Physics-Constrained neural network for solving discontinuous interface K-eigenvalue problem with application to reactor physics

Authors: Qihong Yang, Yu Yang, Yangtao Deng, Qiaolin He, Helin Gong, Shiquan Zhang, Qiaolin He, Helin Gong

Date: 2023-09-29T00:00:00+00:00

Abstract

Machine learning-based modeling of reactor physics problems has attracted increasing interest in recent years. Despite some progress in one-dimensional problems, there is still a paucity of benchmark studies that are easy to solve using traditional numerical methods albeit still challenging using neural networks for a wide range of practical problems. We present two networks, namely the Generalized Inverse Power Method Neural Network (GIPMNN) and Physics-Constrained GIPMNN (PC-GIPMNN) to solve K-eigenvalue problems in neutron diffusion theory. GIPMNN follows the main idea of the inverse power method and determines the lowest eigenvalue using an iterative method. The PC-GIPMNN additionally enforces conservative interface conditions for the neutron flux. Meanwhile, Deep Ritz Method (DRM) directly solves the smallest eigenvalue by minimizing the eigenvalue in Rayleigh quotient form. A comprehensive study was conducted using GIPMNN, PC-GIPMNN, and DRM to solve problems of complex spatial geometry with variant material domains from the field of nuclear reactor physics. The methods were compared with the standard finite element method. The applicability and accuracy of the methods are reported and indicate that PC-GIPMNN outperforms GIPMNN and DRM.

Full Text

Physics-Constrained Neural Network for Solving Discontinuous Interface K-Eigenvalue Problems with Application to Reactor Physics

Qi-Hong Yang,[1] Yu Yang,[1] Yang-Tao Deng,[1] Qiao-Lin He,[1],† He-Lin Gong,[2],‡ and Shi-Quan Zhang[1]
[1]School of Mathematics, Sichuan University, Chengdu, China

[2]Paris Elite Institute of Technology, Shanghai Jiao Tong University, Shanghai, China

Machine learning-based modeling of reactor physics problems has attracted increasing interest in recent years. Despite some progress in one-dimensional problems, there remains a paucity of benchmark studies that are easy to solve using traditional numerical methods yet still challenging for neural networks across a wide range of practical problems. We present two networks, namely the Generalized Inverse Power Method Neural Network (GIPMNN) and Physics-Constrained GIPMNN (PC-GIPMNN), to solve K-eigenvalue problems in neutron diffusion theory. GIPMNN follows the main idea of the inverse power method and determines the lowest eigenvalue using an iterative approach. The PC-GIPMNN additionally enforces conservative interface conditions for the neutron flux. Meanwhile, the Deep Ritz Method (DRM) directly solves for the smallest eigenvalue by minimizing the eigenvalue in Rayleigh quotient form. A comprehensive study was conducted using GIPMNN, PC-GIPMNN, and DRM to solve problems of complex spatial geometry with varying material domains from the field of nuclear reactor physics. The methods were compared with the standard finite element method. The applicability and accuracy of the methods are reported and indicate that PC-GIPMNN outperforms both GIPMNN and DRM.

Keywords: Neural network, Reactor physics, Neutron diffusion equation, Eigenvalue problem, Inverse power method

Introduction

In the nuclear engineering domain, the fundamental mode solution of the K-eigenvalue problem based on steady-state multigroup neutron diffusion theory is crucial for simulation and analysis of nuclear reactors. The eigenvalue equation can be expressed as follows:

$$-\nabla\cdot(D_g\nabla\phi_g)+\Sigma_R^g\phi_g-\sum_{g'\neq g}\Sigma_{g'\to g}\phi_{g'}=\frac{1}{k_{\text{eff}}}\chi_g\sum_{g'=1}^{G}\nu\Sigma_f^{g'}\phi_{g'},\quad g=1,2,...,G,$$

