Extending the Physics Informed Neural Network Model to Multigroup Neutron Diffusion Problems

Mohamed Elhareef and Zeyun Wu

Department of Mechanical and Nuclear Engineering, Virginia Commonwealth University, Richmond VA 23284
elhareefmh@vcu.edu; zwu@vcu.edu

INTRODUCTION

Physics Informed Neural Network (PINN) [1] is an advanced data science technique for solving partial differential equations (PDE) by employing a Neural Network (NN) model to approximate the solution. PINN approach takes advantage of the recent development in Machine Learning (ML) toolbox and applies Automatic Differentiation (AD) approach [2] on the NN model with respect to the independent variables to construct a PINN model based on the residuals of the PDE model. The shared learnable parameters between the two models are learnt by minimizing the PDE residuals within the solution domain and satisfying the boundary and/or initial conditions of the PDE. In this view, the physical laws are integrated in the PINN framework as a constraint on the model outputs.

PINN was successfully applied to various scientific and engineering applications. The overarching advantage of PINN is its capability of obtaining mesh-free solution for nonlinear PDE without any prior assumptions or approximation to the unknowns. One drawback of PINN is the computational complexity of the problems especially for high-dimensional systems may limit its applications.

In the previous work [3, 4], we adopted the PINN for the neutron-diffusion models for nuclear reactor applications. We started by solving the steady state two dimensional (2D), one energy group (1G), fixed source diffusion model for the Loosely-Coupled Reactor Model (LCRM) [3]. We then extended the PINN model to the *k*-eigenvalue mode diffusion model [4]. In this work we are extending PINN diffusion model to the two group (2G) cases. In particular, we are solving the fixed-source 2G diffusion equations defined by:

$$\begin{cases} -\nabla(D_1\nabla\phi_1) + \Sigma_{r,1}\phi_1 = \nu\Sigma_{f,2}\phi_2 + Q_1 \\ -\nabla(D_2\nabla\phi_2) + \Sigma_{a,2}\phi_2 = \Sigma_{s,1\to 2}\phi_1 \end{cases}, \qquad (1)$$

where ϕ_g, D_g, $\Sigma_{a,g}$, and $\Sigma_{f,g}$ (g=1, 2) stand for the flux, the diffusion coefficient, the macroscopic absorption cross section, and the macroscopic fission cross section of the group- g neutrons, respectively; $\Sigma_{s,1\to 2}$ stands for the macroscopic down-scattering cross-section of the fast-group neutrons, and $\Sigma_{r,1} = \Sigma_{a,1} + \Sigma_{s,1\to 2}$ is the removal cross section for the fast group neutrons, and Q_1 is the fixed neutron source, representing the fast neutrons generated from fission.

We demonstrate the applicability of PINN to the fixed-source 2G problems by solving the homogeneous quarter-core test example provided in Ref. [5] with the material and geometry configurations shown in Fig. 1. The corresponding two group material properties are summarized in TABLE I. The boundary conditions of the example problem are given as follows, in which zero flux boundaries are imposed on the left and bottom edges, while the reflective boundaries are assumed on the right and top sides:

$$\text{At surface } x = 0: \quad \phi_g(0, y) = 0\,, \qquad (2)$$

$$\text{At surface } y = 0: \quad \phi_g(x, 0) = 0\,, \qquad (3)$$

$$\text{At surface } x = 350: \quad \frac{\partial}{\partial x}\phi_g(350, y) = 0\,, \qquad (4)$$

$$\text{At surface } y = 350: \quad \frac{\partial}{\partial y}\phi_g(x, 350) = 0\,. \qquad (5)$$

TABLE I. Materials Properties of the Test Example [5].

	Material 1	Material 2
$D_1\ [cm]$	1.269	1.31
$D_2\ [cm]$	0.9328	0.8695
$\Sigma_{a,1}\ [cm^{-1}]$	7.86E-4	0
$\Sigma_{a,2}\ [cm^{-1}]$	4.1E-3	2.117E-4
$\Sigma_{s,1\to 2}\ [cm^{-1}]$	7.368E-3	1.018E-2
$\nu\Sigma_{f,2}\ [cm^{-1}]$	4.562E-3	0
$Q_1\ [n \cdot cm^{-3}s^{-1}]$	0.01	0

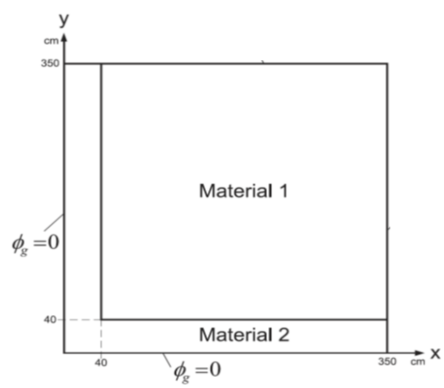

Fig. 1. Geometry of the test example [5].

