Extension of the PINN Diffusion Model to *k*-eigenvalue Problems

Mohamed H Elhareef and Zeyun Wu

Department of Mechanical and Nuclear Engineering,
Virginia Commonwealth University
401 W. Main Street, Richmond VA, USA 23284-3015

elhareefmh@mymail.vcu.edu; zwu@vcu.edu;

doi.org/10.13182/PHYSOR22-37488

ABSTRACT

This paper extends our recent work on the Physics-Informed Neural Networks (PINN) approach for the fixed source diffusion models (Ref. [10]) and applies it to the diffusion theory based *k*-eigenvalue problems. To make the PINN equitable for the eigenvalue problems, we introduce a novel integral regularization term to the loss function in the framework, and allow the direct inference of the principal eigenvalue and the associated eigenfunction. The regularization term enforces a pre-defined value on the integration of the model predictions, and this value can be directly related to a physical property of the system. We also introduce an additional learnable parameter to approximate the principal eigenvalue. As a proof of principle, we solve the one-group two-dimensional *k*-eigenvalue neutron diffusion equation in this work. We then provide two numerical examples to demonstrate the applicability of the PINN approach. In each example, we solve the *k*-eigenvalue diffusion equation in a multi-region configuration constrained with a set of Robin boundary conditions for generality. We use a FEM solution based on the power-iteration method to verify the results of the PINN solution. The results showed relative percentage error in the predicted eigenvalue of $\sim 0.77\%$ and $\sim 1.2\%$ for example 1 and example 2, respectively. The mean absolute error in the predicted flux for example 1 is ~ 0.002 and for example 2 is ~ 0.0024. These results indicate some preliminary successes of the PINN application to *k*-eigenvalue problems.

KEYWORDS: *k*-eigenvalue problem, deep learning, physics-informed neural networks.

1. INTRODUCTION

Physics-Informed Neural Networks (PINN) is a framework that was first introduced by Raissi et al. [1] for solving general non-linear partial differential equations (PDEs). PINN can be applied for two classes of problems: data-driven solution (forward approach) and data-driven discovery (identification approach) of PDEs. In data-driven solution of PDEs, the solution of the PDE is approximated to a deep Neural Network (NN) model that is trained to regenerate the observed data while obeying the physical laws that govern the data. The PDE model can be constructed by applying automatic differentiation on the NN model. Data-driven solutions for PDEs can be obtained by using a set of labeled initial and/or boundary points and a set of collocation points to optimize the NN learnable parameters in order to minimize a loss function that penalizes the model for both: mismatch between the model predictions and the labeled data, and deviation from the PDE model. In data-driven discovery of PDEs, a set of observed data points are available and a parameter that best fit the data to a parametrized PDE model is to be obtained. This problem can be solved

by introducing a learnable parameter that represent the unknown parameter. The additional learnable parameter along with the NN learnable parameters can be learnt in the same way as in the forward approach.

Following with Raissi et al. [1], Yang and Perdikaris [2] developed a method to quantify uncertainty in the PINN predictions by introducing a class of probabilistic PINN. Probabilistic PINN can be trained to approximate an arbitrary conditional probability density function of the observable output depending on its free variables and a collection of random latent variables. This approach enables using noisy and incomplete data along with the physical laws to predict the system states. Jagtap et al. [3] introduced a class of adaptive activation functions (transfer functions) with a scalable hyper-parameter to accelerate the convergence of deep NNs and PINNs. Their results showed that adaptive activation functions are efficient in increasing convergence of NNs and also enhancing the performance due to increased learning capabilities. As of the time of the present work, PINN has been applied to various domains including: fluids [1, 4-7], quantum mechanics [1], Cardiac Activation Mapping [8], diffusion systems [9, 10], nano-optics and metamaterials [11], Power Systems [12], heat transfer [13, 14], etc. The advantages of PINN include providing a straightforward approach for obtaining mesh-free solutions of non-linear systems of PDEs. Moreover, PINN provide a framework for extracting accurate solutions from sparse and noisy observational data allowing the reduction of the need to repeat expensive experiments or numerical simulations.