where $\phi_g\equiv\phi_g(\mathbf{r})$ denotes the neutron flux at spatial point $\mathbf{r}$ in the g-th energy group, and D_g, Σ_R^g, $\Sigma_{g'\to g}$, χ_g, and $\nu\Sigma_f^{g'}$ denote spatially dependent (possibly discontinuous) parameters that reflect the material properties in a reactor core [1]. Nuclear engineers and analysts must numerically determine the fundamental mode eigenvalue (commonly called k_{eff}) and the corresponding eigenvector for a given geometry/material configuration. For numerical solution, Eq. (1) must be discretized and reduced to a set of G-coupled algebraic equations, which can be expressed using matrices as follows:

$$M\Phi = \frac{1}{k_{\text{eff}}} F\Phi.$$

This is often termed a generalized eigenvalue problem because coefficient matrices appear on both sides of the equation. Many mature numerical methods, such as the finite difference method [2], nodal collocation method [3], finite element method [4–6], and nodal expansion method [7–9], have been proposed to solve neutron diffusion equations. Among these methods, the nodal expansion method is widely used because it is easier to implement and requires less computational effort than other methods. The power iteration method is the most well-known numerical approach for solving the principal K-eigenvalue [10]. A detailed review of conventional numerical methods for solving K-eigenvalue problems is available in recent nuclear reactor physics textbooks [1, 11, 12].

Although traditional and mature numerical methods are currently widely used in nuclear reactor physics to solve K-eigenvalue problems with acceptable engineering accuracy and computational cost, state-of-the-art neural network approaches for solving K-eigenvalue problems remain in their infancy. However, neural networks exhibit the potential to provide an alternative option for solving nuclear engineering problems, particularly with advances in new algorithms and hardware.

As mentioned above, many traditional numerical methods have been proposed to solve neutron diffusion equations. However, a dense mesh is required to ensure highly accurate results. The model consumes more computational resources if the mesh is extremely dense, while inaccurate solutions are obtained if the mesh is coarse. Moreover, for high-dimensional problems, classical methods are either less efficient or unsuccessful due to the curse of dimensionality. A neural network can potentially enhance performance by using a capable hypothesis space due to its relatively low statistical error [13].

Neural networks exhibit multiple potential benefits when compared with conventional numerical methods for solving nuclear engineering problems. First, they provide mesh-free solutions to approximate physics fields in nuclear reactors, where mesh generation is significantly complex due to high heterogeneity of geometry and material. Second, they provide a general framework for solving high-dimensional problems governed by parameterized PDEs, particularly for the original neutron transport equation, which comprises seven variables: three spatial variables, two directional variables, one energy variable, and one temporal variable. Third, they are able to seamlessly incorporate prior data (potentially with noise) into existing algorithms, whereas data assimilation is necessary as a post-process for conventional numerical methods [14–19]. Finally, they provide a general framework for solving inverse problems with hidden physics, which are typically prohibitively expensive and require different formulations and elaborate computer codes for conventional numerical methods.

In recent years, neural networks have been widely used to solve partial differ-

ential equations (PDEs) [20] and have achieved remarkable success. Based on neural networks, numerous methods have been proposed to solve PDEs, such as the deep backward stochastic differential equation (BSDE) method [21, 22], deep Galerkin method (DGM) [23], deep Ritz method (DRM) [24], and Physics-Informed Neural Network (PINN) [25]. The deep BSDE method reformulates PDEs using backward stochastic differential equations and approximates the gradient of an unknown solution using neural networks. Although DGM and PINN appear independently under two names in the literature, they are similar in that both train neural networks by minimizing the mean squared error loss of the residual equation. DRM, however, reformulates the original problem into a variational problem and trains neural networks by minimizing the energy function of the variational problem. Specifically, DRM and PINN [25] have attracted widespread attention, with extensive studies focusing on these methods to solve a variety of problems, including fiber optics [26], hyperelasticity [27], solid mechanics [28], heat transfer problems [29–31], inverse problems [32–35], and uncertainty quantification [36–39]. However, only a few studies have focused on solving eigenvalue problems [24, 40–45].

