cdot)$, where β is the parameters associated with the network. More precisely, we will solve the following minimization problem:

$$\min_\beta \frac{w_1}{N_f}\sum_{i=1}^{N_f} |\mathcal{L}(\mathcal{F}_\beta(p_i)) - f(p_i)|^2 + \frac{w_2}{N_b}\sum_{i=1}^{N_b} |\mathcal{B}(\mathcal{F}_\beta(q_i)) - b(q_i)|^2, \tag{3}$$

where $w_1 + w_2 = 1$ are the positive weights; $\{p_i\} \subset \Omega$, $\{q_i\} \subset \partial\Omega$ and N_f, N_b are the number of points used in discretizing the domain and boundary respectively. It should be noted that if the PDE has other constraints such as the initial condition $\mathcal{I}$, we just need to sample some points z_i, substitute the points in the network $\mathcal{F}_\beta(\cdot)$, then approximate the constraints by $\mathcal{I}(\mathcal{F}_\beta(\cdot))$ and include this loss $\mathcal{I}(\mathcal{F}_\beta(\cdot)) - I(\cdot)$ in Equation (3), where $I(\cdot)$ is the given initial conditions. PINN is easy to implement.

Since $\mathcal{F}_\beta$ approximates the solution and is smooth (neural network), the differential operator $\mathcal{L}(\mathcal{F}_\beta(\cdot))$ can be evaluated with the auto differentiation package of the modern deep learning software easily. The minimization problem can then be solved by the variants of the gradient descent algorithm.

2.1. Failure of the PINN

PINN has been used to solve various problems, however, the PINN performance in solving multiscale problems has been compromised. For example, let us consider the following elliptic equation:

$$-\nabla\cdot(\kappa\nabla u(x)) = f, x \in \Omega,$$
$$u = 0, x \in \partial\Omega,$$

where $\Omega = [0, \pi]$ and $f = \sin(x)$. This is a standard elliptic problem with a homogeneous Dirichlet boundary condition. If $\kappa(x)$ is a constant or has no multiscale property, the PINN performs very well; however if the permeability $\kappa(x)$ presents some multiscale property, for example, $\kappa(x) = 0.5\sin(2\pi x/\epsilon) + 2$, where $\epsilon = \frac{1}{8}$, we cannot obtain a satisfied solution (with a low relative error) using the classical PINN. We solve the above problem and demonstrate the results in Fig. 1. We will give more examples of the direct application of PINN in solving multiscale problems later in Section 5 and provide our homogenization strategy; that is, we propose to solve the multiscale problems in three steps with the help of homogenization. Our numerical experiments show that the accuracy of the PINN is greatly improved.

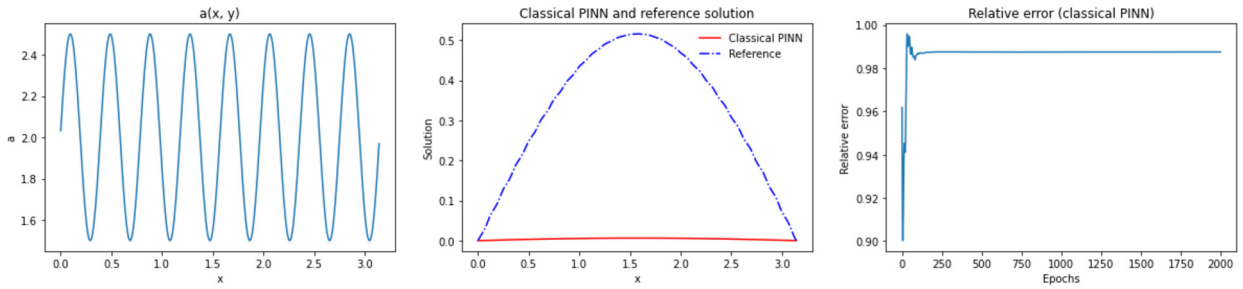

Fig. 1. 1D elliptic problem without slow dependency (classical PINN). Left: demonstration of the permeability $\kappa(x) = 0.5\sin(2\pi x/\epsilon) + 2$, where $\epsilon = 1/8$. Middle: learned solution by the classical PINN vs the reference solution. Right: relative error as a function of the training epochs. The average error of the last 500 epochs is 0.987471. In this example, learning rate is 0.0001 and $w_1 = w_2 = 0.5$ and we train the network with Adam gradient descent for 2000 epochs. We test with different sets of the hyperparameters such as the learning rate and the weights of the loss, however, we cannot obtain a satisfactory result.

2.2. Intuitive explanation of the failure

Now the question is why PINN fails in solving the multiscale equation. We give an explanation motivated by the finite element methods (FEM). For the standard FEM, it is well known that a coarse mesh FEM solution cannot resolve the multiscale solution. For example, in the 1-d case, the multiscale solution u_ϵ satisfying

$$-\partial_x\Big(a(\frac{x}{\epsilon})\partial_x u_\epsilon\Big) = f,$$

converging to the homogeneous solution u_0 satisfying

$$-\partial_x\Big(a^*(x)\partial_x u_0\Big) = f,$$

in L_2-norm when ϵ goes to 0 where a^* is the harmonic average of a [11]. However, if we use the linear finite element method in a coarse grid, we have $\partial_x\phi_i$ as a piecewise constant for any linear basis function ϕ_i. Thus, we have the finite element solution u_{FEM} satisfying

$$\int_\Omega a(\frac{x}{\epsilon})\partial_x u_{FEM}\partial_x\phi_i \approx \int_\Omega \overline{a}(x)\partial_x u_{FEM}\partial_x\phi_i, \forall\phi_i$$

where $\overline{a}$ is the average of a. We can check that the harmonic average and normal average can have a large difference in some cases. For example, if $a = 15\sin^2(x) + 1$, we have $a^* = 4$ but $\overline{a} = 8.5$. Thus, we have $u_{FEM} \approx \frac{8}{17}u_0 \approx u_\epsilon$ and the relative error is $\frac{\|u_{FEM} - u_\epsilon\|_{L_2}}{\|u_\epsilon\|_{L_2}} \approx \frac{9}{17} > 50\%$. We remark that using the higher order coarse mesh finite element method in a higher dimensional case will have similar behavior.

It is shown in the paper [21] that the network tends to learn the low-frequency solution (globally varies without local fluctuation) first. The low-frequency solution is similar to the solution in a coarse grid. If we use the multiscale loss function to train the solution, it will tend to obtain a FEM solution-like solution since when the solution is smooth, the multiscale residual (the loss obtained by applying the multiscale governing equation, i.e., $\mathcal{L}u - f$) is similar to the residual with average parameters. PINN hence fails in solving multiscale problems.

One way to verify our hypothesis is: we can initialize the PINN network such that it can capture the low-frequency component of the solution; if the network cannot capture the low frequency as the training goes on, this implies that the multiscale operator misleads the network and then the training fails. Our proposed method (NH-PINN) can train a network that learns the coarse-scale solution by homogenization; we hence can use the NH-PINN network as the initialization to the classical PINN network, and then train the network further with the classical PINN loss and settings. Our experiments show that the coarse-scale solution initialization cannot improve the classical PINN. This indicates that the set of parameters that results in an approximation of the coarse scale solution may not be optimum for the PINN loss. The optimization algorithm performs well; however, the loss function itself cannot resolve the multiscale features. This verifies our hypothesis of the PINN failure. The details of the experiments can be found in Section 5.

3. Homogenization

As we mentioned in the introduction, to obtain the homogenized parameter, we need to use the solution of the cell problems. For the completeness of the paper, in this section, we briefly revisit the derivation of the homogenization of a diffusion problem and introduce the cell problem in this case. We also included a discussion about the homogenization of a diffusion-reaction equation in the appendix.

We here give an example of the homogenization of multi-dimension elliptic problems. Let us consider:

$$-\frac{\partial}{\partial x_i}\left(a_{ij}(x/\epsilon)\frac{\partial}{\partial x_j}u_\epsilon(x)\right) = f(x), x \in \Omega \tag{4}$$

with $u_\epsilon(x) = 0$ on $\partial\Omega$; here we use the Einstein notation. We seek for $u_\epsilon(x)$ in the asymptotic expansion:

$$u_\epsilon(x) = u_0(x, x/\epsilon) + \epsilon u_1(x, x/\epsilon) + \epsilon^2 u_2(x, x/\epsilon)... \tag{5}$$

where $u_j(x, y)$ are periodic in $y = x/\epsilon$. Denote

$$A^\epsilon = -\frac{\partial}{\partial x_i}\left(a_{ij}(x/\epsilon)\frac{\partial}{\partial x_j}\right)$$

It is not hard to check that:

$$A^\epsilon = \epsilon^{-2}A_1 + \epsilon^{-1}A_2 + \epsilon^0 A_3,$$

where,

$$A_1 = -\frac{\partial}{\partial y_i}\left(a_{ij}(y)\frac{\partial}{\partial y_j}\right),$$
$$A_2 = -\frac{\partial}{\partial x_i}\left(a_{ij}(y)\frac{\partial}{\partial y_j}\right) - \frac{\partial}{\partial y_i}\left(a_{ij}(y)\frac{\partial}{\partial x_j}\right),$$
$$A_3 = -\frac{\partial}{\partial x_i}\left(a_{ij}(y)\frac{\partial}{\partial x_j}\right).$$