In our previous work [10], the PINN methodology has been customized and employed to solve two-dimensional (2D) fixed source neutron diffusion models as a demonstration to show the applicability of PINN for reactor physics problems. Preliminary results showed that nearly the same level of accuracy of the flux solution can be achieved by the PINN method compared to conventional numerical method such as finite element method (FEM). The computational cost of the PINN approach at current state is a little higher. However, by taking advantage of the state-of-the-art open source NN toolbox, the manpower efforts needed for developing the PINN can be significantly reduced compared to that required for the conventional method development.

In the present work, we are extending the PINN solution to *k*-eigenvalue problems, still based on diffusion models. To our knowledge, this work represents the first attempt to implement PINN approach for eigenvalue type problems, which offers some valuable insights in the method implementation. To proceed, we introduce a novel regularization term in the loss function to allow for learning both the principal eigenvalue and the associated eigenfunction. In Section 2, we discuss the methods used for solving the 2D *k*-eigenvalue neutron diffusion equation. In Section 3, we tested the proposed PINN implementation with two numerical examples. In Section 4, we offered some concluding remarks on these research efforts.

2. METHODOLOGY

The one-group (1G) two-dimensional (2D) *k*-eigenvalue neutron diffusion equation may be given as

$$-\left[\frac{\partial}{\partial x}\left(D\frac{\partial \phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(D\frac{\partial \phi}{\partial y}\right)\right]+\Sigma_a(x,y)\phi(x,y)=\frac{1}{k}\nu\Sigma_f(x,y)\phi(x,y) \tag{1}$$

where Σ_a is the macroscopic absorption cross section, Σ_f is the macroscopic fission cross section, D is the diffusion coefficient defined by

$$D=\frac{1}{3\Sigma_t}=\frac{1}{3(\Sigma_a+\Sigma_s)} \tag{2}$$

for the isotropic scattering case, and Σ_t, Σ_s are the macroscopic total and isotropic scattering cross section, respectively.

The most well-known numerical scheme to solve for the eigenvalue [i.e., the k in Eq. (1)] is the power iteration method. In this scheme, a fixed-source type problem is essentially solved iteratively with the source term updated in each iteration. The method starts from an initial guess of the flux distribution and k value to calculate the R.H.S. of Eq. (1). Then, a numerical approach (e.g., FEM) is used to solve the fixed-source problem for the new flux distribution. The new value of k and new source term is then updated from its physical definition. This process then iterates until convergence conditions are satisfied. A flow diagram of the power iteration scheme applied to Eq. (1) is shown in Figure 1.