The need to solve eigenvalue problems using neural networks can be traced back to 2018 [24], when a deep Ritz method for solving variational problems was proposed, and several examples illustrated how to use DRM to solve eigenvalue problems. The original eigenvalue problem was transformed into a variational problem, and a specially defined loss function was constructed using the variational principle [46], termed the Rayleigh quotient. The Rayleigh quotient is a well-known approximation of the eigenvalue of matrix A, defined by $R = \frac{x^T Ax}{x^T x}$. By minimizing this loss function, the smallest eigenvalue could be obtained. Similarly, some studies [41, 42] directly used PINN to solve eigenvalue problems. In contrast to DRM, which transfers the original problem to a variational formulation, PINN solves eigenvalue problems without variation. Neural networks are used in PINN to represent the function, and automatic differentiation (AD) [47] is used to compute the derivatives required by the differential operators. The loss function is obtained using the Rayleigh quotient, and the smallest eigenvalue and corresponding eigenvector are then solved using optimization tools. Moreover, the deep forward-backward stochastic differential equation (FBSDE) method [48] was proposed to solve eigenvalue problems, representing an expansion of the deep BSDE method. In this approach, the eigenvalue problem is reformulated as a fixed-point problem of semigroup flow induced by the operator.

Other studies [43, 44] proposed an alternative method for learning eigenvalue problems by adding one or two regularization terms to the loss function. More recently, a neural network framework based on the power method [50] was presented to solve eigenvalue problems and smallest eigenvalue problems, where the eigenfunction is expressed by the neural network and iteratively solved following the idea of the power method or inverse power method [49]. However, the scope was limited to linear operators and certain special eigenvalue problems.

In a recent study, PINN was applied to solve neutron diffusion equations [45, 51], where the authors used a free learnable parameter to approximate the eigenvalue and a novel regularization technique to exclude null solutions from the PINN framework. A conservative physics-informed neural network (cPINN) was proposed for discrete domains for nonlinear conservation laws [52]. Moreover, cPINN [53] was applied to solve heterogeneous neutron diffusion problems in one-dimensional cases, developing PINN for each subdomain and considering additional conservation laws along the interfaces of subdomains (a general consideration in reactor physics [11]), with the eigenvalue involved in neural network training as a variable to be optimized.

More recently, a data-enabled physics-informed neural network (DEPINN) [40] was proposed to solve neutron diffusion eigenvalue problems. To achieve acceptable engineering accuracy for complex engineering problems, it was suggested that a very small amount of prior data from physical experiments be used to improve training accuracy and efficiency. In contrast to PINN, which solves the neutron diffusion eigenvalue problem directly, an autoencoder-based machine learning method in combination with the reduced-order method [54, 55] was proposed [56] to solve the same problem. However, this approach still relies on solving governing equations with traditional numerical methods such as the finite difference method.

Although DRM provides a way to solve eigenvalue problems with neural networks, as shown in Section IV B 2, results indicate that DRM is not stable when solving two-dimensional cases. First, DRM learns the eigenvalue and eigenfunction at the early stage of the training process. Subsequently, DRM attempts to learn a smaller eigenvalue after it approaches the true eigenvalue. Finally, DRM successfully learns a smaller eigenvalue that may be close to the true eigenvalue but an incorrect eigenfunction that is meaningless and far from the true eigenfunction. Additionally, the framework of a neural network based on the power method [50] is unsuitable for generalized eigenvalue problems. Therefore, it is necessary to propose a new algorithm to solve K-eigenvalue problems.

