Inverse problem of the spatially-dependent thermal conductivity in steady-state equation

Mai Thanh Trung

2024/03/25

1 Background of the task

The objective of this task is to determine the parameter of the black box solver of the steady-state heat equation, given a set of boundary conditions, load terms and material properties, which collectively form:

$$-\nabla \cdot h(x,y)\nabla T = q \tag{1}$$

where $h(x,y)$ is the thermal conductivity and q is the heat flux. In this study, the thermal conductivity $h(x,y)$ is the target non-scalar parameter we aim to approximate.

This task is a so-called "inverse problem" which is an active area of research in many engineering, physics and biology problems. For nonlinear problems, often an analytical solution is not available. In such cases, several methods have been employed to approximate the unknown parameters. Classical methods for inverse problems are usually iterative methods to determine the unknown parameters [1, 2, 3]. While these method can be effective, these methods can be computationally expensive and could have convergence issues [4, 5].

In recent years, numerous physics-based machine learning methods have been employed to solve a variety of problems. Gaussian processes have been particularly useful to model and estimate parameters in systems, where the dynamics are complex or poorly understood [6, 7, 8]. Many different architectures of neural network have also been used to tackle inverse problems as well. These range from fully-connected deep neural networks [4, 9, 10], generative adversarial network [11, 12] to graph networks [10].

This study will focus on the implementation of a specific neural network method, the Physics-Informed Neural Network (PINN) [13]. In recent years, PINNs have shown to be effective at modelling complex physical systems, and have shown capability in determining unknown properties and quantities, including solution inverse problems [14, 13, 5].

2 PINN for inverse problem

2.1 Modelling the non-linear heat equation

For the current study, the heat equation will be modelled using a finite element solver on a unit square domain. The thermal conductivity was modelled as

$$h(x,y) = 1 + 6x^2 + \frac{x}{1+2y^2} \tag{2}$$

with Dirichlet conditions of $300K$ on the bottom ($y = 0$) and the other boundary walls were considered as insulated i.e. Neumann conditions. This model has been solved using the FEniCS software (FEniCS Project, 2019).

2.2 PINN implementation

For the current problem, the thermal conductivity will be determined using PINN. This approach has been used in several studies to approximate the varying coefficients in the Poisson equation and Darcy's flow [14, 5].

In this problem two networks were used, one to approximate the temperature field, and another one to approximate the thermal conductivity. The heat equation was defined in its non-dimensional form as:

$$-\nabla \cdot h(x, y)\nabla T^* = \frac{q}{T_0 - T_1} \tag{3}$$

where the temperature is non-dimensionalised as $T^* = \frac{T-T_0}{T_0-T_1}$, and the non-dimensional length would be considered as 1.

Both networks used a fully-connected network with a width of 50 and a depth of 3, and the activation function used was *tanh* which is a common choice for training PINN to represent nonlinear functions. The PINN was trained by optimizing the loss function, which consist of three terms. The first term is the loss on the data $\mathcal{L}_{data}$ which is calculated as the mean square error between the temperature model prediction and the measurement data used for training the networks.

$$\mathcal{L}_{data} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{u}^n_{data} - \mathbf{u}^n_{PINN})^2 \tag{4}$$

The second term, the PDE loss, was calculated by applying the chain rule to differentiate functions' composition using automatic differentiation. The physics loss function, $\mathcal{L}_{PDE}$, can be calculated as the mean square error of residuals (res) of the heat equation:

$$\mathcal{L}_{PDE} = MSE(res, 0) \tag{5}$$

Lastly, with the boundary conditions known, the boundary loss was calculated as the mean square error across all boundaries. The Neumann conditions have been enforced by calculating the derivatives on the boundary using automatic differentiation.

$$\mathcal{L}_{BC} = \frac{1}{M}\sum_{m=1}^{M}(\mathbf{u}^m_{BC} - \mathbf{u}^m_{PINN})^2 \tag{6}$$

Thus, the total loss function that is being optimized has the form as follows

$$\mathcal{L}_{tot} = \lambda_1\mathcal{L}_{PDE} + \lambda_2\mathcal{L}_{data} + \lambda_3\mathcal{L}_{BC} \tag{7}$$

where λ_{1-3} are the weighting coefficients that determine the contribution of physics, measurement data, and BCs, respectively, to the loss function. In the current implementation, the weight λ_{1-3} were selected as $(1, 5, 1)$.

The Adam optimization algorithm was used to train the network with a learning rate of 10^{-3} and mini-batch of 1024 based on recommendations in Wang et al., 2023 [15].

