DOMAIN-DECOMPOSED PHYSICS-INFORMED NEURAL NETWORK PREDICTION ON CARTESIAN CFD FRAMEWORK

Takashi Misaka[*1], **Yusuke Mizuno**[1], **Shogo Nakasumi**[2], **and Yoshiyuki Furukawa**[1]
[1]Industrial Cyber-Physical Systems Research Center,
National Institute of Advanced Industrial Science and Technology (AIST),
Koto-ku, Tokyo 135-0064, Japan
[2]Industrial Cyber-Physical Systems Research Center,
National Institute of Advanced Industrial Science and Technology (AIST),
Sakai, Fukui 919-0462, Japan

ABSTRACT

A domain decomposition approach for constructing physics-informed neural network (PINN) surrogate models in parallel is investigated using a multi-block Cartesian mesh computational fluid dynamics (CFD) framework called the Building Cube Method (BCM). The PINN is trained using the incompressible Navier-Stokes equations, and the results are compared with ANSYS Fluent predictions. Information exchange between adjacent subdomains is realized by transferring neural network parameters. In addition, objects in the domain are handled by the immersed boundary method. The results indicate that some parts of the flow field, such as the stagnation pressure, are difficult to reproduce and may need to be explicitly constrained by loss functions. Based on the results of the PINN predictions, we also consider overlapping PINN predictions and finite difference solutions such that the PINN surrogate is constructed for the discrepancy between observations and finite difference solutions in a data assimilation problem.

Keywords Physics-informed neural network · Domain decomposition · Cartesian mesh CFD framework

1 Introduction

Deep neural networks have been used in various fields, including scientific/engineering computations. Among these, the physics-informed neural network (PINN), which uses governing equations as constraints during training, has attracted much attention in recent years. [1, 2].

Since the PINN can be trained only by the initial/boundary conditions and governing equations, it is a method for obtaining approximate solutions to the governing equations, not just a surrogate model for approximating inputs and outputs. It reminds us that the PINN can be used in the same way as traditional numerical methods such as finite element, finite volume, and finite difference methods. In this context, a domain decomposition would be effective for large-scale problems, as also proposed for the PINN [3, 4]. It is also pointed out that the decomposition of the PINN alleviates the approximation problem in neural networks and improves the overall prediction accuracy [4].

We consider here an approach that uses a multi-block Cartesian mesh computational fluid dynamics (CFD) framework (Building Cube Method, BCM) [5] to construct the PINN for each multi-block domain that is processed in parallel. The mesh strategy of the BCM is shown schematically in Fig. 1, where multi-level subdomains called cubes contain the same number of mesh cells inside, and the local mesh resolution is defined by the cube size. Utilizing the simple data structure of the BCM, the PINN of each cube can be handled in the same way.

This paper attempts the PINN prediction for incompressible Navier-Stokes flows and compares the results with ANSYS Fluent [6]. We also consider overlapping PINN predictions and finite difference solutions such that the PINN

International Conference on Scientific Computing and Machine Learning 2024 (SCML2024)

[*]takashi.misaka@aist.go.jp

surrogate is constructed for the discrepancy between observations and finite difference solutions in a data assimilation problem. In the appendix, a more basic study using the Laplace equation in a simple three-dimensional domain is presented, and the prediction accuracy is assessed by comparing series solutions, finite difference solutions, and PINN solutions.

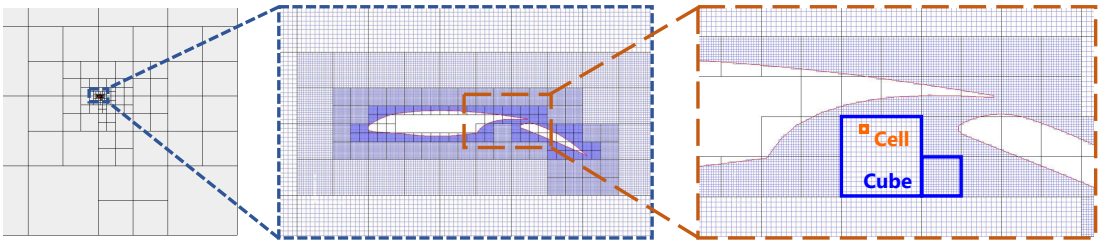

Figure 1: A schematic of mesh arrangement in the BCM framework.