This study focuses on eigenvalue problems that are also interface problems, in which the eigenfunctions are continuous at the interface while the derivatives of the eigenfunction are not continuous at the interface. Specifically, in the nuclear reactor physics domain, this is a general problem in which the reactor core is composed of fuel assemblies with different fissile nuclide enrichments [1]. Some studies have focused on using neural networks to solve elliptic interface problems. Some researchers [57] used the idea of DRM and formulated the PDEs into variational problems that can be solved using deep learning approaches, presenting a novel mesh-free numerical method for solving elliptical interface problems based on deep learning [58]. They employed different neural networks in different subdomains and reformulated the problem as a least-squares problem. A similar case exists where the authors [53] enforced interface conditions using piecewise neural networks. In contrast to these methods, a discontinuity-capturing shallow neural network (DCSNN) [59] has been proposed for elliptic

interface problems. The crucial concept of DCSNN is that a d-dimensional piecewise continuous function can be extended to a continuous function defined in $(d+1)$-dimensional space, where the augmented coordinate variable labels the pieces of each subdomain. However, to the best of the authors' knowledge, only a few studies have focused on using neural networks to solve eigenvalue problems that incorporate interface problems involving regions of different materials. Challenges exist on at least three fronts: (1) designing a neural network that is more suitable for K-eigenvalue problems for more complicated medium-size test problems; (2) dealing with the interface problem in a more general and understandable manner when designing the neural network; and (3) proposing a framework that effectively enhances the robustness of the neural network and improves the efficiency of utilizing noisy prior data.

To address these challenges and advance beyond the state-of-the-art research [45, 51, 53], we initially introduced the study [40], which served as a preliminary demonstration of the applicability of the PINN approach to reactor physics eigenvalue problems in complex engineering scenarios. The contributions of this study are as follows. First, we extend the Inverse Power Method Neural Network (IPMNN) [49] to the so-called Generalized Inverse Power Method Neural Network (GIPMNN) to solve the smallest eigenvalue and the related eigenfunction of generalized K-eigenvalue problems. Compared to DEPINN from our previous study [53], we omit the prior data in the training process and attempt to solve the K-eigenvalue problems from a data-free mathematical/numerical perspective. Second, we propose a Physics-Constrained GIPMNN (PC-GIPMNN) to address the interface problem in a more general and understandable manner than previous studies [45, 51, 53]. Finally, we conduct a thorough comparative study of GIPMNN, PC-GIPMNN, and DRM using a variety of numerical tests. We evaluate the applicability and accuracy of these three methods using typical 1D and 2D test cases in reactor physics, particularly accounting for material discontinuities in different geometries. In the 1D example, we determine the optimal ratio of outer to inner iterations, a finding that may be particularly relevant for GIPMNN. Additionally, we observe the failure of DRM in the 2D experiments, whereas PC-GIPMNN consistently outperforms GIPMNN and DRM over a fixed number of epochs.

The rest of this paper is organized as follows. The governing equations for the eigenvalue problems are presented in Section II. In Section III, we propose GIPMNN and PC-GIPMNN and introduce DRM for our cases. In Section IV, the results of 1D and 2D test cases are presented to verify the three methods. Finally, conclusions and future research directions are discussed in Section V.

II. K-Eigenvalue Problems

This section introduces the equations that govern neutron criticality over a spatial domain. We recall Eqs. (1) and (2). The generalized K-eigenvalue problem can be formulated as follows:

$$\begin{cases} L\phi = \lambda Q\phi, & \text{in } \Omega, \\ B\phi = g, & \text{on } \partial\Omega, \end{cases}$$

where domain $\Omega \subset \mathbb{R}^d$, and L, Q, and B denote differential operators acting on functions defined in the interior of Ω and at the boundary of Ω ($\partial\Omega$). Furthermore, ϕ denotes the eigenfunction of the system and λ denotes the associated eigenvalue. In this preliminary study, inspired by notable work in [56], we utilize the one-group steady-state diffusion equation for criticality, framing it as a generalized eigenvalue problem. It is expressed as:

$$-\nabla \cdot (D\nabla\phi) + \Sigma_a \phi = \lambda \nu \Sigma_f \phi,$$

where eigenfunction ϕ denotes the neutron flux, which is a scalar quantity used in nuclear reactor physics corresponding to the total length traveled by all free neutrons per unit time and volume. Σ_a, Σ_f, and ν denote the absorption cross-section, fission cross-section, and average number of neutrons produced per fission event, respectively. We follow [56] and use the diffusion coefficient approximation $D = \frac{1}{3(\Sigma_a + \Sigma_s)}$, where Σ_s denotes the cross-section for neutron scattering in a different direction. The eigenvalue λ is obtained by multiplying the neutron source in Eq. (4). This value balances the terms that produce neutrons with those that account for losses and is defined as the reciprocal of k_{eff}, i.e., $\lambda = \frac{1}{k_{\text{eff}}}$, where

$$k_{\text{eff}} = \frac{\text{number of neutrons in one generation}}{\text{number of neutrons in the preceding generation}}.$$