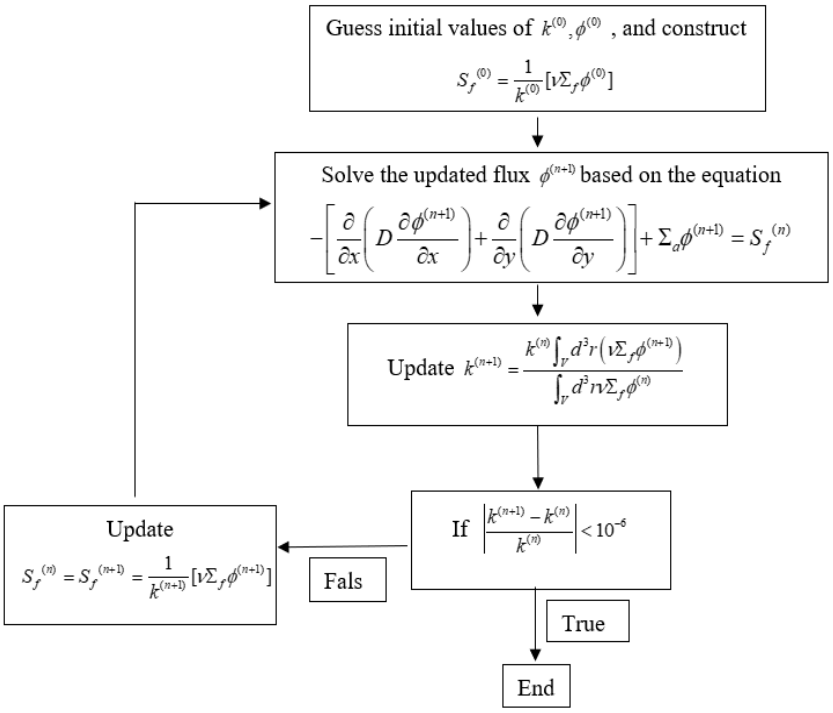

Figure 1. The scheme of power iteration method for solving the k-eigenvalue diffusion problem.

To implement the PINN method to *k*-eigenvalue diffusion problems, we represent the flux solution by a NN model and introduce an extra learnable parameter for the unknown eigenvalue *k*. The method starts by defining a residual form of quantity *f* as follows:

$$f := \frac{1}{k}\nu\Sigma_f(x,y)\phi(x,y) + \left[\frac{\partial}{\partial x}\left(D\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(D\frac{\partial\phi}{\partial y}\right)\right] - \Sigma_a(x,y)\phi(x,y) = 0 \qquad (3)$$

The direct implementation of PINN approach implies the construction of a NN ($\phi_{net}(x,y)$) to approximate the solution. The training loss function has two terms: the first term is evaluated from the PDE model [i.e., Eq. (3)] at a set of collocation points, the second term is evaluated at a set of boundary points. The loss function penalizes the NN model on the deviation of predictions from the target values set by both PDE and its boundary conditions. The value of loss function and its derivatives with respect to the model learnable parameters is used to iteratively update the learnable parameters until a convergence condition is reached. Applying this approach to the eigenvalue diffusion problem will always converge to the trivial

solution ($\phi(x,y) = 0$) and any arbitrary value of the eigenvalue (k). This result arises from the fact that Eq. (1) is homogenous and the attempt to directly minimize the quantity f [i.e., Eq. (3)] will drive the model predictions towards $\phi_{net}(x,y) = 0$. To avoid the trivial solution, we add a regularization term to the loss function. The regularization condition can be defined in terms of some physical quantity such as the reactor power.

The aim of the learning task is to obtain the fundamental solution $(\phi_{net}(x,y)|k)$. The solution is defined in terms of the set learnable parameters (k, w, b), where *w* and *b* refer to the weights and biases of the NN model. The learnable parameters can be learnt by minimizing a loss function that consists of the following three terms:

(1). Internal points loss term

A NN model $f_{net}(x,y)$ can be constructed by using Automatic differentiation on $\phi_{net}(x,y)$ according to Eq. (3). The predictions of $f_{net}(x,y)$ is evaluated at a set of collocation (N_{if}) points that are obtained by sampling the solution domain. The error in $f_{net}(x,y)$ predictions is defined as the MSE as follows:

$$Loss_f = \frac{1}{N_f}\sum_{i=1}^{N_f}[f_{net}(x_i, y_i)]^2 \tag{4}$$

(2). Boundary points loss term

We solve Eq. (1) with zero-incoming fluxes are assumed for all boundary surfaces of the problem (square geometry, side length = 100 cm), which can be expressed as Robin type boundary conditions as follows:

$$\text{At the left surface, } x = 0: \quad \frac{1}{4}\phi(0,y) - \frac{1}{2}D\frac{d\phi}{dx}\Big|_{x=0} = 0 \tag{5}$$

$$\text{At the right surface, } x = 100: \quad \frac{1}{4}\phi(100,y) + \frac{1}{2}D\frac{d\phi}{dx}\Big|_{x=100} = 0 \tag{6}$$

$$\text{At the bottom surface, y} = 0: \quad \frac{1}{4}\phi(x,0) - \frac{1}{2}D\frac{d\phi}{dy}\Big|_{y=0} = 0 \tag{7}$$

$$\text{At the top surface, y} = 100: \quad \frac{1}{4}\phi(x,100) + \frac{1}{2}D\frac{d\phi}{dy}\Big|_{y=100} = 0 \tag{8}$$