2 Methodology

The PINN is composed of a fully connected network and a tanh activation function with spatial coordinates (x, y, z) as input and flow variables $(u, v, w, p, f_x, f_y, f_z)$ as output. The minimization of the loss function, consisting of the residual and boundary conditions of the governing equations, is performed by the Adam optimizer. The three-dimensional incompressible steady Navier-Stokes equations are considered to constrain the neural network. The mesh/data structure is based on the BCM framework in Fortran [5], and the PINN training in each cube is performed by PyTorch using Forpy [7] as Fortran/Python interface. Cube-wise parallelism is considered in the BCM framework [8, 9], where cubes can be assigned to available GPUs based on several criteria, including computational load. In this paper, parallel computation is performed using two GPUs.

In constructing the PINN on the BCM framework, information exchange between neighboring cubes is performed according to the extended PINN (XPINN) [3]. That is, the residuals and predictions of the governing equations are connected at the domain boundary by including the corresponding loss functions. Since these quantities can be computed from the network parameters of each subdomain, the network parameters are exchanged between neighboring cubes. The BCM framework allows neighboring cubes to be half or twice the size of the cube (see Fig. 1 for a two-dimensional case). To exchange the network parameters between cubes of different sizes, we define the interface's network parameters in the case of a small cube to a large cube. On the other hand, information exchange from a large to a small cube can be done by entering the appropriate coordinates.

An object embedded in the domain (a sphere or circular cylinder in this study) is represented by the immersed boundary method that constrains zero velocity at the object surface and adds a forcing term f_x, f_y, f_z in the vicinity of the object [10]. In addition, flow conservation is explicitly enforced by considering integral continuity planes in some sections of the domain [11].

The overlap between PINN predictions and finite difference solutions is also investigated, taking advantage of the ability of the PINN surrogate model constructed on the BCM framework. Aiming for the application to data assimilation problems, the PINN surrogate is constructed for the discrepancy between observations and finite difference solutions. That is, the flow variables $\boldsymbol{u} = (u, v, w, p)^T$ are decomposed as $\boldsymbol{u} = \boldsymbol{u}_{FDM} + \boldsymbol{u}_{PINN}$. By substituting the decomposed variables into the governing equations and assuming the $\boldsymbol{u}_{FDM}$ satisfies the governing equations in discrete form, we obtain the equations of $\boldsymbol{u}_{PINN}$ for the discrepancy between observations and finite difference solutions. This treatment facilitates the application of boundary conditions because they can be given by finite difference solutions. This approach takes full advantage of the BCM framework, allowing easy access to finite difference solutions.

3 Results

3.1 PINN on BCM framework

Figure 2(a) shows the computational domain consisting of a sphere of unit diameter and a box as the outer boundary. The Dirichlet boundary condition of $u = 1, v = w = 0$ is set for the boundaries except for the sphere surface ($u = v = w = 0$) and the outlet boundary. On the other hand, the pressure boundary condition of $p = 0$ is applied

only to the outlet boundary. For comparison, the ANSYS Fluent analysis is also performed with the same geometry and boundary conditions as the PINN prediction.

Figures 2(b)-(g) show the cross-sectional distributions around a sphere from the PINN and ANSYS Fluent predictions. The Reynolds number is set to 50 in these cases. The learning points of the PINN prediction are shown in Fig. 2(f), where the Cartesian mesh of the BCM framework with approximately 0.5 million points is used. The unstructured mesh in Fig. 2(g) is used for the ANSYS Fluent prediction, which also has approximately 0.5 million points. The solid lines in Fig. 2(b) and (d) indicate cube boundaries.

The overall velocity field is reproduced in the PINN prediction as shown in Fig. 2(b); however, there are discrepancies between the PINN and ANSYS Fluent predictions, such as the non-smoothness in the cube interface in the front of the sphere. In addition, the stagnation pressure is lower than that of ANSYS Fluent, as shown in Fig. 2(d), which should be explicitly constrained in the loss function.