Two main boundary conditions are imposed on the diffusion equation. One condition represents a surface on which neutrons are reflected back into the domain (reflective condition), and the other represents surfaces that allow neutrons to escape from the system (vacuum or bare condition). Both conditions are satisfied by relating the flux solution to its gradient on the boundary:

$$D\nabla\phi \cdot \mathbf{n} = \begin{cases} 0, & \text{bare surface}, \\ \frac{1}{2}\phi, & \text{reflective surface}, \end{cases}$$

where $\mathbf{n}$ denotes an outward-pointing normal to the surface.

A. PINN as an Eigenvalue Solver

In this subsection, we discuss using PINN to solve the generalized eigenvalue problem (2). The eigenfunction of operator L is approximated by N_θ, i.e., $\phi(x) = N_\theta(x)$. Then, $L\phi$ and $B\phi$ can be computed using AD. For the boundary conditions, a penalty term is added to the loss function in PINN, which penalizes

the discrepancy between the approximated value on the boundary and the exact boundary condition:

$$\text{Loss}_b = \sum_{i=1}^{N_b} |B\Phi(x_i) - g(x_i)|^2,$$

where N_b denotes the number of sampling points on $\partial\Omega$ and x_i is one point in the sampling set $\{x_i\}_{i=1}^{N_b}$.

As mentioned previously, we are concerned with the smallest eigenvalue and associated eigenfunction. Hence, the eigenvalue (Rayleigh quotient) is viewed as a loss term in PINN to attain the lowest eigenvalue:

$$\lambda = \frac{\langle L\phi, \phi\rangle}{\langle \phi, \phi\rangle}.$$

Finally, the total loss function in PINN corresponds to the weighted sum of the two objectives (8) and (9):

$$\text{Loss}_{\text{total}} = \alpha\lambda^2 + \beta\text{Loss}_b,$$

where α and β correspond to the weights.

Unfortunately, PINN does not work for the cases in this study and even performs worse than DRM. Therefore, we only compared the results of our method with those of DRM.

Remark 1. It should be noted that the square of the eigenvalue is used as the loss function in PINN. Given that the smallest eigenvalue implies that the absolute value is the smallest, PINN would attempt to determine a negative infinity value without using a square term.

III. Methodologies

In this section, we extend our previous study [49] and discuss the use of a neural network to numerically solve the smallest eigenvalue problem. The main concept is to use a neural network to approximate the eigenfunction and compute the eigenvalue based on the Rayleigh quotient using an eigenvector expressed by points calculated from the eigenfunction.

A. Neural Network Architecture

We discuss the neural network structure used to approximate eigenfunction $\phi(x)$. The neural network architecture employed in this study is the same as ResNet [60], which is built by stacking several residual modules. It is one of the most popular models used in deep learning and is also commonly used in the field

of solving PDEs via neural networks [24, 61, 62]. Each module has one skip connection, and each block consists of two fully connected layers as shown in Figure 1 [Figure 1: see original paper].