For each boundary surface, a NN model is constructed by differentiating $\phi_{net}(x,y)$ according to the corresponding ODE. For instance, for the left boundary we define b_L as follows:

$$b_L := \frac{1}{4}\phi_{net}(0,y) - \frac{1}{2}D\frac{d\phi_{net}(0,y)}{dx} = 0 \tag{9}$$

Eq. (9) is evaluated at a set of points sampled from the left surface to calculate its contribution to the model error. The total boundary loss is defined as follows:

$$loss_b = \frac{1}{N_b}\left(\sum_j^{N_b}\left[b_L\left(0, y_L^j\right)\right]^2 + \left[b_R\left(100, y_R^j\right)\right]^2 + \left[b_B\left(x_B^j, 0\right)\right]^2 + \left[b_T\left(x_T^j, 100\right)\right]^2\right) \tag{10}$$

where the subscripts *L*, *R*, *B*, and *T* refer to the surfaces: Left, Right, Bottom, and Top respectively.

(3). Regularization term

The regularization term is used to enforce a non-zero solution for the homogenous equation. This can be done by restricting the integration of the converged flux to a fixed non-zero value. This condition can be written as:

$$\iint \phi(x, y)dxdy = C \tag{11}$$

where *C* is a parameter that can be defined in terms of other physical quantities, or simply can be set to equal 1 to obtain a normalized solution. The integral in Eq.(11) can be estimated from the NN predictions at the set collocation points and the regularization term is defined as:

$$R = \left[\frac{1}{N_f}\left(\sum_{i=1}^{N_f}\phi_{net}(x_i, y_i)\right) - \frac{C}{N_f}\right]^2 \tag{12}$$

Thus, the global training loss function to be optimized is:

$$Loss = Loss_f + Loss_b + R \tag{13}$$

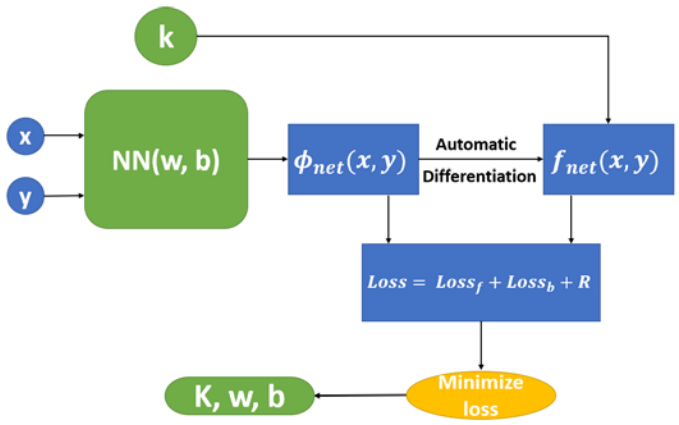

Figure 2. Schematic of PINN for learning the principal solution of the *k*-eigenvalue problems.

A schematic flow chart of the learning procedure of the PINN approach for solving the *k*-eigenvalue diffusion problem is shown in Figure 2. The training process starts by passing the training data sets through the NN model to evaluate the model predictions and automatic differentiation is used to evaluate the gradients of the predictions according to the PDE model and the boundary conditions. The learnable parameter k is then passed to the function $f_{net}(x, y)$to evaluate the predicted values of f. The model predictions are used to evaluate the loss function and its gradients with respect to the learnable parameters (k, w, b). An optimization algorithm (gradient decent-based) is then used to minimize the loss with respect to the learnable parameters. Finally, the optimum learnable parameters are used to construct the PINN solution as the predicted eigenvalue is represented by the learnable parameter k, and the predicted eigenfunction is represented by the model $NN(w, b)$.