We hence have $A_1u_\epsilon + A_2u_\epsilon + A_3u_\epsilon = f$. Equating the terms with the same power, it follows that,

$$A_1u_0 = 0, \tag{6}$$
$$A_1u_1 + A_2u_0 = 0, \tag{7}$$
$$A_1u_2 + A_2u_1 + A_3u_0 = f. \tag{8}$$

Substitute the A_1 and the equation (6) becomes:

$$-\frac{\partial}{\partial y_i}\left(a_{ij}(y)\frac{\partial}{\partial y_j}\right)u_0 = 0$$

The theory of the second-order ODE implies that u_0 is independent of y and this will further simplify the equation (7); it follows that,

$$-\frac{\partial}{\partial y_i}\left(a_{ij}(y)\frac{\partial}{\partial y_j}\right)u_1 = \left(\frac{\partial}{\partial y_i}a_{ij}(y)\right)\frac{\partial}{\partial x_j}u_0.$$

$u_1(x, y)$ can be solved by introducing $\chi_j(y)$ which is the solution of the problem:

$$-\frac{\partial}{\partial y_i}\left(a_{ij}(y)\frac{\partial}{\partial y_j}\right)\chi_j = \frac{\partial}{\partial y_i}a_{ij}(y), \tag{9}$$
$$\chi_j \text{ is periodic in } y \text{ with mean } 0. \tag{10}$$

The above problems (10) are called cell problems, and they are to be solved in one period of y, or, in the unit cell $Y = [0, 1]^d$ where d is the dimension of the problem. u_1 can be then written as:

$$u_1(x, y) = \chi_j\frac{\partial u_0}{\partial x_j}(x).$$

Finally, we have,

$$-\frac{\partial}{\partial y_i}\left(a_{ij}(y)\frac{\partial}{\partial y_j}\right)u_2 = A_2u_1 + A_3u_0 - f.$$

The ODE has a solution only if the right-hand side has zero mean in one period of y, i.e.,

$$\int_Y (A_2 u_1 + A_3 u_0 - f) dy = 0.$$

This solvability condition then gives,

$$-\frac{\partial}{\partial x_i}\left(a^*_{ij}\frac{\partial}{\partial x_j}\right)u_0 = f, \tag{11}$$

where $a^*_{ij} = \int_Y (a_{ij} + a_{ik}\frac{\chi_j}{\partial y_k})dy$ is called the homogenized coefficient and u_0 is the homogenized solution.

Remark. Note that $a_{ij}(x/\epsilon)$ has no slow dependency x. If $a_{ij}(\cdot)$ also depends on slow variable x, the cell problems solutions also have the x dependency, i.e.,

$$-\frac{\partial}{\partial y_i}\left(a_{ij}(x,y)\frac{\partial}{\partial y_j}\right)\chi_j(x,y) = \frac{\partial}{\partial y_i}a_{ij}(x,y),$$

$u_1(x,y)$ then becomes:

$$u_1(x,y) = \chi_j(x,y)\frac{\partial u_0}{\partial x_j}(x)$$

The homogenized coefficient $a^*(x)$ then has slow dependency since we only integrate in y of $\chi_j(x,y)$. This case will be illustrated in Section 5.3.

We now briefly discuss the convergence property of homogenization. The goal is to estimate the remainder R which is defined as:

$$u_\epsilon = u_0 + \epsilon u_1 + R.$$

R then satisfies that $|R| < C\epsilon$ where C is a constant. We also have the energy estimate $\int a|\frac{d}{dx}R|^2 \le C\epsilon$.

4. Neural homogenization-based PINN (NH-PINN)

In this section, we introduce our proposed method: neural homogenization-based PINN (NH-PINN). We have seen in Section 2 (and will see more examples in Section 5) that PINN is unable to solve the multiscale problems with a satisfying accuracy; however from Section 3, the homogenized equation loses its multiscale property since the homogenized coefficient is either constant or depends only on the slow variable; besides, the previous studies of the PINN have shown that the PINN can deal with equations with constant coefficients; this motivates us to use homogenization.

The idea is to decompose the original hard problem into easier and solvable problems. Asymptotic expansion homogenization consists of three steps, and we have shown that the homogenized solution converges to the real solution in Section 3; if each of the steps can be solved by PINN easily, we believe that the accuracy of the final data-driven (by PINN) solution can approximate the real solution closely. Our method consists of three steps:

1. Solve the cell problems using PINN.
2. Evaluate the homogenized coefficients.
3. Solve the homogenized equation using PINN.

The first step is to solve the cell problems. Since there is no fast dependency, PINN can solve cell problems easily. It should be noted that the cell problems are equipped with the periodic boundary condition; we find that the performance is satisfactory when solving the 1D periodic problem; however, for the high dimensional cases, the PINN cannot give a good result. To deal with this issue, we propose an oversampling trick that facilities the PINN with a few more sampling points. Our numerical experiments show that this trick significantly improves the PINN performance for handling $2D$ periodic problems. We introduce this trick in detail in Section 4.1.

The second step is to evaluate the coefficients. This step usually involves calculating the derivatives of the cell problem solutions. PINN solves the cell problems, i.e., the solution is approximated by a smooth network; hence with the help of auto differentiation, these derivatives can be calculated very easily. This is one of the benefits of our method.

The last step is to solve the homogenized equation with PINN. Since PINN has shown its power in solving the equations with constant or slow varying coefficients, we do not expect any difficulty in this step.

There are three error sources in our problem. Firstly, the error comes from the homogenization, however, this error is small, in particular when ϵ is small and the scales are very different. The other two errors come from the PINN solver accuracy when solving the cell problems and the homogenized equation; we will show in the numerical section that all three errors can be controlled, but the error of solving the homogenized equation dominates the total error of our proposed method.

4.1. Oversampling

As we have discussed before, each cell problem is equipped with a periodic boundary condition. We find that the generalization error is large when we solve the $2D$ equation with a periodic boundary condition. Hence, we are going to introduce an oversampling trick that is used to aid PINN in solving an equation with periodic boundary conditions. Our experiments show that this trick greatly improves the training for the $2D$ problems. For simplicity, we will illustrate this idea in the $1D$ case and the extensions to the high dimensional cases are similar.

Suppose $\Omega = [x_0, x_t]$ and recall (3), we then have $x_0 = q_1$ and $x_t = q_2$. Since we are solving the periodic problem, the minimization problem then becomes:

$$\min_{\beta} \frac{w_1}{N_f} \sum_{i=1}^{N_f} |\mathcal{L}(\mathcal{F}_\beta(p_i)) - f(p_i)|^2 + \frac{w_2}{2} \sum_{i=1}^{2} |(\mathcal{F}_\beta(q_1) - \mathcal{F}_\beta(q_2)|^2, \tag{12}$$

where the second term above indicates the periodic boundary condition. To enforce boundary learning, we over-sample some points on both sides of the domain. Denote $\Delta x = 1/(N_f - 1)$, we then include additional 2 sets of points $\{q_i^{left}\}_{i=1}^{N_o}$ and $\{q_i^{right}\}_{i=1}^{N_o}$, where N_o is the number of oversampling layers and $N_o < N_f - 2$. In this work, we consider the uniform mesh, we hence have, $q_{i+1}^{left} - q_i^{left} = \Delta x$ and $q_{i+1}^{right} - q_i^{right} = \Delta x$. We also require:

$$q_{N_o}^{left} + x_t - x_0 = x_t - \Delta x,$$
$$q_1^{right} - (x_t - x_0) = x_0 + \Delta x.$$

Due to the above assumptions, we have $q_i^{left} + x_t - x_0 \in \{p_j\}$ and $q_i^{right} - x_t + x_0 \in \{p_j\}$ for all $i = 1, ..., N_o$. If we denote $q_i^{left} + x_t - x_0 := p_i^{left}$ and $q_i^{right} - x_t + x_0 = p_i^{right}$, the minimization becomes:

$$\min_{\beta} \left\{ \frac{w_1}{N_f} \sum_{i=1}^{N_f} |\mathcal{L}(\mathcal{F}_\beta(p_i)) - f(p_i)|^2 + \frac{w_2}{2} \sum_{i=1}^{2} |(\mathcal{F}_\beta(q_1) - \mathcal{F}_\beta(q_2)|^2 \right.$$
$$\left. + \frac{w_3}{N_o} \sum_{i=1}^{N_0} \left(|(\mathcal{F}_\beta(q_i^{left}) - \mathcal{F}_\beta(p_i^{left})|^2 + |(\mathcal{F}_\beta(q_i^{right}) - \mathcal{F}_\beta(p_i^{right})|^2 \right) \right\},$$

where $w_1 + w_2 + w_3 = 1$ are positive constants.

5. Numerical examples

In this section, we perform numerical experiments to demonstrate our proposed method (neural homogenization based PINN: NH-PINN). We consider three equations with different cell problems, and the corresponding homogenized equations are also different. We will first define all necessary notations in Section 5.1. The remaining sections are the experiment results.