In the network, let $x, x_k \in \mathbb{R}^d$ be the input and let W^l and b^l, $l = 1, 2, 3, 4, 5$ be the parameters in the fully connected layers. We use W_r^k and b_r^k for $k = 1, 2$ to denote the parameters in the residual connections. The results s_1 and s_2 for the modules can be expressed as follows:

$$\begin{aligned} s_1 &= \sigma(W^2(\sigma(W^{1x} + b^1)) + b^2) + r_1(x), \\ s_2 &= \sigma(W^4(\sigma(W_1^{3s} + b^3)) + b^4) + r_2(s_1), \end{aligned}$$

where σ denotes the activation function, chosen as tanh. Furthermore, r_k, $k = 1, 2$ are functions in the residual connections, which can be represented as:

$$r_k(x_k) = \sigma(W_r^k x_k + b_r^k).$$

Subsequently, the neural network $N_\theta(x)$ is expressed as:

$$N_\theta(x) = W^5 s_2(s_1(x)) + b^5.$$

Therefore, the eigenfunction $\phi(x) = N_\theta(x)$, where θ denotes the neural network parameters.

Remark 2. Given that $\phi(x)$ represents the scalar neutron population and denotes the density of the free-moving neutron distribution over the spatial domain, we require $\phi(x) \geq 0$. Therefore, the eigenfunction can be expressed as $\phi(x) = (N_\theta(x))^2$.

B. Recap of Inverse Power Method Neural Network

We consider the following linear eigenvalue problem:

$$\begin{cases} L\phi = \lambda\phi, & \text{in } \Omega, \\ B\phi = g, & \text{on } \partial\Omega, \end{cases}$$

which differs from the generalized eigenvalue problem $L\phi = \lambda Q\phi$ in Eq. (17). The main idea is that it is not necessary to calculate L^{-1}, and the eigenfunction can be approximated iteratively by minimizing the defined loss to approach Eq. (17):

$$\text{Loss}_{\text{ipmnn}}(\theta) = \sum_{i=1}^{N} \left(\frac{L\Phi_k(x_i)}{\|L\Phi_k\|} - \Phi_{k-1}(x_i) \right)^2,$$

where $x_i \in S$ is the dataset and N denotes the number of points in S. The eigenvalue in the k-th iteration is obtained using:

$$\lambda_k = \frac{\langle L\Phi_k, \Phi_k \rangle}{\langle \Phi_k, \Phi_k \rangle}.$$

C. Generalized Inverse Power Method Neural Network

In standard nuclear engineering procedures, we normally employ the inverse power method to solve for the smallest eigenvalue and associated eigenvector, which relies on the discretization of Eq. (3). IPMNN [49] was proposed to solve the smallest eigenvalue problem, representing a mesh-free method realized by a neural network. However, the method is restricted to solving the equation $L\phi = \lambda\phi$, which is a simple form. Therefore, we propose GIPMNN to solve Eq. (3).

The algorithm details of GIPMNN are presented in Algorithm 1. First, the generalized inverse power method is used to solve Eq. (20). The key step is given in Eq. (21), where A and B are two matrices and λ_{k-1} and ϕ_{k-1} are the results from the previous iteration. Therefore, λ_k and ϕ_k are obtained using Eq. (21):

$$A\phi = \lambda B\phi,$$

$$A\phi_k = \lambda_{k-1} B\phi_{k-1}, \quad \lambda_k = \frac{\langle A\phi_k, \phi_k \rangle}{\langle B\phi_k, \phi_k \rangle}.$$

Inspired by the idea of the inverse power method, IPMNN [49] was proposed to solve for the smallest eigenvalue and associated eigenfunction. Equation (16) depicts the key step of the inverse power method, and Eq. (17) is analogous to Eq. (16), where A denotes a matrix and λ_{k-1} and ϕ_{k-1} denote the results from the previous iteration. Therefore, λ_k and ϕ_k are obtained using Eq. (16):

$$\begin{cases} p_k = A^{-1}\phi_{k-1}, \\ \phi_k = \frac{p_k}{\|p_k\|}, \\ \lambda_k = \frac{\langle A\phi_k, \phi_k \rangle}{\langle \phi_k, \phi_k \rangle}. \end{cases}$$