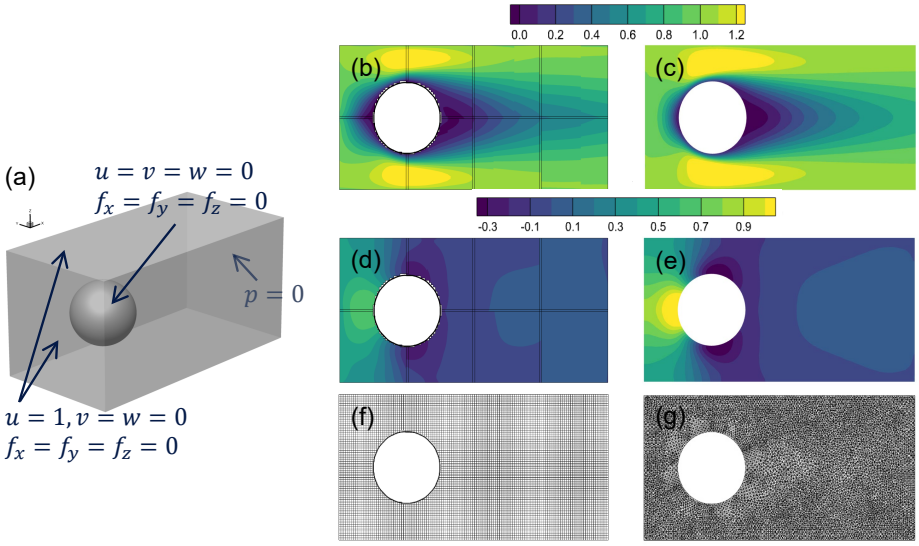

Figure 2: The computational domain consists of a sphere and outer boundary in (a). The comparison of PINN (b, d, f) and ANSYS Fluent (c, e, g) predictions for streamwise velocity (b, c), pressure (d, e) and computational mesh (f, g) around a sphere.

3.2 PINN superimposed on finite difference solution

As a data assimilation problem to be estimated by PINN surrogates, we consider the estimation of the macroscopic drag coefficient due to a porous medium embedded in the wake of the flow field. Figure 3(a) shows a cross-section of the computational domain with a unit-diameter circular cylinder embedded in a square duct. In this case, the Reynolds number is set to 200. The porous medium is defined within the spherical region of 0.5 diameter shown in the wake of the circular cylinder, as shown in the figure. The drag induced in the porous medium is modeled such that the drag force is proportional to the flow velocity. The coefficient controlling the strength of the porous drag, as well as the corresponding flow field, are estimated based on pseudo-observations extracted by thinning from the reference finite difference simulation.

Figures 3(a)-(d) show the comparison of cross-sectional velocity distributions obtained by the PINN superimposed on the finite difference solution and the reference finite difference solution. In this configuration, the velocity deficit and pressure disturbance due to the porous region are estimated by the PINN. Figure 3(e) shows the history of the porous parameter during PINN training. The porous parameter becomes close to the reference value of 10 in approximately 10^5 training steps.

4 Conclusions

In this study, we attempted the PINN prediction on the multi-block Cartesian mesh CFD framework. The PINN loss function was modified such that the differences in terms of the governing equation residual and the prediction at the

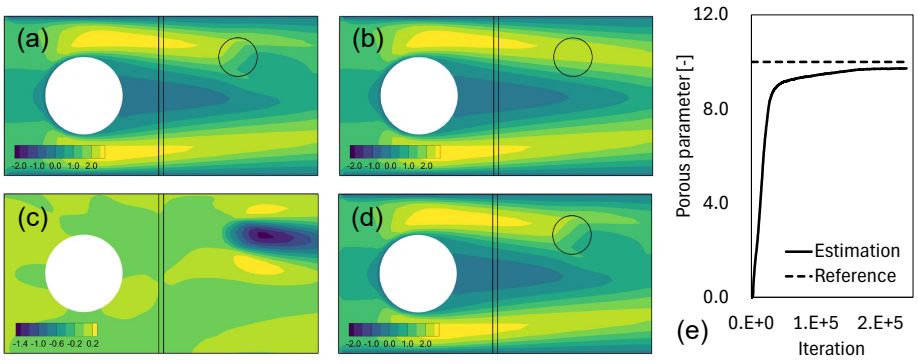

Figure 3: The comparison of streamwise velocity, where (a) the reference solution by the finite difference method, (b) the finite difference solution without a spherical porous region, (c) PINN solution after estimation of the porous region in the wake, (d) PINN superimposed on finite difference solution as a final estimate, (e) history of the porous parameter during PINN training.

domain boundaries become small. The interface communication between neighboring subdomains was performed by transferring neural network parameters. The results showed that the overall flow field around the sphere was reproduced; however, there were discrepancies from the ANSYS Fluent prediction, especially regarding the stagnation pressure. Based on the results of the PINN prediction, we also considered overlapping the PINN predictions and finite difference solutions such that the PINN surrogate is constructed for the discrepancy between observations and finite difference solutions in a data assimilation problem.