It should be noted that we obtain reference solutions by the finite element method. This includes solving cell problems and homogenized equations. To evaluate the homogenized coefficients, one usually needs to calculate the derivatives of the cell problems and do the integration. For the reference solution, we use the finite difference scheme to evaluate the derivatives; and the quadrature rule to compute the integration.

5.1. Notations

We first introduce the notations and define the relative errors. Suppose we are solving the following equation with a proper boundary condition:

$$L(a(x, x/\epsilon), u_\epsilon) = f, x \in \Omega \tag{13}$$

where L is a well-defined operator; $a(x, x/\epsilon)$ is the coefficient that contributes the multiscale property to the system; and f is the source with proper regularity. u_ϵ is the reference solution that is obtained by the finite element methods (FEM) with the fine mesh. To get the homogenized equation, we first compute the cell problems; the solutions of the cell problems are the key component in determining the homogenized equation. The cell problems can be solved by PINN and the traditional numerical methods. We denote the homogenized equation as

$$L^*(b^*(x), v(x)) = f, \tag{14}$$

Table 1
Notations of the solutions. u_ϵ which is not listed here is the reference solution that solves the multiscale PDE directly by fine-scale finite element methods.

Solution notation	Cell problems solver	Homogenized equation solver
$p(x)$	PINN	PINN
$v(x)$	FEM	FEM
$w(x)$	PINN	FEM

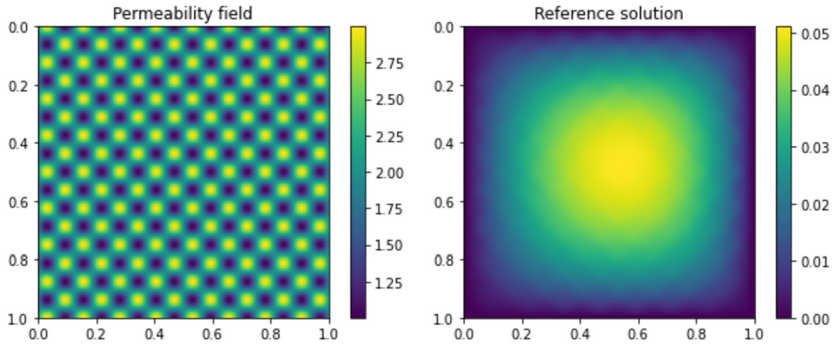

Fig. 2. Left: permeability of the 2D elliptic problem. Note that $\epsilon = 1/8$. Right: the reference solution. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

if $b^*(x)$ (cell problems) is computed by the FEM; here $v(x)$ is the FEM solution to the system. We denote the homogenized equation as

$$L^*(a^*(x), \cdot) = f, \tag{15}$$

if $a^*(x)$ (cell problems) is computed by PINN. If the system is solved by the PINN, the solution will be denoted as $p(x)$; and if this equation is solved by the FEM, the solution will be denoted as $w(x)$. This solution $w(x)$ can quantify the errors of the PINN method in solving the homogenized equation (15). The notations of the different solutions are summarized in Table 1. We compute the relative errors to quantify the accuracy of the proposed method (NH-PINN) and the definitions are given as follows,

$$e_1 = \frac{\|p(x) - u_\epsilon(x)\|}{\|u_\epsilon\|}, e_2 = \frac{\|w(x) - u_\epsilon(x)\|}{\|u_\epsilon(x)\|}, e_3 = \frac{\|p(x) - w(x)\|}{\|w(x)\|}, e_4 = \frac{\|v(x) - u_\epsilon(x)\|}{\|u_\epsilon\|}$$

where $\|.\|$ is the L_2 norm. e_1 is the error of the proposed method (NH-PINN); this error is the ultimate measure of NH-PINN and the number will be compared with the relative error of the classical PINN. e_2 is the error coming from the cell problems since we fix the homogenized equation solvers; if this error is big, this means the cell problems are not well solved by the PINN. e_3 measures the PINN solver's accuracy in solving the homogenized equation; if this error is small, it can then show that the homogenized equation is easy to be solved by PINN. e_4 is the error of the homogenization implemented by the classical numerical methods, this is a theoretical lower bound, NH-PINN cannot be better than this since there are errors in both solving the cell problems and the homogenized equations.

5.2. 2D elliptic equation

The first example is a $2D$ elliptic problem; the detailed homogenization is presented in Section 3. We consider the following $2D$ elliptic equation:

$$-\frac{\partial}{\partial x_i}\left(a(\frac{x}{\epsilon})\frac{\partial}{\partial x_i}u_\epsilon(x)\right) = f(x), x \in \Omega, \tag{16}$$

$$u_\epsilon(x) = 0, x \in \partial\Omega. \tag{17}$$

In our examples, $\Omega = [0,1]^2$ and the permeability $a(x/\epsilon) = 2 + \sin(2\pi x_1/\epsilon)\cos(2\pi x_2/\epsilon)$ and $\epsilon = \frac{1}{8}$ (shown in Fig. 2). The source is $f(x) = \sin(x_1) + \cos(x_2)$.

We first present the results of solving Equation (17) using the classical PINN; we use the same network structure and the result of the prediction is shown in Fig. 3.

For NH-PINN, to solve the cell problem, we use a 4-layer fully connected network ($2 \times 64 \to 64 \times 64 \to 64 \times 64 \to 64 \times 1$) activated by Tanh. The domain is discretized with a 101×101 mesh; we use these points and an additional 2-layer

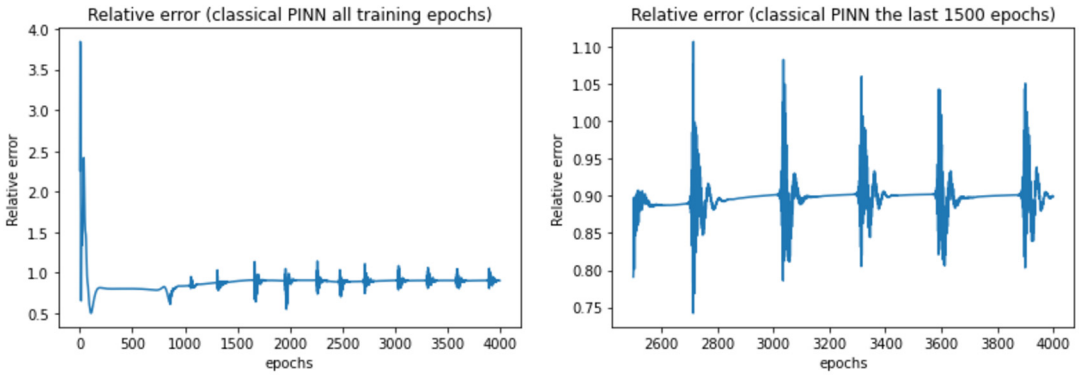

Fig. 3. 2D elliptic problem by the classical PINN. Relative error as a function of the training epochs, the entire history (left), and the last 1500 epochs (right). We train the network for 4000 epochs with Adam gradient descent (learning rate 0.0001). The average relative error of the last 500 epochs is 0.900758.

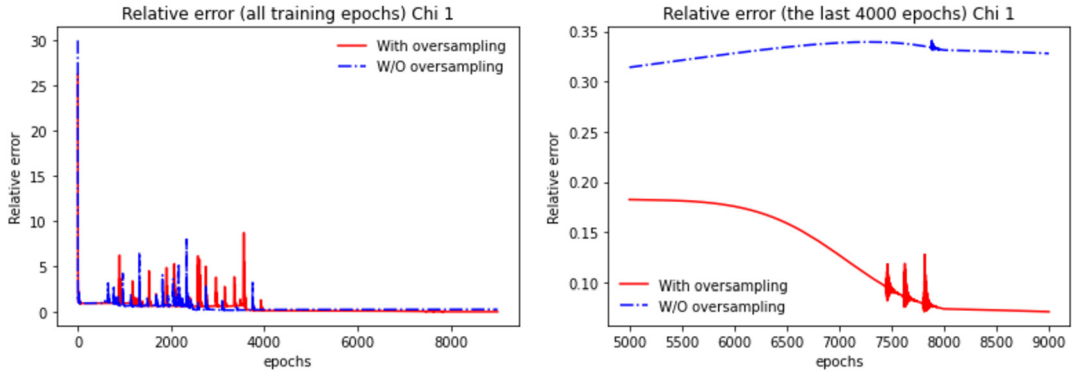

Fig. 4. 2D elliptic cell problem χ_1. Relative error as a function of the training epochs for χ_1. Left: all training epochs; right: the last 4000 epochs. The average relative errors of the last 300 epochs for the oversampling and without oversampling are 0.072034 and 0.329296 respectively. For both methods, 23×23 points are used in training, while two layers of oversampling are used in the oversampling experiments.