3. NUMERICAL EXAMPLES

We demonstrated our approach by solving the *k*-eigenvalue diffusion problem for two different configurations shown in Figure 3. The material properties of each region, as defined in Figure 3, are listed in Table I.

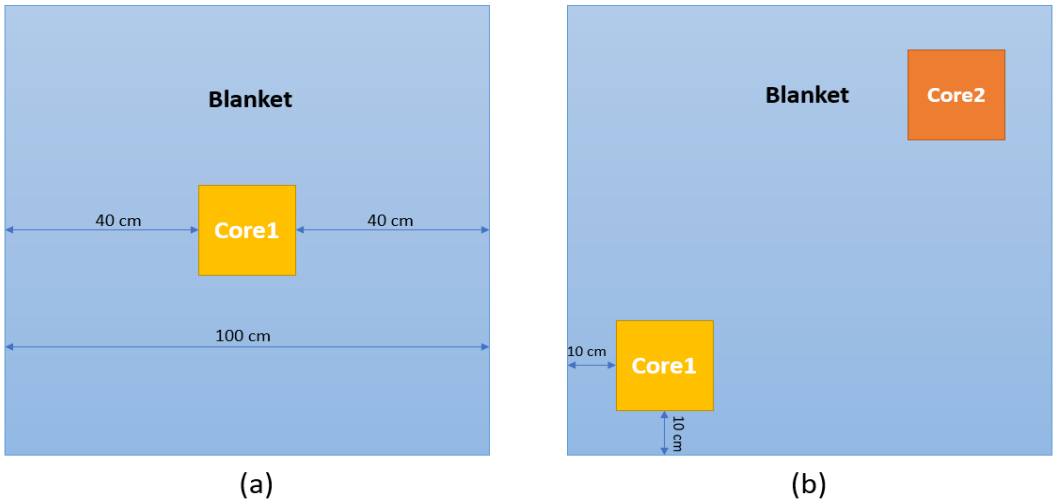

Figure 3. The geometric configuration of the two numerical examples: (a) is the configuration of Example 1, and (b) is the configuration of Example 2.

Table I. Material for the two numerical examples

Region Material	Σ_a (cm^{-1})	D (cm)	$\nu\Sigma_f$ (cm^{-1})
Core1	0.062158	2.2008	0.107622
Core2	0.062158	2.2008	0.102622
Blanket	0.064256	2.095	0.0

Training parameters and NN architecture are the same that were used in our previous work [10] for solving the fixed-source diffusion model. For each of the two numerical example we used a NN model with 8 hidden layers, 40 neurons per layer, and the hyperbolic tangent sigmoid transfer function is used as the activation function. The learnable-parameters optimization was conducted by using Adam optimizer [15], which is a stochastic gradient decent optimization algorithm, for fixed number of iterations (10^5 iterations) before training on L-BFGS algorithm [16] until convergence. All models were implemented using

Tensoflow1.0 [17] python library. We used a training set with $N_f = 10^4$ and $N_b = 10^2$. All points were generated based on the Latin-hyperbolic sampling (LHS) strategy [18]. The value of the regularization parameter was taken to be $C = 10^3$. To verify the results of PINN approach, we obtained the reference solution of the examples by developing an algorithm based on the power iteration framework using the FEM solver integrated in COMSOL5.0 [19] as the flux solver.

3.1. Example 1

In the first numerical example, we solve the k-eigenvalue diffusion model for the configuration given in Figure 3 (a). The predicted value of $k = 0.9538$ and the power iteration algorithm converged at $k = 0.9612$. The relative percentage error in the predicted value is $\approx 0.77\%$. The predicted flux along with the pointwise different between the FEM solution and predicted flux is shown in Figure 4. The mean absolute error in the predicted flux is ≈ 0.002.

