Mitigating Coordinate Transformation for Solving Partial Differential Equations with Physic-Informed Neural Networks

Hyo-Seok Hwang
School of Electrical Engineering
Korea University
Seoul, Korea
shdhkj960@korea.ac.kr

Suhan Son
School of Electrical Engineering
Korea University
Seoul, Korea
anaplasia29@korea.ac.kr

Yoojoong Kim
School of Computer Science and Information Engineering
The Catholic University of Korea
Seoul, Korea
yoojoongkim@catholic.ac.kr

Junhee Seok*
School of Electrical Engineering
Korea University
Seoul, Korea
jseok14@korea.ac.kr

***Abstract*—** In this work, we investigate some coordinate systems to solve partial differential equations (PDEs) using a neural network. We approximate the solution using physics-informed neural networks (PINNs) both before and after the coordinate transformation for two cases: a coordinate system with periodicity and without periodicity. We demonstrate that PINNs with Cartesian coordinate shows better approximation accuracy. This implies in PINNs training the Cartesian coordinate system is superior to the other coordinate systems derived by coordinate transformation. To the best of our knowledge, this is the first work to test training of PINNs by modifying PDEs according to the boundary shape.

Keywords—partial differential equation; deep learning; physics-informed neural network

I. Introduction

Partial Differential Equations (PDEs) are used to model numerous natural phenomena [1]. Although they are used in many applications, the number of PDEs that can be solved analytically is small, and most PDEs do not even know the existence of solutions [2]. Therefore, in practical applications, the solution is numerically calculated and applied where desired. However, as the number of points to seek numerical solutions increases, the computation time increases accordingly. Thus, methods for approximating the solutions of PDEs using various deep learning techniques have been proposed [2-6]. They have the advantage of being able to produce a solution with high accuracy even for the points that are not used to approximate the solution.

When solving PDEs analytically, coordinate transformation is performed appropriately according to the given boundary conditions [1]. For example, when a boundary condition is given as a circle in two dimensions, the PDE is transformed into a polar coordinate system and then solved analytically. In the case of three-dimension, if the boundary condition is given in the form of a cylinder, the PDE is transformed into a cylindrical coordinate and then solved analytically. However, it is very inconvenient and cumbersome to properly transform the PDE according to the shape of the boundary. In this paper, we investigate some coordinate systems to solve PDEs with neural network and show that an accurate approximation can be made even if they are not transformed according to the shape of the boundary. We tested our hypothesis in the case where the boundary has periodicity as well as the case where it does not have periodicity.

* Corresponding Author

II. Preliminaries

A. Partial Differential Equations

PDEs are differential equations that involve an unknown function of several independent variables and its partial derivatives. The PDEs are widely used to describe various physical phenomena such as wave, heat, diffusion, etc. One of the representative PDEs is Laplace's Equation: this equation is used to describe steady-state temperature distribution in thermodynamics. If the domain is [0, a] × [0, b], i.e., two-dimensional rectangular region, Laplace's equation has the following form:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{1}$$

The solution of Laplace's equation depends on the given boundary conditions. For example, in the above rectangular

region, the boundary conditions are given in the following form:

$$u(x,0) = f_{lower}(x) \quad u(x,b) = f_{upper}(x)$$
$$u(0,y) = f_{left}(y) \quad u(x,0) = f_{right}(y) \tag{2}$$

A problem involves Laplace's equation with boundary conditions on the given domain is called a Dirichlet problem [1].

B. Coordinate Transformation

The domain may simply be rectangular, but in many cases, there are various types of domains such as circular, cylindrical, spherical, etc. In this case, the solution of the Laplace's equation cannot be obtained analytically in the rectangular form. Therefore, the equation must be transformed into the corresponding coordinate system [1]. For instance, if the boundary is given as a circle with radius r and centered at the origin, the Laplace's equation is transformed into polar coordinates, and its form is as following:

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} = 0 \tag{3}$$

, where $x = r \cdot cos\theta$, $y = r \cdot sin\theta$. After such coordinate transformation, the Laplace's equation can finally be solved analytically.

C. Physics-Informed Neural Networks

Numerous number of data is required to train a neural network. However, in physics or engineering systems, it is often very difficult to collect dataset. Ref. [3, 5] presents physics-informed neural networks (PINNs), which overcome a small amount of data by using many formulas accumulated in these fields for neural network training. PDEs are one of the representative formulas available in these fields. To find a solution of a given PDE, PINNs use not only the training dataset but also the PDE itself to make the neural network to approximate the solution. It utilizes the auto-differentiation provided by several python deep learning packages. If the solution of a PDE is approximated by a neural network, the partial derivative with respect to the one input variable can be obtained by using the auto-differentiation. In this way, after organizing the given PDE, we can regularize the neural network to satisfy it. Since this PDE must be satisfied not only with the collected data, but also with any data in the domain, it is possible to use an arbitrary number of collocation points for regularization. Therefore, an accurate solution can be approximated with only a small amount of collected data. It trains a neural network by minimizing the error function expressed as follows: :

$$MSE = MSE(u) + MSE(f) \tag{4}$$

, where *MSE(u)* is the mean squared error (MSE) between exact solution given by training data and the predicted value of neural networks and *MSE(f)* is the mean squared error between the left-hand side and right-hand side of the PDE for randomly selected collocation points.