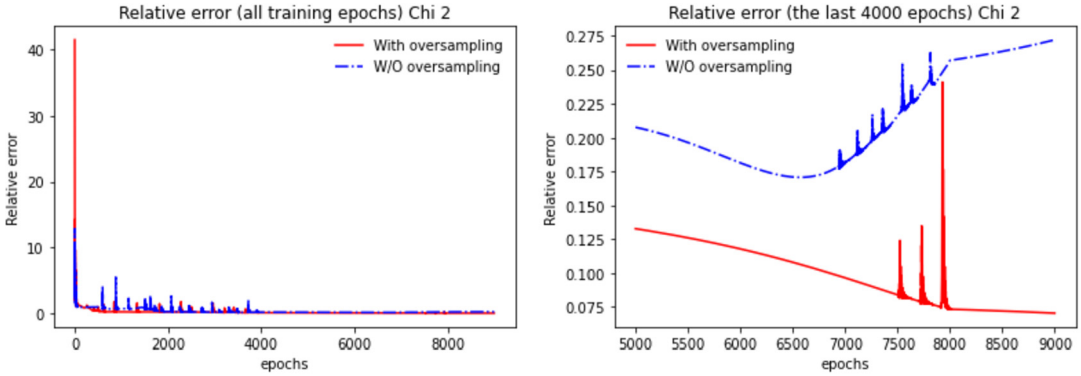

Fig. 5. 2D elliptic cell problem χ_2. Relative error as a function of the training epochs for χ_2. Left: all training epochs; right: the last 4000 epochs. The average relative errors of the last 300 epochs for the oversampling and without oversampling are 0.071396 and 0.266358 respectively. For both methods, 23×23 points are used in training, while two layers of oversampling are used in the oversampling experiments.

oversampling on each side of the domain in training. We find that the relative errors for the oversampling methods are very robust to the learning rate, the weights of the losses, and other hyperparameters; that is, relative error drops to a similar value even if we change some hyperparameters. We do not observe similar results for the standard method without oversampling, and hence do a series of experiments and then choose the best results. The results for χ_1 and χ_2 are shown in Fig. 4 and Fig. 5 respectively.

For solving the learned homogenized equation, we use a network with the same structure as the cell problem. The domain is discretized with a uniform mesh of the size 21×21, and these grid points are then used in the training. To test the network performance, we use a 101×101 mesh. The relative errors are plotted in Fig. 6 and we are interested in the

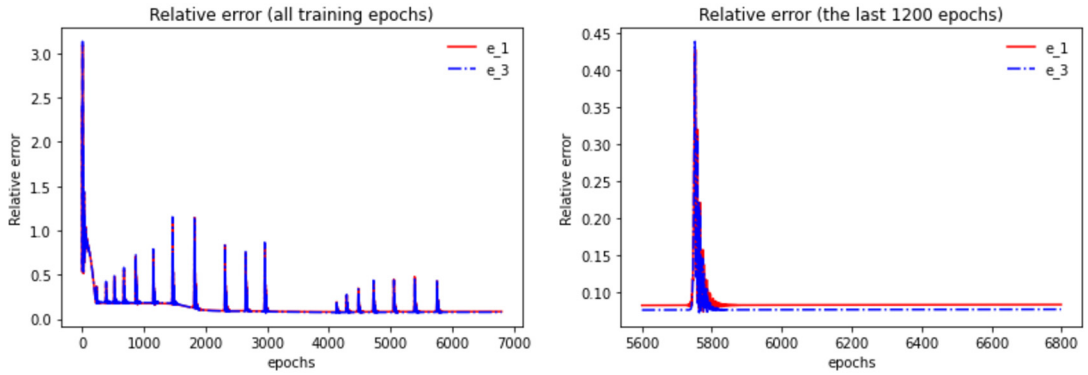

Fig. 6. $2D$ elliptic e_1 and e_3 relative errors with respective to the training epochs. Left: history of all training epochs; right: history of the last 1200 epochs. The average relative errors of the last 500 epochs are: $e_1 = 0.082994$ and $e_3 = 0.076604$.

Table 2
Relative errors for the $2D$ elliptic problem.

e_1	e_2	e_3	e_4
0.082994	0.0212534	0.076604	0.021316

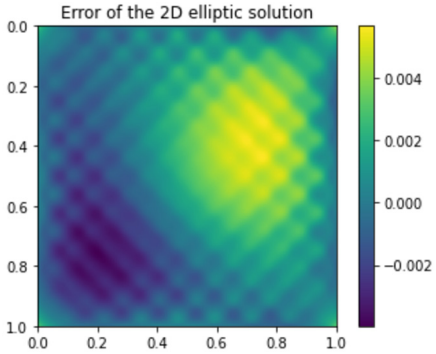

Fig. 7. $2D$ elliptic error distribution. The error is calculated as the difference between the reference solution and the NH-PINN solution.

relative errors after the training is stabilized, i.e., we compute the average errors of the last 500 epochs. The numbers are shown in Table 2.

Since the $2D$ solution is not easy to observe, we also calculate the error $u_{reference} - u_{NH-PINN}$ and plot the $2D$ error distribution; please check Fig. 7 for the illustration.

5.2.1. Interpretation of the results

We can see from Table 2 and Fig. 10, that the e_1 drops to around 8%; this is a great improvement when compared to the classical method (relative error 0.900758) which we apply the PINN directly to Equation (13). e_2 is close to e_4 which is the theoretical optimal relative error, this means that the cell problems can be solved by PINN very accurately; besides, e_1 is slightly bigger than e_3, together these two comparisons imply that most of the errors come from solving the homogenized equation. e_3 is around 0.076604, this is the error of solving the homogenized equation using PINN; however, the homogenized equation is much easier to be solved by PINN when compared to the multiscale PDE.

One interesting fact is that $e_2 < e_4$. We observe a similar phenomenon in some other examples. We have an intuitive explanation for this phenomenon. The problem may originate from the evaluation of the $\partial \chi_i / \partial y$. For the traditional method, the easiest way to calculate the derivative is the finite difference method; computationally, this approximation contributes an error and, as a result, a^* may not be evaluated exactly. On the other hand, for the NH-PINN method, we approximate χ_i by the network. The derivatives can be derived by auto differentiation. If χ_i are approximated accurately, we may expect a small error in the derivatives when compared to the exact derivatives. Consequently, we obtain a better a^* and this results in a smaller error. The derivatives can be approximated by some high-order methods; however, the computational costs may be higher than the neural network method. We observe a similar situation in some other examples, we hence conclude that our proposed NH-PINN method of solving the cell problems potentially can give an accurate approximation of the

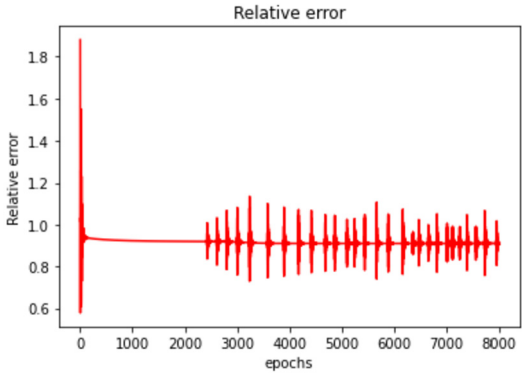

Fig. 8. 2D elliptic transfer learning of classical PINN.

homogenized coefficients in a relatively easier way; as a result, the final solution of the resulting homogenized equation may also be improved.

5.2.2. Comments on the ϵ

We choose $\epsilon = \frac{1}{8}$ in this experiment but when ϵ is small the conclusions of the work still hold. On the one hand, from the theory point of view, when $\epsilon \to 0$, the homogenized solution will converge to the reference solution (without homogenization). On the other hand, from the implementation point of view, we solve two equations (cell problems (10) and homogenized equation (11)) and both equations do not depend on ϵ directly. Hence, smaller ϵ will not have a great impact on the method.

5.2.3. Transfer learning

In this section, we present a transfer learning result that can explain the reason for the failure of the PINN. As we have discussed in Section 2.2, we can initialize the training of the classical PINN with a set of parameters that can capture the low-frequency component of the solution. We solve the homogenized equation with the network and the homogenized solution is the coarse scale component, we hence use the trained network of the NH-PINN method as the initialization of the classical PINN (with the standard multiscale loss coming from the multiscale equations). All settings including the weights, training mesh size, and learning rate are kept the same; but we get an inaccurate result. The relative error blows up very quickly and the stabilized solution has a relative error (0.909675, check the Fig. 8) which is similar to applying PINN directly to the multiscale PDE. This implies that the set of model parameters which is optimal (or close) to the NH-PINN loss, and gives an accurate approximation to the coarse scale solution is not optimal (local optima and global optima) to the multiscale loss of the classical PINN; the multiscale loss prevents the network from learning the coarse-scale solution.

5.3. 1D slowly varying periodic coefficients

In this section, we consider again an elliptic equation. The difference is that the coefficient $a(x, x/\epsilon)$ has a slow dependency x; as a result, the homogenization process is different: we need to solve a cell problem for each x in the domain; besides, the a^* in the homogenized equation also depends on x. We consider the following elliptic equation:

$$-\frac{d}{dx}\left(a(x, \frac{x}{\epsilon})\frac{d}{dx}u_\epsilon\right) = f, x \in [0, \pi], \tag{18}$$

$$u_\epsilon(0) = u_\epsilon(\pi) = 0. \tag{19}$$

In this example, $a(x, x/\epsilon) = 0.5\sin(2\pi x/\epsilon) + \sin(x) + 2$ where $\epsilon = \frac{1}{8}$, it is demonstrated in Fig. 9 Left, we can observe the high frequency oscillations. The source $f(x) = \sin(x)$. We first present the results of solving Equation (19) using the classical PINN. We use the same network structure as NH-PINN detailed later and the result of the prediction is shown in Fig. 10.