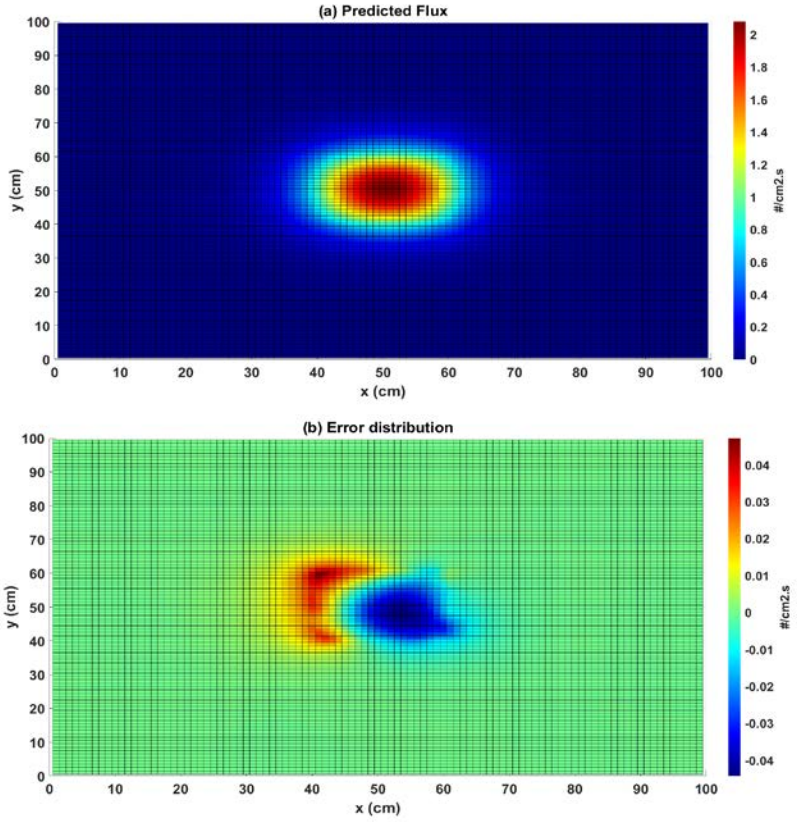

Figure 4. Flux solution for Example 1: (a) heatmap of the predicted flux, and (b) relative difference between the reference solution and PINN solution.

3.2. Example 2

In the first numerical example, we solve the PDE for the configuration given in Figure 3 (b). The predicted value of $k = 0.94861$and the power iteration algorithm converged at $k = 0.9602$. The relative percentage error in the predicted value is $\approx 1.2\%$. The predicted flux along with the pointwise different between the FEM solution and predicted flux is shown in Figure 5. The mean absolute error in the predicted flux is ≈ 0.0024.