Acknowledgments

This work was supported by JSPS KAKENHI Grant-in-Aid for Scientific Research(C), Grant Number 23K11140.

References

[1] Raissi, M., Perdikaris, P., and Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. In *Journal of Computational Physics*, Vol. 378, pages 686–707, 2019.

[2] Shiratori, S., Nakamura, Y., and Sugihara, I. Trend in physics-informed neural networks and application to liquid film flows. In *Journal of the Japanese Society for Artificial Intelligence*, Vol. 38, pages 335–344, 2023.

[3] Shukla, K., Jagtap, A. D., and Karniadakis, G. E. Parallel physics-informed neural networks via domain decomposition. In *Journal of Computational Physics*, Vol. 447, page 110683, 2021.

[4] Moseley, B., Markham, A., and Nissen-Meyer, T. Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations. In *Advances in Computational Mathematics*, Vol. 49, page 62, 2023.

[5] Nakahashi, K. High-density mesh flow computations with pre-/post-data compressions. In *17th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2005–4876, 2005.

[6] ANSYS Fluent. https://www.ansys.com/products/fluids/ansys-fluent/.

[7] Forpy: A library for Fortran-Python interoperability. https://github.com/ylikx/forpy.

[8] Misaka, T., Sasaki, D., and Obayashi, S. Adaptive mesh refinement and load balancing based on multi-level block-structured Cartesian mesh. In *International Journal of Computational Fluid Dynamics*, Vol. 31, pages 476–487, 2017.

[9] Misaka, T. Space-time adaptive model order reduction utilizing local low-dimensionality of flow field. In *Journal of Computational Physics*, Vol. 493, page 112475, 2023.

[10] Huang, Y., Zhang, Z., and Zhang, X. A direct-forcing immersed boundary method for incompressible flows based on physics-informed neural network. In *Fluids*, Vol. 7, page 56, 2022.

[11] NVIDIA Modulus v22.09. https://docs.nvidia.com/deeplearning/modulus/modulus-v2209/user_guide/theory/recommended_practices.html.

Appendix: Accuracy Evaluation with Laplace Equation

The analysis of the Laplace equation is performed in a simple cubic domain (Fig. 4), and the prediction accuracy is evaluated by comparing series solutions, finite difference numerical solutions, and PINN solutions. The boundary conditions of the normalized computational domain are set as $\phi_s(x, y, 0.5) = 1$ on the top surface and $\phi_s(\pm 0.5, y, z) = \phi_s(x, \pm 0.5, z) = \phi_s(x, y, -0.5) = 0$ on other surfaces. In this case, the series solution of the three-dimensional Laplace equation is as follows, considering up to 100 modes.

$$\phi_s(x, y, z) = \frac{16}{\pi^2} \sum_{m,n=1}^{100} \frac{\sinh \gamma_{mn} z}{mn \sinh \gamma_{mn}} \sin m\pi x \sin n\pi y \tag{1}$$

where $\gamma_{mn} = \sqrt{(m\pi)^2 + (n\pi)^2}$. The finite difference numerical solution is obtained by the second-order central difference method. In the following discussion, parametric studies are performed concerning the number of layers and nodes in the network, the distribution of learning points (collocation points, residual points) used for evaluating the residuals of the governing equations, etc. The accuracy of the trained PINN is evaluated by the mean square error (MSE) from the series solution.

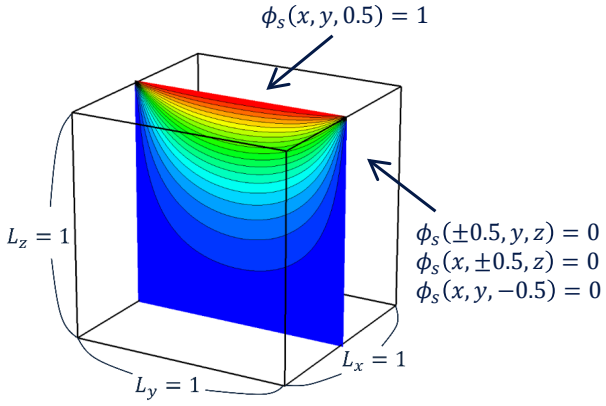

Figure 4: The computational domain for solving the Laplace equation.