For NH-PINN, to solve each cell problem, we use a network of a structure $1 \times 64 \to 64 \times 64 \to 64 \times 64 \to 64 \times 1$; the network is activated with Tanh as usual. To train the network, we use 101 uniform-spaced points. We solve the cell problems by PINN and denote the resulting homogenized coefficient as $a_p^*(x)$; the relative error is then calculated as $e_a = \|a^*(x) - a_p^*(x)\|/\|a^*(x)\| = 0.010304$. We present $a^*(x)$ and $da^*(x)/dx$ in Fig. 11.

Lastly, we need to solve the learned homogenized equation. We use a network with the same structure as before and 101 points for training. The model is tested with 401 points. The history of the relative errors e_1 and e_3 are shown in Fig. 12. We also compute the average relative errors (when the training is stable) in Table 3; more precisely, e_1 and e_3 are calculated as the average errors of the last 500 epochs of the training. The solution is demonstrated in Fig. 9 Right.

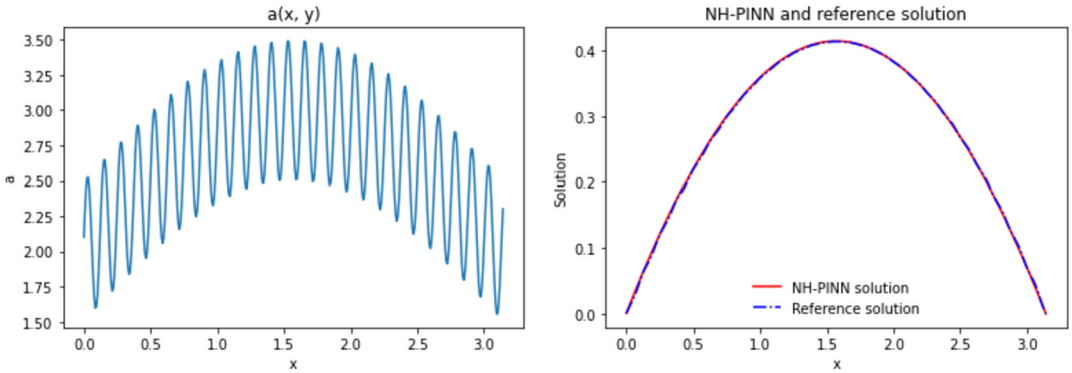

Fig. 9. 1D slowly varying elliptic problem. Left: $a(x, x/\epsilon)$, where $\epsilon = 1/8$. Right: NH-PINN solution vs the reference solution. The relative error $e_1 = 0.005819$.

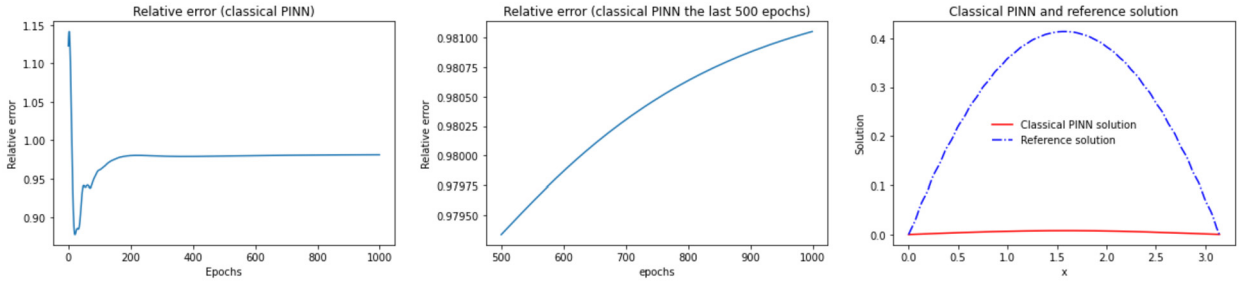

Fig. 10. $1D$ slowly varying elliptic problem by the classical PINN. Relative error as a function of the training epochs, the entire history (left), the last 500 epochs (middle) and the solution (right). We train the network for 1000 epochs with Adam gradient descent (learning rate 0.0001). $\omega_1 = 1/11$ and $\omega_2 = 10/11$. The average relative error of the last 100 epochs is 0.980968; however, we observe a climb in the error (check the middle image). We vary the learning rate and the loss weights, however, for all the combinations, we fail to get a satisfactory result.

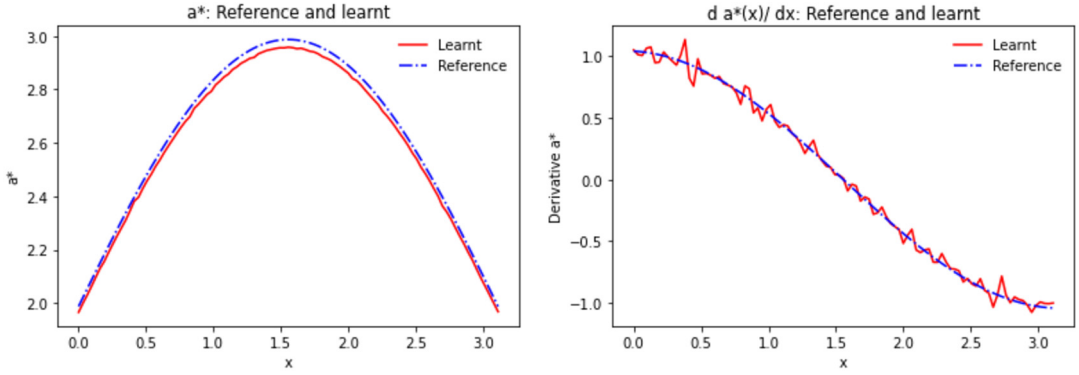

Fig. 11. $a^*(x)$ and $da^*(x)/dx$ of the slowly varying problem.

Table 3
Relative errors for the $1D$ slowly varying elliptic problem.

e_1	e_2	e_3	e_4
0.005819	0.007668	0.003665	0.006201

5.3.1. Interpretation of the results

We can see from Table 3 and Fig. 10 that, the relative error e_1 of NH-PINN is much better than the one of the classical PINN (0.980968). This again shows that our method improves the classical PINN performance. There is one comment about this example. Different from the other two examples, the homogenized coefficients $a^*(x)$ can be evaluated exactly. We hence do not observe that $e_2 < e_4$ as the other two examples; however, we can see from the Table 3 that $e_1 < e_4$. This is not observed in the other two examples. This is probably due to the FEM solver accuracy being lower than the PINN. Since

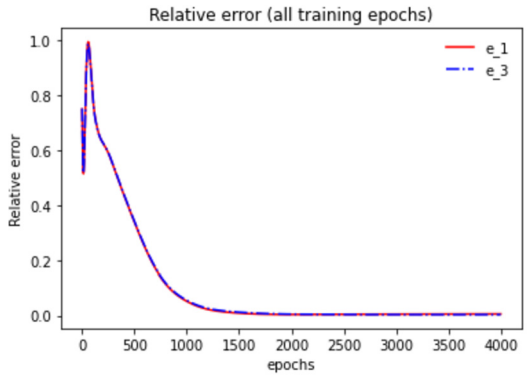

Fig. 12. 1D slowly varying problem e_1 and e_3 relative errors with respective to the training epochs. The average relative errors of the last 500 epochs are: $e_1 = 0.005819$ and $e_3 = 0.003665$.

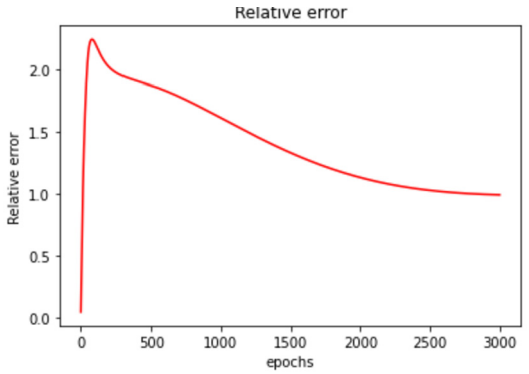

Fig. 13. 1d slow varying elliptic transfer learning of classical PINN.

e_3 is very small, this indicates that the PINN solver is very accurate. It follows that the total error (error in the homogenized coefficients and error in solving the homogenized equation) of PINN is even smaller than the theoretical upper bound.

5.3.2. Transfer learning

Similar to the first example, we use the trained network of NH-PINN as the initialization of the classical PINN directly. All settings, including the weights, and learning rate are kept the same; but we get an inaccurate result with a relative error of 1.005152. The relative error increases and stabilizes at a similar value as before (please check Fig. 13).