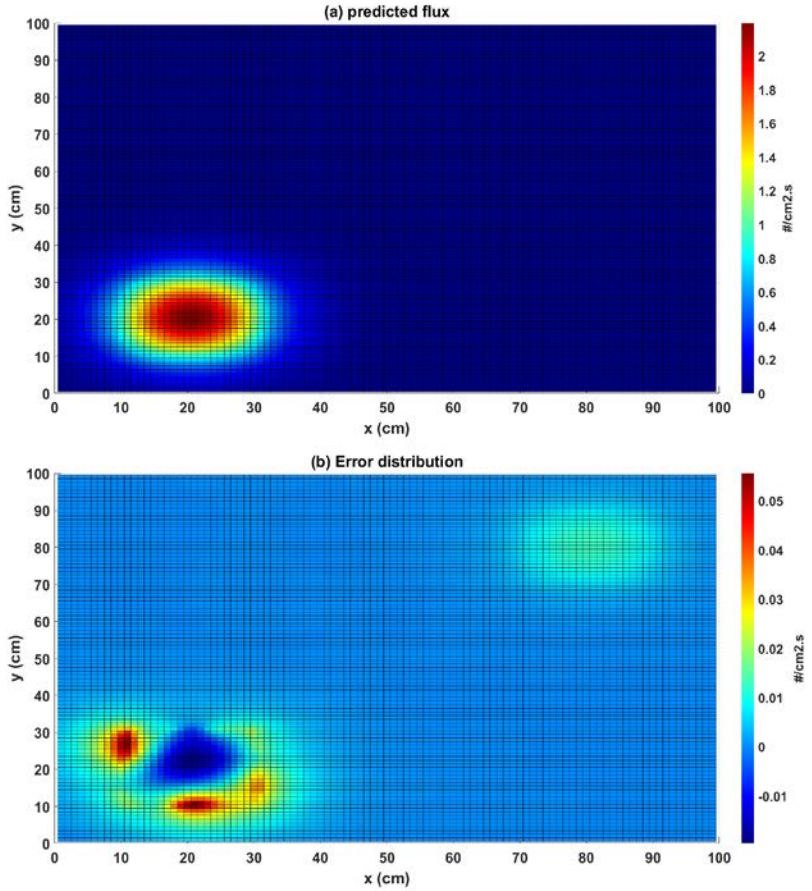

Figure 5. Flux solution for Example 2: (a) heatmap of the predicted flux, and (b) relative difference between the reference solution and PINN solution.

4. CONCLUSIONS

In this work, the forward PINN was extended to *k*-eigenvalue problems based on the diffusion model. In particular, the one-group (1G) two-dimensional (2D) *k*-eigenvalue neutron diffusion equation was solved. To accommodate the PINN for the eigenvalue problem, we modified the loss function in the forward PINN frame by introducing a novel integral regularization term. The addition of the regularization term eliminates the trivial solution, which would always result from the straightforward implementation of the PINN approach for homogenous eigenvalue problems. The value of the regularization constant in the regulation term can be related to certain physical property of the system. In addition to the common NN learnable parameters (w, b) that are used to approximate the solution of the PDE model in the forward PINN frame, we also introduced a learnable parameter to approximate the principal eigenvalue. All learnable parameters can be trained by minimizing the loss function which penalizes the model for: mismatch between predictions and boundary conditions, the value of the residual (f), and the deviation of the estimated integration value from the predefined regularization constant.

We demonstrated the proposed approach by applying it in numerical examples. We used the same hyperparameters that were optimized in our previous work [10] for the loosely coupled reactor model (LCRM) problem. Two variants of the LCRM were established as the numerical examples. For each

example, we used a NN model with 8 hidden layers, 40 neuron per hidden layer, and the hyperbolic tangent sigmoid activation function. The solution domain was sampled using 10^4 collocation points and 10^2 boundary points per side. All points were generated using the LHS technique. The training was performed using Adam optimizer for fixed number of iterations (10^5) followed by training on L-BFGS algorithm until convergence criteria are met. The obtained solutions were verified against a FEM solution based on the power iteration method. For numerical example 1, the relative percentage error in the predicted eigenvalue is $\sim 0.77\%$ and the mean absolute error in the predicted flux is ~ 0.002. The errors are slightly higher for example 2 as the relative percentage error in the predicted eigenvalue is $\sim 1.2\%$ and the mean absolute error in the predicted flux is ~ 0.0024. We used the mean error as a performance metric for the predicted flux instead of the relative error to distinguish the error distribution inside the solution domain because the relative error is much higher (two orders of magnitude) at boundaries compared to the internal points.

PINN solutions encompass two types of errors. The first type is the approximation error which results from using specific NN configuration to approximate the unknown target function. Using appropriate depth and width for the NN model and a suitable activation function should reduce the approximation error. The second type is the training error which results from both the training data and the optimization algorithm. The second type of error has a stochastic nature and depends on the number of points sampled from the solution domain and the sampling strategy. However, with a sufficient number of training points there should be a small dependence on the training data set. The major contribution in the PINN error is due to the optimization approach. The accuracy of PINN-based solutions mainly depends on the minimum achieved value of the loss function. While there is no guarantee for the convergence to a global minimum of the loss function and due to the nature of stochastic gradient descent-based optimizers, the predicted solution will have some level of error randomly scattered across the solution domain. The magnitude of this error depends on the minimum value of the loss function achieved which in turn depends on the optimization algorithm hyperparameters and the number of iterations.