III. Problem Setup

To demonstrate that PINNs can properly approximate the solution of PDE without coordinate transformation, we will look at the following two settings for Laplace's equation: polar coordinates which have periodicity about the angle θ, and symmetric cylindrical coordinates which have no periodicity. In both cases, PDEs should be converted into appropriate coordinate systems to obtain solutions analytically. However, if PINNs accurately approximate the solution in both cases without coordinate transformation, there will be no need to define any other neural network after performing coordinate transformation in order to apply PINNs. In other words, it is possible to define PINNs only for Cartesian coordinates, which is the simplest form, and use them for boundaries of various shapes.

The polar form of the Laplace's equation is as given in (3).

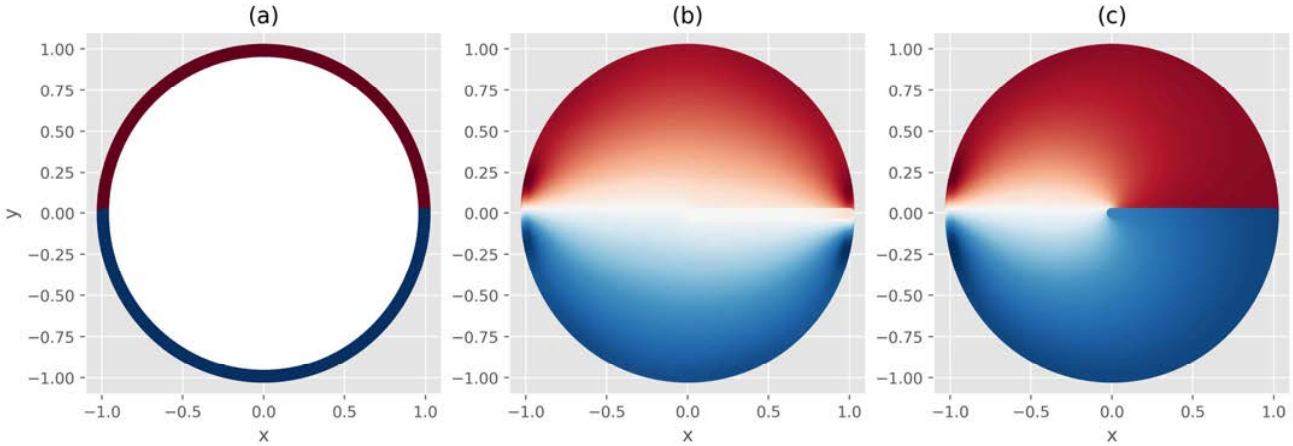

Fig. 1. Visualization of Dirichlet problem of polar coordinate in section IV.A. (a) is the boundary condition. Heat is constantly applied to the upper semicircle, and cooling is continuously applied to the lower semicircle. (b) is the solution approximated by PINNs with a Cartesian coordinate and (c) is the solution approximated by PINNs with a polar coordinate.

However, when calculating the solution numerically, r may become 0, so multiply both sides by the square of r then form the following equation:

$$r^2 \frac{\partial^2 u}{\partial r^2} + r \frac{\partial u}{\partial r} + \frac{\partial^2 u}{\partial \theta^2} = 0 \tag{4}$$

In the case of cylindrical coordinates, the Laplace's equation is transformed as follows:

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} + \frac{\partial^2 u}{\partial z^2} = 0 \tag{5}$$

However, what we are trying to do now is to show a case in which the coordinate system has no periodicity, unlike the former case. Therefore, we will look at the case where the boundary condition is independent of θ. In this case, Laplace's equation has a reduced from as follows:

$$r \frac{\partial^2 u}{\partial r^2} + \frac{\partial u}{\partial r} + r \frac{\partial^2 u}{\partial z^2} = 0 \tag{6}$$

Similar to the polar coordinate system, this is the form in which both sides are multiplied by r because r can be zero.

We use 100 boundary points and 10,000 collocation points to train the former problem setup. For letter problem setup, we use 800 boundary points and 100,000 collocation points to train the PINNs because in the former case, the dimension of the input is two, whereas in the latter case, the dimension of the input is three. The architecture of the PINNs which take two input variables is three-layer multi-layer perceptron with 20 neurons per hidden layer and the architecture of the PINNs that take three input variables is four-layer multi-layer perceptron. All PINNs are trained to minimize the (4) using the L-BFGS optimizer.