5.4. Diffusion and reaction equation

In this section, we consider another example whose cell problems and the homogenized equation are different from before; The detailed homogenization process can be seen in the Appendix A. We are going to test the NH-PINN with two different scales. The purpose of the experiments is to show that NH-PINN can deal with different scales; however, the classical PINN fails for both scales and performs even worse when ϵ is small. We consider the following example:

$$\frac{\partial u_\epsilon}{\partial t} - D\nabla \cdot \nabla u_\epsilon + \frac{1}{\epsilon} r(\frac{x}{\epsilon}) u_\epsilon = f, x \in \Omega, \tag{20}$$

$$u_\epsilon(x) = 0, x \in \partial\Omega. \tag{21}$$

In our example, $\Omega = [-\pi, \pi]$. We set $r(x/\epsilon) = \cos(x/\epsilon)$. We use different ϵ and present the results in the following two sections.

5.4.1. Scale $\epsilon = 1/10$

In this section, we set $\epsilon = \frac{1}{10}$ and $D = 2$, the $r(x/\epsilon)$ is demonstrated in Fig. 14 Right. The source is $f(x) = \sin(2\pi x)$. As we have discussed before, the classical PINN cannot give us an accurate prediction; the results of applying PINN directly are shown in Fig. 15.

For NH-PINN, the cell problem is solved with a 4-layer network of the structure $1 \times 64 \to 64 \times 64 \to 64 \times 64 \to 64 \times 1$; the network is activated by the Tanh function. For the network training, we use 101 points uniformed placed in the domain

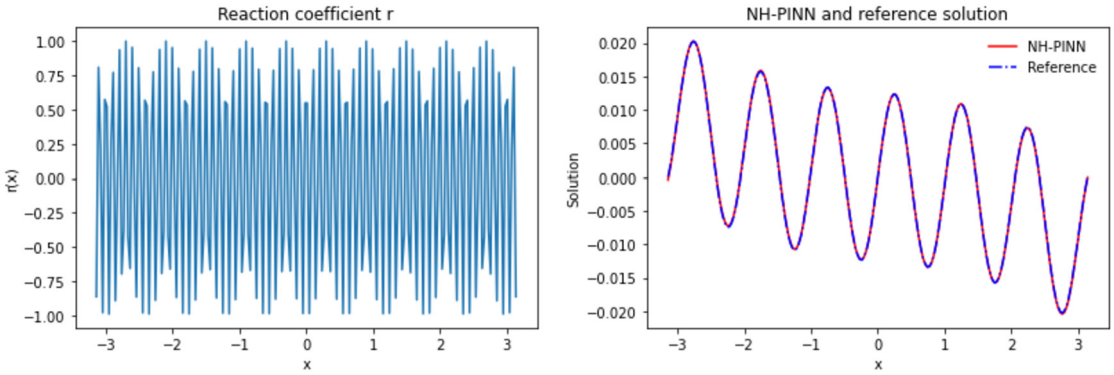

Fig. 14. Diffusion and reaction problem. Left: $r(x) = cos(x/\epsilon)$, where $\epsilon = 1/10$. Right: NH-PINN solution vs the reference solution. The relative error $e_1 = 0.012388$.

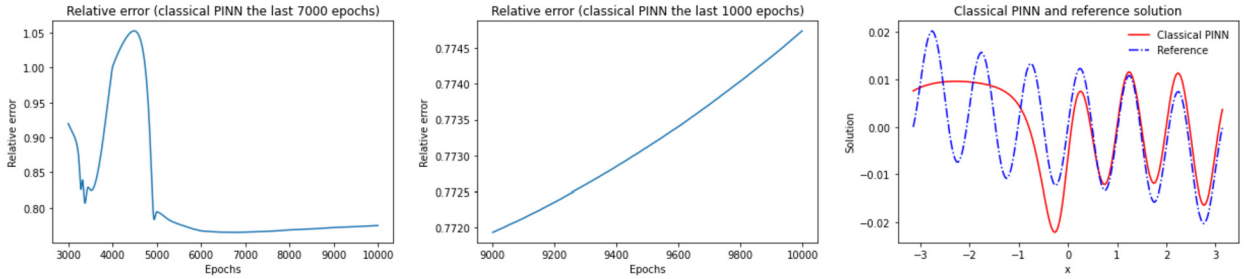

Fig. 15. Diffusion and reaction problem with $\epsilon = 1/10$ solved by the classical PINN. Relative error of the last 7000 epochs (left), the last 1000 epochs (middle), and the solution (right) of the classical PINN. We train the network for 10000 epochs with Adam gradient descent (learning rate 0.015). $\omega_1 = 1/7$, $\omega_2 = 5/7$ and $\omega_3 = 1/7$, where ω_3 is the weight of the initial condition. The average relative error of the last 100 epochs is 0.774558; however, we observe a climb in error (check the middle image). We vary the learning rate and the loss weights; however, for all the combinations, we fail to get a satisfactory result.

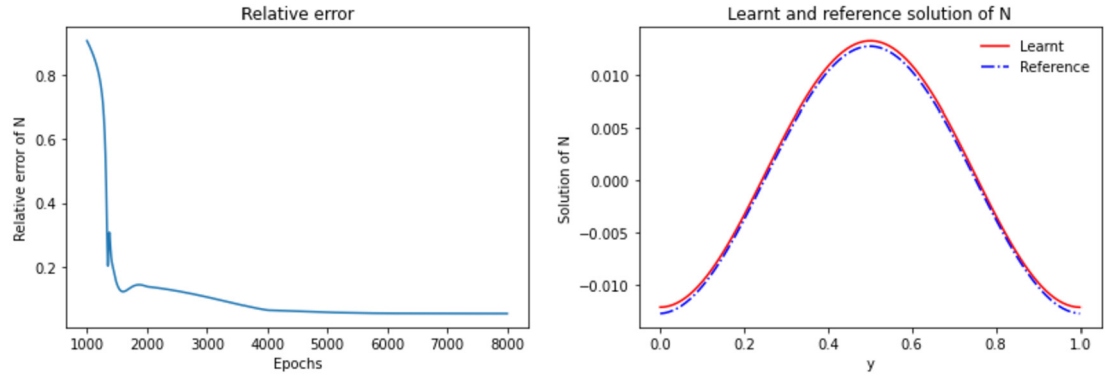

Fig. 16. Diffusion and reaction equation cell problem (A.2) with $\epsilon = 1/10$. Left: relative error with respected to the training epochs; note: only last 7000 epochs are shown. The average relative error in $N(y)$ of the last 500 epochs is 0.053309. Right: N(y) reference vs learnt.

and additional 2-layer oversampling on each side of the domain. The relative error in N drops to 0.053309 when the training is stable (we take the average of the last 500 epochs). The solution and training history of the cell problem are shown in Fig. 16.

To solve the homogenized equation, we use another 4-layer network activated by Tanh; the network structure is the same as the one used in the cell problems except that the first layer's dimension is 2×64. The spatial domain and the temporal domain are both discretized with 100 uniformed placed mesh points; the grid points are then used in training. We will test the solution at the terminal time and uniformly place 201 points in space. The training epoch is set to be $10,000$ and we use the standard Adam gradient descent algorithm. The history of the relative errors is shown in Fig. 17, and the average relative errors when the training is stable are presented in Table 4. We also solve Equation (21) using the classical PINN; we use the same network structure, and the result of the prediction is shown in Fig. 15.

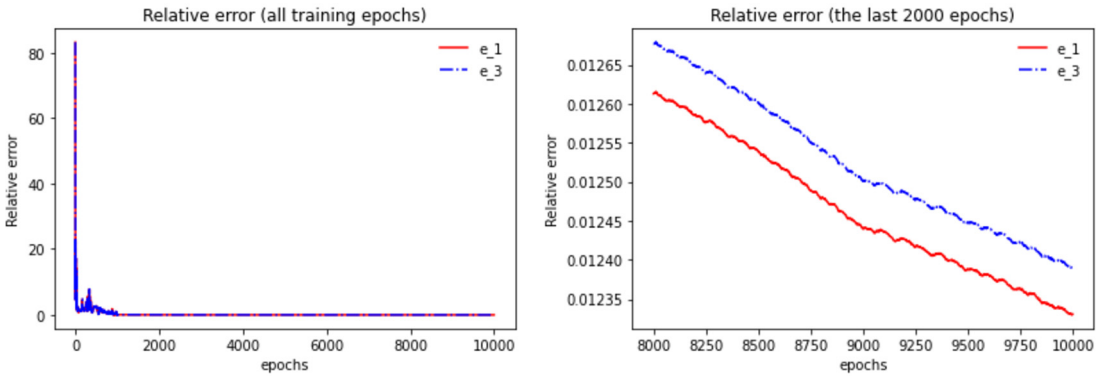

Fig. 17. Diffusion and reaction problem with $\epsilon = 1/10$, e_1 and e_3 relative error as a function of the training epochs. Left: history of all training epochs; right: history of the last 2000 epochs. The average relative errors of the last 500 epochs are: $e_1 = 0.012388$ and $e_3 = 0.012448$.