Before studying the domain-decomposed PINN, the computation is performed in a single cubic computational domain. Figure 5(a) shows the history of the loss function and MSE during training for finite difference and PINN predictions. Figure 5(b) shows the influence of the number of layers and nodes in the network on the MSE. The MSE decreases as the number of layers and nodes increases; however, the decrease becomes modest starting at approximately three layers and 40 nodes. Figure 5(c) shows the influence of the weight w_{bc} applied for the boundary condition loss. In this example, it can be seen that the MSE becomes smaller when w_{bc} is set to approximately 1000. Usually, the distribution of learning points is based on a random number or a design of experiments that efficiently fills the space; however, the case of using the Cartesian mesh of the BCM as learning points is investigated. Figure 5(d) shows the effect of the learning point distribution. The MSE for an equally spaced Cartesian mesh (Cart36^3, etc.) becomes relatively small. Fig. 6 shows the distribution of the series solution in (a) and the error distributions of the finite difference and PINN solutions in (b) and (c), respectively. The magnitude of the MSE is smaller for the finite difference solution.

The results using the BCM framework are shown in Figs. 7 and 8, where the number of cubes is set to eight so that each process handles four cubes. In addition to the boundary condition weight w_{bc}, the XPINN considers the interface weight w_{if} between neighboring cubes. The effect of these weights is shown in Fig. 7(b). Figure 8 shows the error distributions of the finite difference and PINN solutions in (b) and (c), respectively. The PINN solution in the decomposed domain shows that the error is large near the domain boundaries. In addition, the overall error is more significant than in the single domain case.

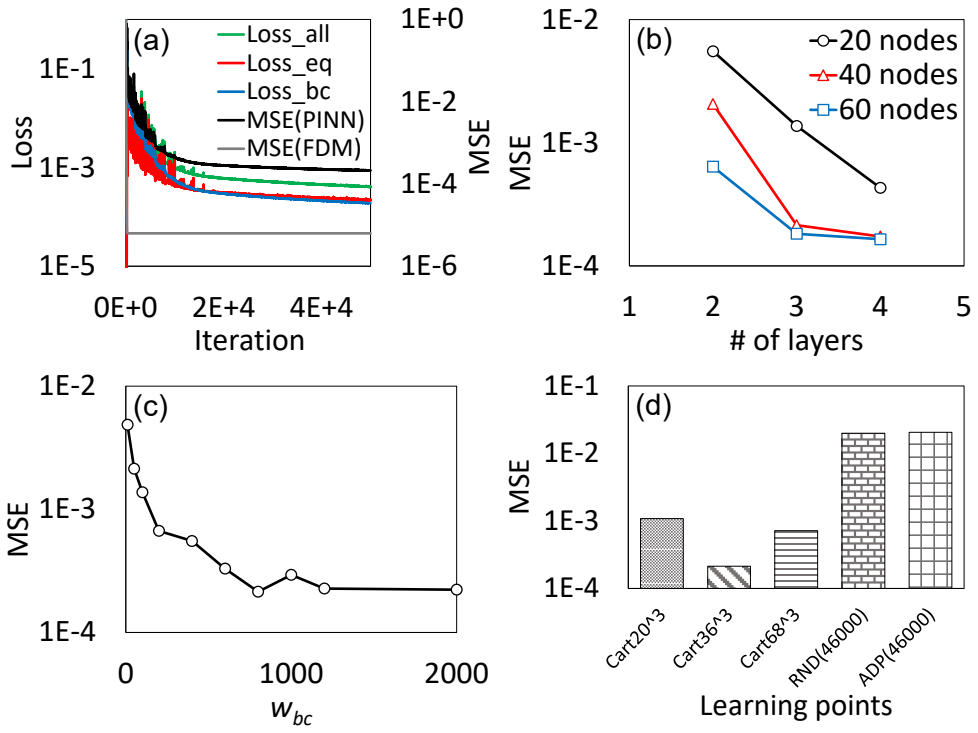

Figure 5: The computation in the single domain, (a) loss functions, and MSE trends by varying (b) network structure, (c) the boundary condition weight w_{bc} and (d) the distribution of learning points.