3 Results and Discussion

The neural network was trained on a NVIDIA GeForce GTX 1050Ti for $5e10^5$ iterations. The results of the regressed temperature field and the thermal conductivity are presented in Figures 1 and 2

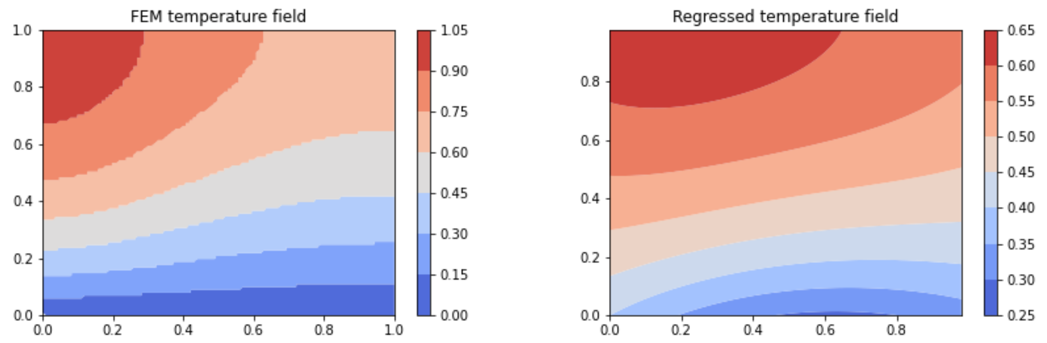

Figure 1: The temperature field (non-dimensional) for the steady-state heat equation solved using FEM (left) and regressed using PINN (right)

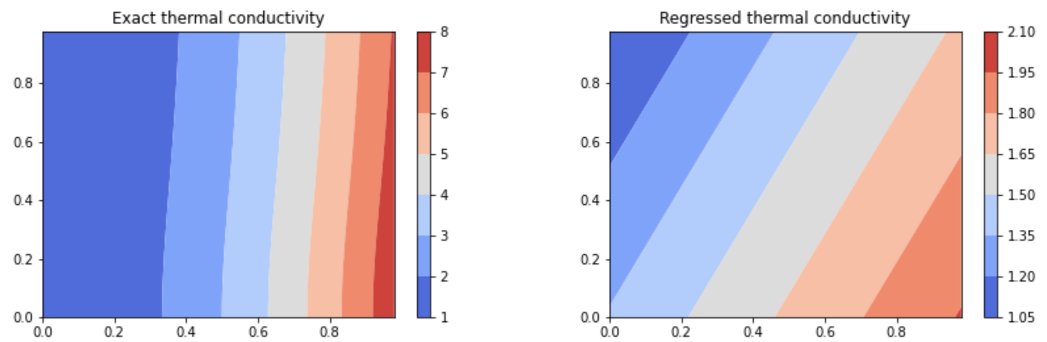

Figure 2: The thermal conductivity exact solution from the equation 2 (left) and regressed from the PINN (right)

However, it is evident that the PINN was unable to accurately reach a solution for both the temperature field and thermal conductivity. The mean absolute relative error on temperature is 50.91%, and on thermal conductivity is 41.94%. The difference plot can be seen in the Figure 3

Training PINNs has proven to be a challenging task in several studies, mainly in nonlinear cases [16, 17, 18, 19]. Therefore, it is necessary to identify the possible bottlenecks that could hinder the PINN to train effectively.

Loss function weights

In the current implementation, the weights were manually tuned by adjusting the weight based on the convergence of the loss function. As a result, the weighting (1, 5, 1) has been chosen as it allows the loss function to converge effectively.

However, setting these static weights can result in one loss term converging faster than the others, leading to loss imbalances and difficulty for other loss terms to converge to a solution. This could be mitigated by applying adaptive weights in order to take into account all the loss terms equally [19, 15, 20].

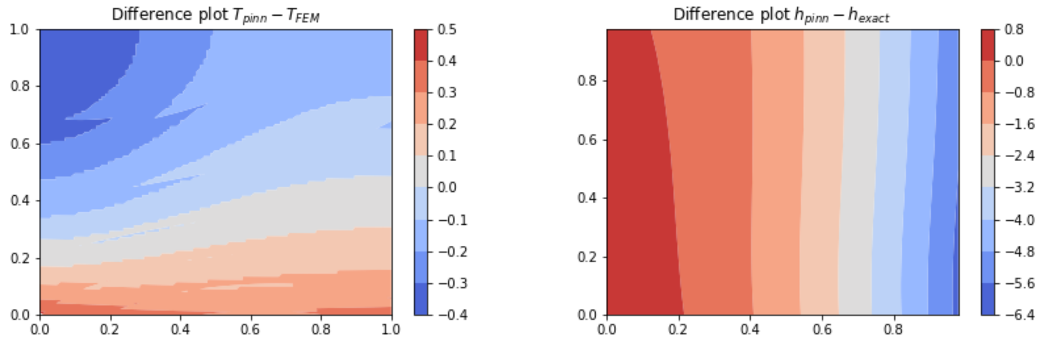

Figure 3: The difference on temperature field plot between the PINN and the FEM results (left), and thermal conductivity between the PINN and the exact solution (right)

Optimization algorithm

In our current implementation, the Adam method, a common choice for PINNs, was used as the optimization algorithm. However, several studies have suggested that it can lead to difficulty to converge to a correct solution of the PDE [5, 18].