IV. Experiments

A. Polar Coordinates

First, to examine the case where the coordinate system has periodicity, PINNs are trained to approximate the solution of Laplace's equation given the boundary conditions as follows:

$$u(1,\theta) = \begin{cases} 1, & if\ \theta \in [0,\pi] \\ 0, & if\ \theta \in (\pi, 2\pi) \end{cases} \tag{7}$$

This boundary condition is taken from the example in [1]. Looking at Fig. 1(a) this is a case where 100 degrees Celsius is applied to the upper part of the circular room and 0 degrees Celsius is applied to the lower part. When this boundary condition is applied, isothermal lines are created in order from the upper semicircle to the lower semicircle, i.e., the solution as the form of Fig. 1(b) should appear. On the other hand, as can be seen in Fig. 1(c), the PINNs was not properly approximate the solution with polar coordinate, a form in which the solution could be obtained analytically. The first row of Table 1 shows the MSE between each PINNs and the exact numerical solution. The PINNs with polar coordinates has an MSE of 0.691, whereas the PINNs with Cartesian coordinates has an MSE of 0.017. From this, we can find that if the coordinate system has periodicity, it is better to train PINNs with Cartesian without coordinate transformation. Someone think that this result may be natural because the neural network does not learn the periodicity of the function that it wants to approximate, but when we check the solutions obtained analytically, both Cartesian and Polar coordinate have periodic solutions. This is beyond the scope of this paper, so we will stop here and recommend that look for other materials if you are interested.

B. Symmetric Cylindrical Coordinates

Second, to examine the case where the coordinate system has not periodicity, PINNs are trained to approximate the solution of Laplace's equation given the boundary conditions as follows:

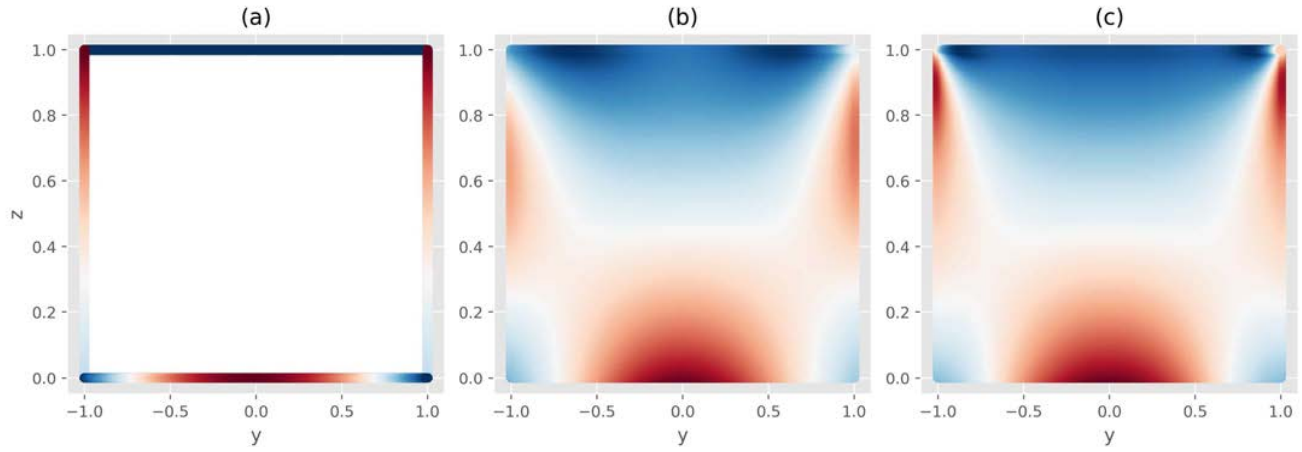

Fig. 2. Visualization of Dirichlet problem of cylindrical coordinate in section IV.B when x is zero. (a) is the boundary condition. Heat is constantly applied to the base circle and the upper part of the side. (b) is the solution approximated by PINNs with a Cartesian coordinate and (c) is the solution approximated by PINNs with a symmetric cylindrical coordinate.

$$u(r,\theta,z) = \begin{cases} 1-r^2, & if\ z = 1 \\ 0, & if\ z = 0 \end{cases} \quad (8)$$

$$u(1,\theta,z) = \exp\{z\ - 1\} \quad (9)$$

Fig. 2(a) is the graph of the boundary condition in *yz*-plane. The heat is applied to the base circle and the upperpart of the side. Fig. 2(b) and Fig. 2(c) is the solution trained by the PINNs with Cartesian coordinate and Cylindrical Coordinate, respectively. The second row of Table 1 shows the MSE between each PINNs and the exact numerical solution. The PINN with cylindrical coordinates has an MSE of 0.187 while the PINN with Cartesian coordinates has an MSE of 0.037. From this, we can demonstrate that the PINN with Cartesian coordinates approximate better even in coordinate systems that have no periodicity, similar to the case of having periodicity. More noteworthy, in the PINN with cylindrical coordinates, due to symmetry, the number of input variables was two, whereas the PINN with Cartesian coordinates was three. This implies that even if the number of dimensions is reduced, it may still have better performance using Cartesian coordinates.