METHOD

As mentioned earlier, the key idea behind the PINN approach is to construct a NN model approximating the solution of the PDE, and then employ AD technique to construct a NN model for the residuals of the PDE. Within the PINN framework, a gradient-descent-based algorithm is used to train the NN model by minimizing a loss function that accounts for both the residuals and other physical constraints such as boundary conditions. The trained model can be used for making predictions about the system state (i.e., the dependent variable) within the solution domain. A general description of the PINN framework applied to neutron diffusion models can be found in Ref. [3], and thus will not be repeated here. The needed modifications in the PINN model in the extension of multigroup problems, specifically for the 2G test example, are detailed in the following section.

To apply the PINN to the 2G neutron diffusion model, we construct a NN model with two inputs and two outputs:

$$\begin{bmatrix} \phi_1(x,y) \\ \phi_2(x,y) \end{bmatrix} = NN(x,y) \, . \tag{6}$$

We proceed by defining the residuals NN model by differentiation the NN based on Eq. (1)

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = -\nabla \begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} \nabla NN + \begin{bmatrix} \Sigma_{r,1} & -\nu\Sigma_{f,2} \\ -\Sigma_{s,1\to 2} & \Sigma_{a,2} \end{bmatrix} NN + \begin{bmatrix} Q_1 \\ 0 \end{bmatrix} \, . \tag{7}$$

In a similar manner, we construct a NN model at each boundary based on the corresponding boundary condition given Eqs. (2)-(5). For instance, the residuals at the top edge ($y = 350$) is defined as:

$$\begin{bmatrix} f_{T1} \\ f_{T2} \end{bmatrix} = \frac{\partial}{\partial y} NN(x, 350) \tag{8}$$

The next step in the PINN framework is to define a loss function that sums up the mean-squares of all the model residuals. In other words, the loss function measures the deviation of the NN prediction for the target values (i.e., zero residuals). By minimizing the loss function, the model predictions converge towards the solution of the PDE system. One practical form of the loss function is defined as follows

$$loss_1 = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| \begin{matrix} f_1(x_i^f, y_i^f) \\ f_2(x_i^f, y_i^f) \end{matrix} \right|^2 + \frac{1}{N_b} \sum_{j=1}^{N_b} \left| \begin{matrix} f_{T1}(x_j^T, y_j^T) \\ f_{T2}(x_j^T, y_j^T) \end{matrix} \right|^2 + \cdots \tag{9}$$

Due to the complexity of this joint-learning task, we added an additional term in the loss function representing the physical constraint for positive fluxes

$$\phi_g(x,y) \geq 0 \tag{10}$$

This physical constraint was integrated in the learning task to speed-up the convergence. The additional loss term has the following form:

$$loss_2 = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| \begin{matrix} f_1(x_i^f, y_i^f) - \left| f_1(x_i^f, y_i^f) \right| \\ f_2(x_i^f, y_i^f) - \left| f_2(x_i^f, y_i^f) \right| \end{matrix} \right|^2 \tag{11}$$

Therefore, the final loss function for the test example is given by

$$loss = loss_1 + loss_2 \, . \tag{12}$$

RESULTS

We implemented the PINN approach outlined in method section using Tensorflow1.0 [6] toolbox and applied it to the 2G test problem. We used the limited memory BFGS [7] method for optimizing the loss function. We sampled the solution domain using a set of 10^5 collocation points and a set of 200 boundary points per side (i.e., $N_f = 10^5$, $N_b = 200$). All training points were generated using the Latin Hypercube Sampling strategy (LHS).

We constructed a fully connected, deep neural network with 10 hidden layers, 60 neurons per hidden layer to approximate the solution. The training was carried on using Google Colab and the run time was about 30 minutes.

For solution verification, we used COMSOL Multiphysics [8] to obtain a high order Finite Element (FEM) solution. We normalized the flux solutions for the sake of pointwise comparison. The fast and thermal flux distributions yield from the PINN approach are shown in Fig. 2.

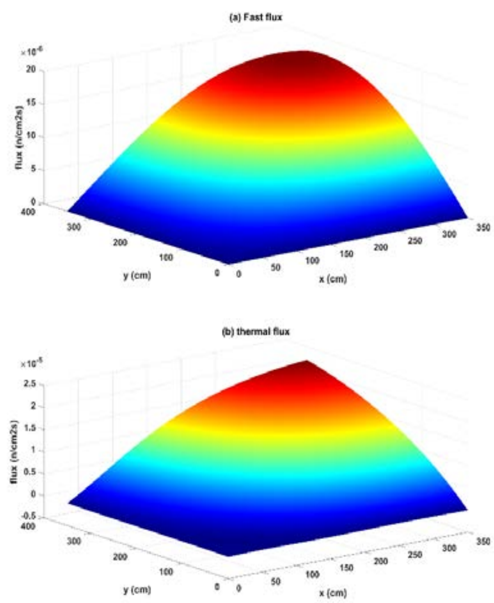

Fig. 2. PINN solution for fast flux distribution (a) and thermal flux distribution (b) of the test problem.