We use a neural network N_θ to represent the approximated eigenfunction Φ. In the algorithm of GIPMNN, Eq. (22) is an analog of Eq. (21), where L and Q denote linear differential operators implemented by AD rather than specially discretized matrices. In a manner similar to the generalized inverse power method, we record the results λ_{k-1} from the previous iteration. In contrast to the generalized inverse power method, instead of recording ϕ_{k-1}, we record $Q\Phi_{k-1}$. It should be noted that Φ_{k-1} denotes the eigenfunction represented by the neural

network in the $(k-1)$-th iteration, and $Q\Phi_{k-1}$ is realized by AD. In the k-th iteration, we directly compute Φ_k using the neural network, i.e., $\Phi_k = N_\theta$, and calculate $L\Phi_k$ using AD. We obtain Φ_k directly through the neural network instead of solving the equation $L\Phi_k = \lambda_{k-1}Q\Phi_{k-1}$. We define the loss function $\text{Loss}_{\text{gipmnn}}$ in Eq. (23) to propel the neural network to learn Φ_k:

$$\text{Loss}_{\text{gipmnn}} = \sum_{i=1}^{N} |L\Phi_k(x_i) - \lambda_{k-1}Q\Phi_{k-1}(x_i)|^2.$$

When the neural network converges, we obtain the smallest eigenvalue, and the associated eigenfunction can be expressed using the neural network.

Remark 3. In the algorithm of GIPMNN, given that the initial function Φ_0 is not represented by the neural network, it is not possible to obtain $Q\Phi_0$ using AD. Therefore, we choose an arbitrary function for $Q\Phi_0$.

The Neumann and Robin boundary conditions were used for the eigenvalue problem. It is difficult to enforce them by encoding the boundary conditions into a neural network, as in [49, 63, 64], where Dirichlet and periodic boundary conditions are used. We form the loss function Loss_b in Eq. (8), where N_b denotes the number of sampling points on $\partial\Omega$, and x_i is a point in the sampling set $\{x_i\}_{i=1}^{N_b}$. Therefore, the loss function is defined as:

$$\text{Loss}_b = \begin{cases} \sum_{i=1}^{N_b} |D(x_i)\nabla\Phi(x_i)\cdot\mathbf{n}|^2, & \text{bare surface,} \\ \sum_{i=1}^{N_b} |D(x_i)\nabla\Phi(x_i)\cdot\mathbf{n} - \Phi(x_i)|^2, & \text{reflective surface,} \end{cases}$$

where the surface indicates that neutrons are reflected back into the domain or that the surface allows neutrons to escape from the system.

The total loss function denotes the weighted sum of the objectives (23) and (8):

$$\text{Loss}_{\text{total}} = \alpha\text{Loss}_{\text{gipmnn}} + \beta\text{Loss}_b.$$

For the process of GIPMNN, refer to Algorithm 1.

Remark 4. In Eq. (26), α and β denote the weights of the two losses. It is noted that $\beta \geq \alpha$ when training the neural network, particularly in the GIPMNN method. Given this condition, it is easier for the neural network to determine the eigenvalue and eigenfunction.

D. Physics-Constrained Generalized Inverse Power Method Neural Network

Although GIPMNN can solve the eigenvalue problems of Eq. (3), it remains difficult to solve eigenvalue problems with discontinuous coefficients in different regions. We discuss interface problems, which imply that the eigenfunction may

be continuous at the interface while the derivatives of the eigenfunction may not be continuous at the interface. The enforcement of interface conditions is crucial for GIPMNN.

In this study, inspired by the idea of a piecewise deep neural network [58], we propose PC-GIPMNN to solve eigenvalue problems with interface conditions. However, instead of employing different neural networks in different subdomains, we use only one neural network with multiple neurons in the output layer, as shown in Fig. 2 [Figure 2: see original paper]. Each neuron in the output layer corresponds to a subdomain, allowing us to obtain outputs in different subdomains that can be used to enforce conditions at the interface.