V. Related works

Several techniques using deep learning to solve PDEs have been proposed [2] and two of them are [6] and [5]. Ref. [6] presents the limitations of solving the PDEs numerically in the high-dimensional domains, and then proposes a method for solving the high-dimensional PDEs using deep learning: it converts the corresponding PDE into a related stochastic differential equation, and then approximates the gradient of the unknown solution using a neural network. Ref. [5] is to approximate the neural network to satisfy both the current data and PDEs, as discussed earlier. Ref. [5] has been applied in many fields: stochastic-diffusion-reaction equations [7], wave equations [8], fluid dynamics [9, 10], fractional advection-diffusion equations [11], etc.

VI. Conclusion

In this paper, we investigated some coordinate systems when approximating the solution of a PDE using a neural network. We showed that the Cartesian coordinate systems has better approximation performance than the other systems. This means that there is no need to apply the coordinate transformation, a technique used to find solutions analytically.

TABLE I. The Mean square Error between PINNs and Exact solutions

	Cartesian PINNs	***Transformed PINNs***
Polar Coor.	**0.017**	0.691
Cylindrical Coor.	**0.037**	0.187

Moreover, if you want to use a different coordinate system, you may need to apply additional techniques. We tested the hypothesis on the polar coordinate system in which the coordinate system has periodicity as well as the symmetric cylindrical coordinate system in which the coordinate system has no periodicity. The PINNs were used to utilize the boundary conditions of the PDE. Experiments demonstrated that PINNs with Cartesian coordinates approximate the solution with higher accuracy in both cases. In addition, in the case of symmetric cylindrical boundary, PINN had higher accuracy with the Cartesian coordinates than with symmetric cylindrical coordinates, although the number of dimensions was higher. We are confident that these experimental results will provide good guidance when applying PINN to other PDEs as well as Laplace's equation for various boundary shapes.

Acknowledgment

This work was supported by a grant from the National Research Foundation of Korea (NRF-2022R1A2C2004003)

References

[1] Asmar, N.H.: 'Partial differential equations with Fourier series and boundary value problems' (Courier Dover Publications, 2016. 2016)

[2] Blechschmidt, J., and Ernst, O.G.: 'Three ways to solve partial differential equations with neural networks—A review ′ , GAMM-Mitteilungen, 2021, 44, (2), pp. e202100006

[3] Raissi, M., Perdikaris, P., and Karniadakis, G.E.: 'Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations', arXiv preprint arXiv:1711.10561, 2017

[4] Raissi, M., Perdikaris, P., and Karniadakis, G.E.: 'Numerical Gaussian processes for time-dependent and nonlinear partial differential equations', SIAM Journal on Scientific Computing, 2018, 40, (1), pp. A172-A198

[5] 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', Journal of Computational physics, 2019, 378, pp. 686-707

[6] Han, J., Jentzen, A., and Weinan, E.: 'Solving high-dimensional partial differential equations using deep learning', Proceedings of the National Academy of Sciences, 2018, 115, (34), pp. 8505-8510

[7] Chen, X., Duan, J., and Karniadakis, G.E.: 'Learning and meta-learning of stochastic advection–diffusion–reaction systems from sparse measurements', European Journal of Applied Mathematics, 2021, 32, (3), pp. 397-420

[8] Moseley, B., Markham, A., and Nissen-Meyer, T.: 'Solving the wave equation with physics-informed deep learning', arXiv preprint arXiv:2006.11894, 2020

[9] Lye, K.O., Mishra, S., and Ray, D.: 'Deep learning observables in computational fluid dynamics', Journal of Computational Physics, 2020, 410, pp. 109339

[10] Mao, Z., Jagtap, A.D., and Karniadakis, G.E.: 'Physics-informed neural networks for high-speed flows', Computer Methods in Applied Mechanics and Engineering, 2020, 360, pp. 112789

[11] Pang, G., Lu, L., and Karniadakis, G.E.: 'fPINNs: Fractional physics-informed neural networks', SIAM Journal on Scientific Computing, 2019, 41, (4), pp. A2603-A2626