For more quantitate comparisons, we defined Peaking Factor (PF) metric for the fluxes:

$$\mathrm{PF} = \frac{\max\left(\phi_g(x,y)\right)}{\frac{1}{A}\iint \phi_g(x,y)\,dx\,dy} \tag{13}$$

The values of PF and the relative errors between the PINN solutions and FEM solutions are presented in Table II. As can be seen, a general acceptable agreement between the two results was observed. However, in contrast to nearly negligible discrepancy appeared in fast flux, the discrepancy in PF for thermal flux was rather large, which also contributed to the non-negligible PF errors in the total flux. The large discrepancy in the thermal flux PF is mainly due to the non-uniform nature for our equation. We essentially have a loss function that is a sum of two losses. The loss due to the thermal flux equation is at least one order of magnitude smaller than the fast flux loss. This may be a result of the fixed-source term in the fast flux equation. Applying a gradient-descent-based algorithm to such a loss function results in non-uniform optimization (i.e., the optimizer is biased towards the larger terms in the loss function due to their higher gradient). This effect leads to large residuals in the thermal flux equation, which in turn leads to large residuals in the fast flux due to the coupling of the equations. This complex topology of the loss function resulted insufficient training in all numerical experiments. The minimum value achieved of the loss function is in order of 10^{-5}. This is at least three orders of magnitude higher than the loss achieved in our previous one group test examples [3].

Table II. Values of PFs for the PINN and FEM Solutions.

	PINN PF	FEM PF	Error (%)
Fast flux	2.1932	2.1917	0.068%
Thermal flux	2.6927	2.1917	22.85%
Total flux	2.4078	2.1917	9.86%

The flux distributions and point-wise errors along the diagonal line ($y = x$) of the space domain for the fast and the thermal flux are shown in Fig. 3 and Fig. 4 respectively.

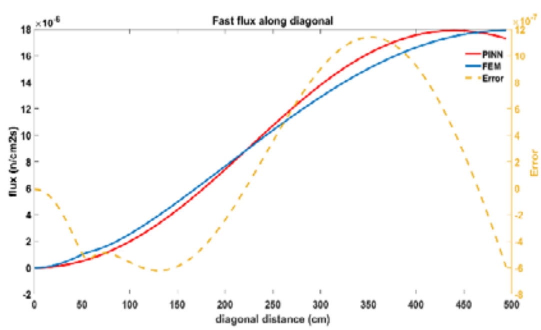

Fig. 3. A comparison between PINN solution and FEM solution for the fast flux along the diagonal (x=y).

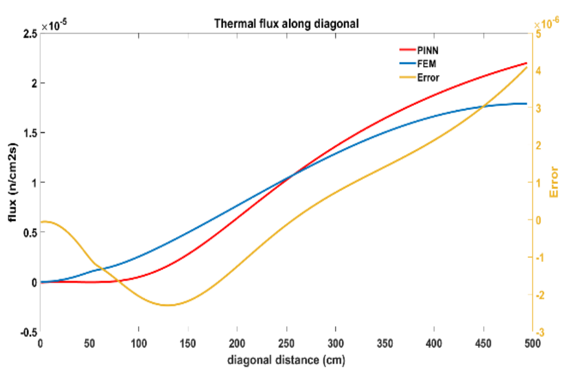

Fig. 4. A comparison between PINN solution and FEM solution for the thermal flux along the diagonal line.

CONCLUSIONS

This work extends the PINN approach to the multigroup diffusion models. Our results on the application of PINN model to a test example justified the feasibility of the PINN approach for the 2G diffusion model. However, more effort is needed for the optimization of this joint-learning task. The complexity of the loss function arising from the fact the PDE model is represented by a two coupled equations that have to be solved simultaneously. This complexity makes the learning task very sensitive to both the training hyperparameters and the optimization algorithm. A parametric study is needed to optimize these conditions to obtain more accurate solutions for the 2G diffusion models.

REFERENCES

[1]. M. RAISSI et al., "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, **378**, (2019).

[2]. A. GRIEWANK and A. WALTHER, *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*, 2nd Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA (2008).

[3]. M. H. ELHAREEF, Z. WU, and Y. MA, "Physics-Informed Deep Learning Neural Network Solution to the Neutron Diffusion Model," *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2021)*, Raleigh, North Carolina (2021).

[4]. M. H. ELHAREEF and Z. WU, "Extension of the PINN Diffusion Model to k-eigenvalue Problems", submitted to the PHYSOR 2022 Conference, May 15-20, Pittsburg, PA (2022).

[5]. J. P. SENECAL and W. JI, "Characterization of the proper generalized decomposition method for fixed-source diffusion problems", *Annals of Nuclear Energy*, **126**, (2019)

[6]. M. ABADI et al., "Tensorflow: large-scale machine learning on heterogeneous distributed systems," *arXiv:1603.04467* (2016).

[7]. D. LIU and J. NOCEDAL, "On the limited memory BFGS method for large scale optimization," *Math. Program*, **45**, (1989).

[8]. COMSOL Multiphysics Reference Manual, COMSOL Multiphysics® v. 5.0, 742-750 (2014).