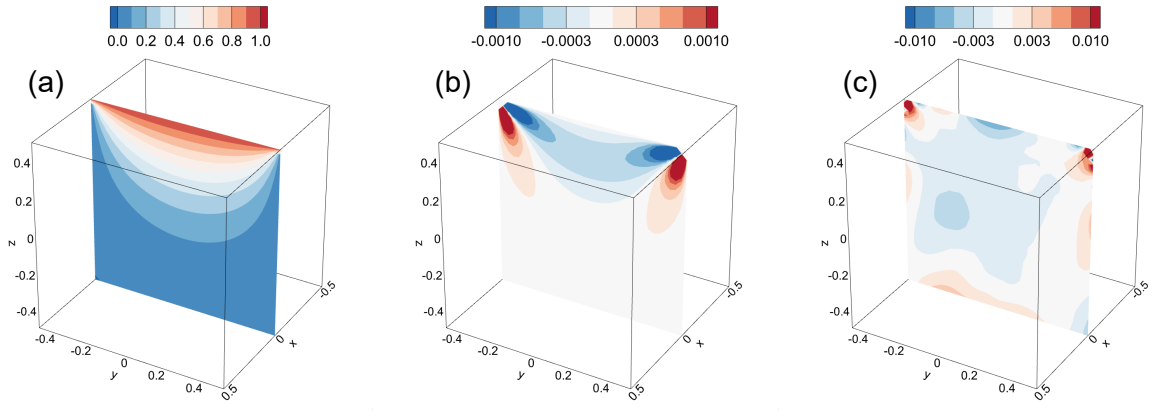

Figure 6: Series solution distribution of the single domain in (a), and the error distributions for finite difference and PINN solutions in (b) and (c), respectively.

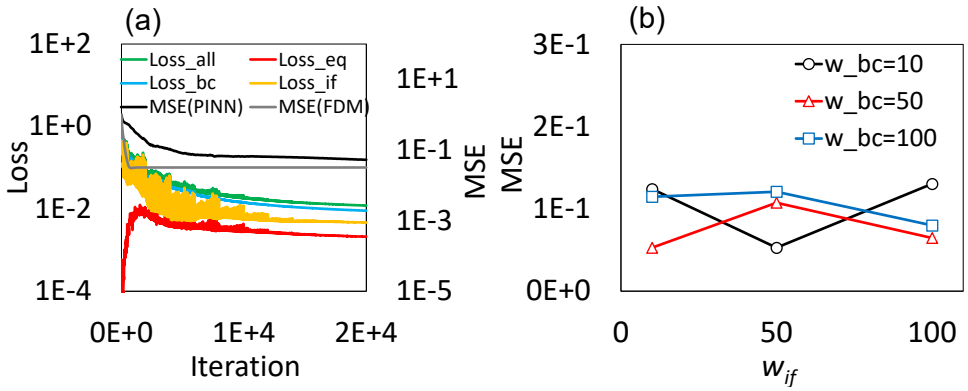

Figure 7: The computation in the decomposed domain, (a) loss functions, and MSE trends by varying (b) network structures, (c) the boundary condition weight w_{bc} and (d) the distribution of learning points, respectively.

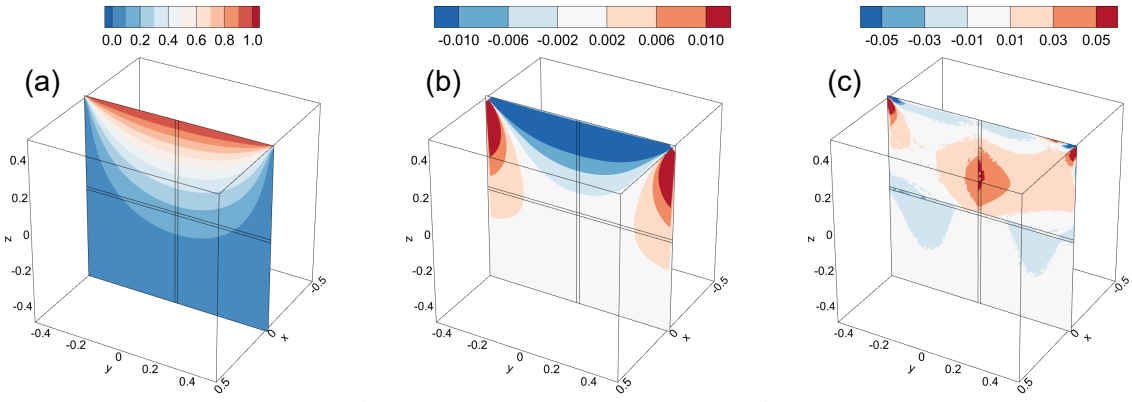

Figure 8: Series solution distribution of the decomposed domain in (a), and the error distributions for finite difference and PINN solutions in (b) and (c), respectively.