Our work demonstrated the capability of the PINN approach for the direct inference of the principal eigenvalue and the corresponding eigenfunction with the same level of accuracy compared with conventional numerical approaches. The computational cost of the PINN approach for eigenvalue problems is essentially the same as the cost of the PINN approach for fixed problems. This results from the fact that the solution is learnt directly form the PDE model without the need to reduce the problem to a fixed problem that need to be solved iteratively, such as the power-iteration framework. This makes the computational cost of PINN approach comparable to conventional numerical approaches for eigenvalue problems. Moreover, this approach potentially has the advantage of being insensitive to the dominance ratio (DR) of the reactor because the conventional power iteration procedure is precluded in the PINN approach. Other advantages of PINN approach include: reduction in manpower efforts, and the applicability of complex geometries and versatile boundary conditions with very little sacrifice to the accuracy. Future considerations of this work include extensions of the current efforts to multigroup diffusion equations as well as neutron transport models for reactor problems.

REFERENCES

1. 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*, **378**, pp. 686-707 (2019).
2. Y. Yang and P. Perdikaris, “Adversarial uncertainty quantification in physics-informed neural networks,” *Journal of Computational Physics*, **394**, pp. 136-152 (2019).
3. A.D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, “Adaptive activation functions accelerate convergence in deep and physics-informed neural networks,” *Journal of Computational Physics*, **404**, p. 109136 (2020).

4. Z. Mao, A. D. Jagtap, and G. E. Karniadakis, “Physics-informed neural networks for high-speed flows,” *Computer Methods in Applied Mechanics and Engineering*, **360**, p. 112789 (2020).
5. J. Berg and K. Nyström, “A unified deep artificial neural network approach to partial differential equations in complex geometries,” *Neurocomputing*, **317**, pp. 28-41 (2018).
6. X. Jin et al., “NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations,” *Journal of Computational Physics*, **426**, p. 109951 (2021).
7. H. Wessels, C. Weißenfels, and P. Wriggers, “The neural particle method – An updated Lagrangian physics informed neural network for computational fluid dynamics,” *Computer Methods in Applied Mechanics and Engineering*, **368**, p. 113127 (2020).
8. F. Sahli Costabal et al., ‘Physics-Informed Neural Networks for Cardiac Activation Mapping,” *Frontiers in Physics*, (2020).
9. D. Zhang et al., “Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems,” *Journal of Computational Physics*, **397**, p. 108850 (2019).
10. 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).
11. Y. Chen et al., “Physics-informed neural networks for inverse problems in nano-optics and metamaterials,” *Optics Express*, **28**(8), pp. 11618-11633 (2020).
12. G. S. Misyris, A. Venzke, and S. Chatzivasileiadis, “Physics-Informed Neural Networks for Power Systems,” *Proceedings of 2020 IEEE Power & Energy Society General Meeting (PESGM)*, (2020).
13. S. Mishra and R. Molinaro, ‘Physics informed neural networks for simulating radiative transfer,” *Journal of Quantitative Spectroscopy and Radiative Transfer*, **270**, p. 107705 (2021).
14. S. Cai et al., “Physics-Informed Neural Networks for Heat Transfer Problems,” *Journal of Heat Transfer*, **143**(6) (2021).
15. D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint, arXiv:1412.6980 (2014).
16. D. C. Liu and J. Nocedal, “On the limited memory BFGS method for large scale optimization,” *Mathematical Programming*, **45**(1), pp. 503-528 (1989).
17. M. Abadi et al., “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” arXiv preprint, arXiv:1603.04467 (2016).
18. M. D. Mckay, R. J. Beckman, and W. J. Conover, “A comparison of three methods for selecting values of input variables in the analysis of output from a computer code,” *Technometrics* **42**(1), pp. 55-61 (1979).
19. COMSOL Muliphysics Reference Manual, COMSOL Multiphysics® (2014).