Table 4
Relative errors for the diffusion-reaction problem.

e_1	e_2	e_3	e_4
0.012388	0.0012916	0.012448	0.0012917

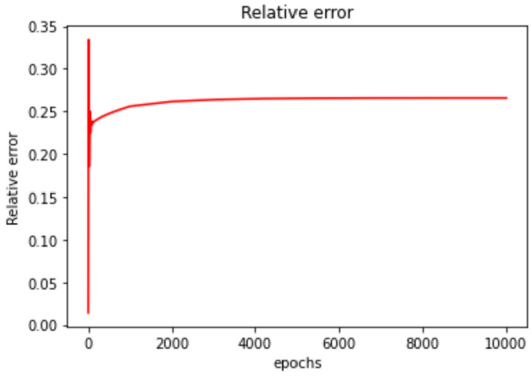

Fig. 18. Diffusion reaction equation with $\epsilon = 1/10$ transfer learning of classical PINN.

From Table 4 and Fig. 15, firstly we can observe that the relative error e_1 of NH-PINN is much better than the classical PINN. If we look at Fig. 17, the relative errors of NH-PINN are still decaying. Since e_2 and e_4 are very closed to each other ($e_2 < e_4$), we conclude that the PINN aided homogenization is a potential alternative to the traditional numerical-driven homogenization. Finally, because e_2 is small, this implies that most errors of the method still come from solving the homogenized equation with PINN; however, this has been much improved when compared to applying PINN on the multiscale PDE. The transfer learning is the same as before, the stabilized relative error is 0.265721 and cannot give us a better result (please check Fig. 18).

5.4.2. Scale $\epsilon = 1/50$

In this section, we set $\epsilon = \frac{1}{50}$ and $D = 2$, the $r(x/\epsilon)$ is demonstrated in Fig. 19 Right. The source is $f(x) = \sin(2\pi x)$. As we have discussed before, the classical PINN cannot give us an accurate prediction; the results of applying PINN directly are shown in Fig. 20.

For $\epsilon = 1/50$, we use the same network setting as before. The relative error of N drops to 0.057602 when the training is stable (we take the average of the last 500 epochs). The solution and training history of the cell problem is shown in Fig. 21.

For the homogenized equation with $\epsilon = 1/50$, we use the same setting as the $\epsilon = 1/10$ case. The history of the relative errors is shown in Fig. 22, and the average relative errors when the training is stable are presented in Table 5. We also solve Equation (21) using the classical PINN; we use the same network structure, and the result of the prediction is shown in Fig. 20.

Similar as the $\epsilon = 1/10$ case, firstly, we can observe from Table 5 and Fig. 20 that the relative error e_1 of NH-PINN is much better than the classical PINN. Since $e_2 < e_4$ are closed to each other, we conclude that PINN-aided homogenization is

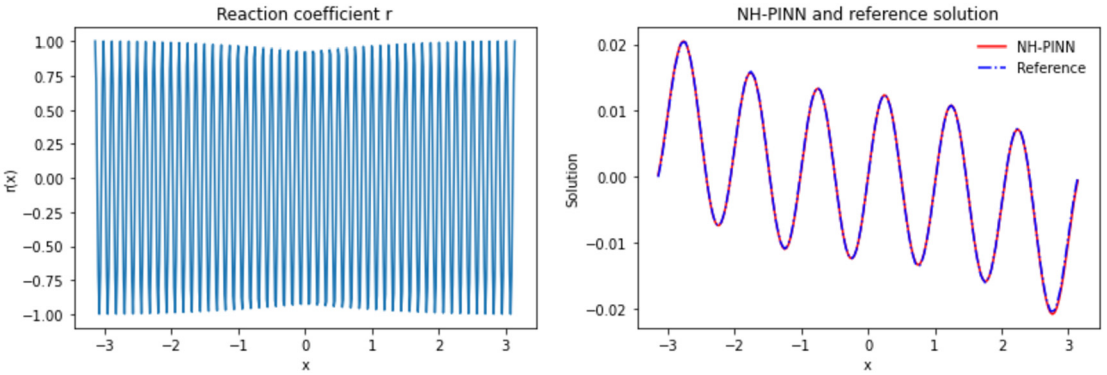

Fig. 19. Diffusion and reaction problem. Left: $r(x/\epsilon) = cos(x/\epsilon)$, where $\epsilon = 1/50$. Right: NH-PINN solution vs the reference solution. The relative error $e_1 = 0.018582$.

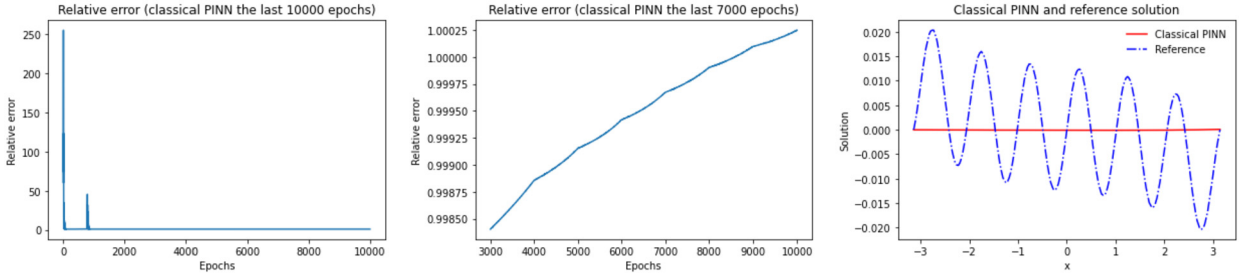

Fig. 20. Diffusion and reaction problem with $\epsilon = 1/50$ solved by the classical PINN. Relative error of the last $10,000$ epochs (left), the last $7,000$ epochs (middle) and the solution (right) of the classical PINN. We train the network for 10000 epochs with Adam gradient descent (learning rate 0.05). $\omega_1 = 1/7$, $\omega_2 = 5/7$ and $\omega_3 = 1/7$, where ω_3 is the weight of the initial condition. The average relative error of the last 100 epochs is 1.000237; however, we observe a climb in the error (check the middle image). We vary the learning rate and the loss weights; however, for all the combinations, we fail to get a satisfactory result.

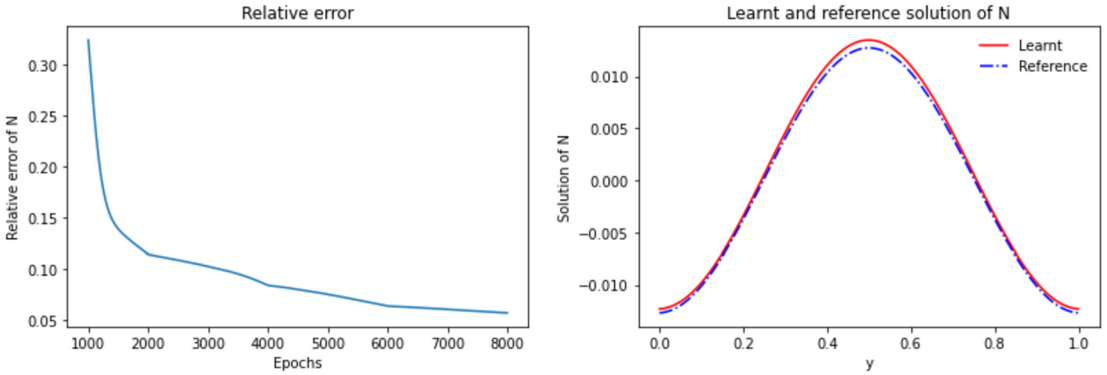

Fig. 21. Diffusion and reaction equation cell problem (A.2) with $\epsilon = 1/50$. Left: relative error with respected to the training epochs; note: only last 7000 epochs are shown. The average relative error in $N(y)$ of the last 500 epochs is 0.057602. Right: N(y) reference vs learnt.

Table 5
Relative errors for the diffusion-reaction problem, $\epsilon = 1/50$.

e_1	e_2	e_3	e_4
0.018582	0.011637	0.022184	0.0116435

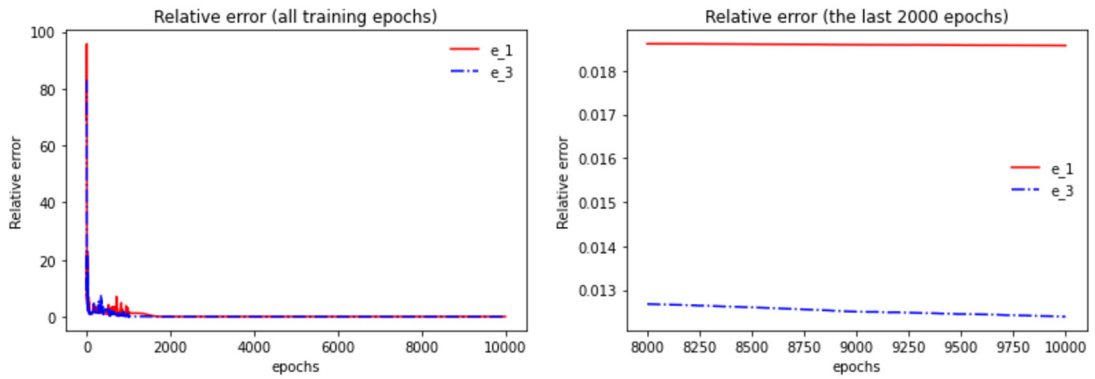

Fig. 22. Diffusion and reaction problem with $\epsilon = 1/50$, e_1 and e_3 relative error as a function of the training epochs. Left: history of all training epochs; right: history of the last 2000 epochs. The average relative errors of the last 500 epochs are: $e_1 = 0.012388$ and $e_3 = 0.022184$.