Several studies recommend the use of L-BFGS or a combination of Adam and L-BFGS to achieve convergence [5, 18, 17]. Thus, a more in-depth investigation of optimization algorithms would be necessary to choose an appropriate optimization method.

Other parameters to consider

In our current study, the network size was chosen to be 50 neurons wide and 3 layers deep, which was a common choice for several studies on inverse problems using PINN [5, 14]. Although Wang et al. [15] suggest that wider (with a minimum of 128 neurons) would yield more optimal and robust solutions, due to hardware limitations, wider networks, which have more parameters, would results in a much longer training times.

Furthermore, in this study, the activation function $Tanh$ is commonly used for PINN. Some other options also include the $Sigmoid$ or the $swish$ function [13, 19]. However, there is a lack of general consensus on which activation function would be most appropriate.

Lastly, applying other training methods, such as random Fourier embeddings or curriculum training methodologies, to the network could significantly enhance the PINN's ability to learn the physics [15, 16, 17].

4 Conclusion

This study aimed to address the inverse problem using PINNs. The parameter that was being estimated was the thermal conductivity parameter in the steady-state heat equation. However, the training of the PINN proved to be challenging, as it was unable to accurately estimate both the temperature field and thermal conductivity.

Potential bottlenecks have been identified, specifically the weighting of the loss function and the optimization algorithm. These factors could potentially hinder the successful application of PINNs in solving inverse problems. Therefore, further exploration and understanding of these limitations are necessary for the future application of PINNs in solving inverse problems.

References

[1] A. Yıldırım, "Variational iteration method for inverse problem of diffusion equation," *International Journal for Numerical Methods in Biomedical Engineering*, vol. 26, no. 12, pp. 1713–1720, 2010.

[2] J. Baumeister and A. Leitão, "On iterative methods for solving ill-posed problems modeled by partial differential equations," *Journal of Inverse and Ill-Posed Problems*, vol. 9, no. 1, pp. 13–29, 2001.

[3] C. L. Byrne, *Iterative optimization in inverse problems.* CRC Press, 2014.

[4] K. Xu and E. Darve, "The neural network approach to inverse problems in differential equations," *arXiv preprint arXiv:1901.07758*, 2019.

[5] D. N. Tanyu, J. Ning, T. Freudenberg, N. Heilenkötter, A. Rademacher, U. Iben, and P. Maass, "Deep learning methods for partial differential equations and related parameter identification problems," *Inverse Problems*, vol. 39, no. 10, p. 103001, 2023.

[6] Y. Chen, B. Hosseini, H. Owhadi, and A. M. Stuart, "Solving and learning nonlinear pdes with gaussian processes," *Journal of Computational Physics*, vol. 447, p. 110668, 2021.

[7] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Numerical gaussian processes for time-dependent and nonlinear partial differential equations," *SIAM Journal on Scientific Computing*, vol. 40, no. 1, pp. A172–A198, 2018.

[8] J. Zhang, S. Zhang, and G. Lin, "Pagp: A physics-assisted gaussian process framework with active learning for forward and inverse problems of partial differential equations," *arXiv preprint arXiv:2204.02583*, 2022.

[9] L. Yuan, Y.-Q. Ni, X.-Y. Deng, and S. Hao, "A-pinn: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations," *Journal of Computational Physics*, vol. 462, p. 111260, 2022.

[10] T. Fan, K. Xu, J. Pathak, and E. Darve, "Solving inverse problems in steady-state navier-stokes equations using deep neural networks," *arXiv preprint arXiv:2008.13074*, 2020.

[11] M. Gillhofer, H. Ramsauer, J. Brandstetter, B. Schäfl, and S. Hochreiter, "A gan based solver of black-box inverse problems," in *NeurIPS 2019 Workshop on Solving Inverse Problems with Deep Networks*, 2019.

[12] S. Krylov and V. Krylov, "Inverse problem solving approach using deep network trained by gan simulated data," in *Journal of Physics: Conference Series*, vol. 1727, no. 1. IOP Publishing, 2021, p. 012002.

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

[14] I. The MathWorks, "Inverse problems using pinns," https://github.com/matlab-deep-learning/Inverse-Problems-using-Physics-Informed-Neural-Networks-PINNs, 2023.

[15] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, "An expert's guide to training physics-informed neural networks," *arXiv preprint arXiv:2308.08468*, 2023.

[16] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, "Characterizing possible failure modes in physics-informed neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 26 548–26 560, 2021.

[17] S. Basir and I. Senocak, "Critical investigation of failure modes in physics-informed neural networks," in *AiAA SCITECH 2022 Forum*, 2022, p. 2353.

[18] P. Rathore, W. Lei, Z. Frangella, L. Lu, and M. Udell, "Challenges in training pinns: A loss landscape perspective," *arXiv preprint arXiv:2402.01868*, 2024.

[19] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[20] R. Bischof and M. Kraus, "Multi-objective loss balancing for physics-informed deep learning," *arXiv preprint arXiv:2110.09813*, 2021.