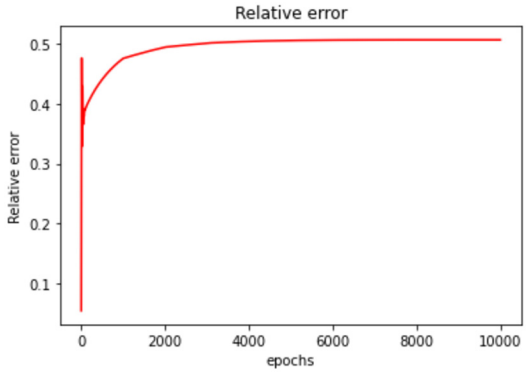

Fig. 23. Diffusion reaction equation with $\epsilon = 1/50$ transfer learning of classical PINN.

a potential alternative to the traditional numerical-driven homogenization. The transfer learning (check Fig. 23) is the same as before (with a relative error of 0.506884) and cannot give us a better result.

6. Conclusion

Multiscale problems widely exist in real life; to solve multiscale problems, fine-scale solvers are required, but the computational cost is very high. Researchers consider solving the multiscale problem with data-driven approaches. In this work, we propose to solve the multiscale problems by the physics-informed neural network (PINN) with the help of homogenization. We first find that the classical PINN cannot solve the multiscale problems. Homogenization is a 3-step PDE technique that is used to approximate the multiscale equation by a homogenized equation. We find that all homogenization steps can be implemented by the PINN accurately. In particular, we propose an oversampling strategy that greatly improves PINN accuracy for solving periodic problems; this technique is used in the first step of homogenization. We conduct several experiments and find that the accuracy of PINN is greatly improved by our method. We also observe that PINN-assisted homogenization is also an accurate approach to implementing homogenization; we hence conclude that the PINN is a potential alternative to numerical-driven homogenization.

CRediT authorship contribution statement

Wing Tat Leung: Conceptualization, Formal analysis, Investigation, Methodology, Writing – original draft. **Guang Lin:** Conceptualization, Funding acquisition, Supervision, Writing – review & editing. **Zecheng Zhang:** Conceptualization, Formal analysis, Investigation, Methodology, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

The authors gratefully acknowledge the support of the National Science Foundation (DMS-1555072, DMS-2053746, and DMS-2134209), Brookhaven National Laboratory Subcontract 382247, and U.S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research program DE-SC0021142).

Appendix A. Homogenization of the diffusion-reaction equation

In this section, we provide the details of the homogenizing of a diffusion-reaction equation. The problem is defined as:

$$\frac{\partial u_\epsilon}{\partial t} - D\nabla\cdot\nabla u_\epsilon + \frac{1}{\epsilon} r(x/\epsilon)u_\epsilon = f, \tag{A.1}$$

where $\int_Y r(y)dy = 0$ for the solvability; $Y = [0,1]^d$ is the unit cube. We seek the same asymptotic expansion as (5) and by equating the power of ϵ, we have:

$$-D\nabla\cdot\nabla u_0 = 0$$

We have u_0 is independent of y and we can then further simplify the ϵ^{-1} term,

$$-D\nabla\cdot\nabla u_1 = r_0 u_0,$$

where $u_1(x,y)$ is double periodic in y with period Y. The cell problem can then be defined as:

$$-D\nabla\cdot\nabla N(y) = -r(y),\ y\in Y. \tag{A.2}$$

The problem has the double periodic boundary condition, and we assume $N(y)$ has zero average in Y; the $\int_Y r(y)dy = 0$ guarantees the equation is solvable. u_1 can then be expressed as:

$$u_1(x,y) = N(y)u_0(x).$$

Finally for ϵ^0 term,

$$\frac{\partial u_0}{\partial t} - D\nabla\cdot\nabla u_0 - D\Delta_{xy}u_1 - D\nabla\cdot\nabla u_2 + r(t)u_1 = f(x).$$

Integrate the above equation over y in one period Y, we finally have,

$$\frac{\partial u_0}{\partial t} - D\nabla\cdot\nabla u_0 + r^* u_0 = f, \tag{A.3}$$

where the homogenized coefficient r^* is:

$$\int_Y r(y)N(y)dy.$$

References

[1] A. Bensoussan, J.L. Lions, G. Papanicolaou, Asymptotic Analysis for Periodic Structures, vol. 374, American Mathematical Soc., 2011.
[2] B. Chetverushkin, E. Chung, Y. Efendiev, S.M. Pun, Z. Zhang, Computational multiscale methods for quasi-gas dynamic equations, J. Comput. Phys. 440 (2021) 110352.
[3] E. Chung, Y. Efendiev, T.Y. Hou, Adaptive multiscale model reduction with generalized multiscale finite element methods, J. Comput. Phys. 320 (2016) 69–95.
[4] E. Chung, Y. Efendiev, W.T. Leung, S.M. Pun, Z. Zhang, Multi-agent reinforcement learning accelerated mcmc on multiscale inversion problem, arXiv preprint arXiv:2011.08954, 2020.
[5] E. Chung, Y. Efendiev, S.M. Pun, Z. Zhang, Computational multiscale methods for parabolic wave approximations in heterogeneous media, arXiv preprint arXiv:2104.02283, 2021.
[6] E. Chung, W.T. Leung, S.M. Pun, Z. Zhang, A multi-stage deep learning based algorithm for multiscale model reduction, J. Comput. Appl. Math. 394 (2021) 113506.
[7] E.T. Chung, Y. Efendiev, W.T. Leung, Constraint energy minimizing generalized multiscale finite element method, Comput. Methods Appl. Mech. Eng. 339 (2018) 298–319.
[8] E.T. Chung, Y. Efendiev, W.T. Leung, Z. Zhang, Cluster-based generalized multiscale finite element method for elliptic pdes with random coefficients, J. Comput. Phys. 371 (2018) 606–617.

[9] Y. Efendiev, L. Durlofsky, S. Lee, Modeling of subgrid effects in coarse-scale simulations of transport in heterogeneous porous media, Water Resour. Res. 36 (2000) 2031–2041.
[10] Y. Efendiev, J. Galvis, T.Y. Hou, Generalized multiscale finite element methods (gmsfem), J. Comput. Phys. 251 (2013) 116–135.
[11] Y. Efendiev, T.Y. Hou, Multiscale Finite Element Methods: Theory and Applications, vol. 4, Springer Science & Business Media, 2009.
[12] Y.R. Efendiev, T.Y. Hou, X.H. Wu, Convergence of a nonconforming multiscale finite element method, SIAM J. Numer. Anal. 37 (2000) 888–910.
[13] B. Engquist, P.E. Souganidis, Asymptotic and numerical homogenization, Acta Numer. 17 (2008) 147–190.
[14] E. Gildin, M. Ghasemi, A. Romanovskay, Y. Efendiev, Nonlinear complexity reduction for fast simulation of flow in heterogeneous porous media, in: SPE Reservoir Simulation Symposium, OnePetro, 2013.
[15] T.Y. Hou, X.H. Wu, A multiscale finite element method for elliptic problems in composite materials and porous media, J. Comput. Phys. 134 (1997) 169–189.
[16] G. Lin, Y. Wang, Z. Zhang, Multi-variance replica exchange stochastic gradient mcmc for inverse and forward Bayesian physics-informed neural network, arXiv preprint arXiv:2107.06330, 2021.
[17] L. Liu, T. Zeng, Z. Zhang, A deep neural network approach on solving the linear transport model under diffusive scaling, arXiv preprint arXiv:2102.12408, 2021.
[18] X. Ma, M. Al-Harbi, A. Datta-Gupta, Y. Efendiev, An efficient two-stage sampling method for uncertainty quantification in history matching geological models, SPE J. 13 (2008) 77–87.
[19] A.A. Pankov, G-Convergence and Homogenization of Nonlinear Partial Differential Operators, vol. 422, Springer Science & Business Media, 2013.
[20] P. Popov, G. Qin, L. Bi, Y. Efendiev, R.E. Ewing, J. Li, Multiphysics and multiscale methods for modeling fluid flow through naturally fractured carbonate karst reservoirs, SPE Reserv. Eval. Eng. 12 (2009) 218–231.
[21] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, A. Courville, On the spectral bias of neural networks, in: International Conference on Machine Learning, in: PMLR, 2019, pp. 5301–5310.
[22] 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.
[23] L. Tartar, Compensated compactness and applications to partial differential equations, in: Nonlinear Analysis and Mechanics: Heriot-Watt Symposium, 1979, pp. 136–212.
[24] S. Wang, H. Wang, P. Perdikaris, On the eigenvector bias of Fourier feature networks: from regression to solving multi-scale pdes with physics-informed neural networks, Comput. Methods Appl. Mech. Eng. 384 (2021) 113938.
[25] L. Yang, X. Meng, G.E. Karniadakis, B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data, J. Comput. Phys. 425 (2021) 109913.
[26] Z. Zhang, E.T. Chung, Y. Efendiev, W.T. Leung, Learning algorithms for coarsening uncertainty space and applications to multiscale simulations, Mathematics 8 (2020) 720.