Suppose there are two domains Ω_l and Ω_r with an interface Γ, which is the cross region between the two domains. Given the properties of the neutron population $\phi(x)$, we can summarize that the eigenfunction will satisfy two interface conditions:

$$\begin{cases} \phi_l = \phi_r, & \text{on } \Gamma, \\ -D_l \nabla \phi_l \cdot \mathbf{n} = -D_r \nabla \phi_r \cdot \mathbf{n}, & \text{on } \Gamma, \end{cases}$$

where ϕ_l and ϕ_r represent the eigenfunctions defined in Ω_l and Ω_r, respectively, and $\mathbf{n}$ denotes the normal vector pointing from Ω_r to Ω_l. D_l and D_r are the coefficients defined in Ω_l and Ω_r, respectively, which are discontinuous at the interface. Equation (27) indicates that the eigenfunction is continuous at the interface (i.e., the neutron flux is continuous), while Eq. (28) indicates that neutron current is continuous at the interface.

Assume that S_Γ corresponds to the set of points at the interface Γ and $|S_\Gamma|$ denotes the number of points in S_Γ. We then introduce two penalty terms to enforce the two interface conditions:

$$\text{Loss}_{i1} = \sum_{x_i \in S_\Gamma} |\Phi_l(x_i) - \Phi_r(x_i)|^2,$$

$$\text{Loss}_{i2} = \sum_{x_i \in S_\Gamma} |(-D_l \nabla \Phi_l(x_i) \cdot \mathbf{n}) - (-D_r \nabla \Phi_r(x_i) \cdot \mathbf{n})|^2.$$

By combining (26), (29), and (30), the total loss defined in our PC-GIPMNN method is:

$$\text{Loss}_{\text{total}} = \alpha \text{Loss}_{\text{gipmnn}} + \beta \text{Loss}_b + \gamma \text{Loss}_{i1} + \delta \text{Loss}_{i2},$$

where α, β, γ, and δ are the weights of the different losses. In subsequent experiments, we chose all weights as 1. The study focused on the proposed algorithms and neglected the influence of weights, expecting that our algorithms are universal and achieve better results even without adjusting the weights.

Moreover, the eigenfunction can be represented as:

$$\phi = l_l\phi_l + l_r\phi_r,$$

where l_l and l_r denote indicator functions: $l_l = 1$ in Ω_l, $l_l = 0$ in Ω_r, $l_r = 1$ in Ω_r, and $l_r = 0$ in Ω_l.

Remark 5. Conservative PINN (cPINN) [53] developed PINN for each subdomain and considered additional conservation laws along the subdomains' interfaces (a general consideration in reactor physics [11]). However, in neural network training, the eigenvalue is involved as a variable to be optimized, and the numerical examples presented correspond to only one-dimensional cases. Furthermore, the relative errors of k_{eff} in these cases are 4.4800×10^{-4} and 3.3500×10^{-4}. Similarly, we use Eqs. (29) and (30) to enforce interface conditions. As shown below, our methods are more generic and yield better results.

Remark 6. In [45], the impact of interface conditions was ignored, and the relative errors of k_{eff} in their cases corresponded to 1.3×10^{-3} and 4.4×10^{-3}, respectively, and the study did not involve the smallest eigenvalue problem. In this study, we obtain the lowest eigenvalue using the inverse power method as opposed to using a free learnable parameter to approximate the eigenvalue. Our numerical results demonstrate that accurate results can be obtained in more complicated cases.

E. Deep Ritz Method

DRM is a deep-learning-based method for numerically solving variational problems [24]. It reformulates the original PDEs into equivalent variational equations and defines the loss function based on variational formulations. The solutions of PDEs are represented by a neural network, and derivatives are calculated using AD. DRM [24] is also used to solve eigenvalue problems. We specify how to use DRM to solve Eq. (3).

We consider the variational principle of the smallest eigenvalues:

$$\lambda = \frac{\int_\Omega L\phi \cdot \phi \, dx}{\int_\Omega Q\phi \cdot \phi \, dx}, \quad B\phi|_{\partial\Omega} = g,$$

where the Rayleigh quotient is used. The boundary conditions are enforced by adding a penalty term:

$$\int_{\partial\Omega} |B\phi - g|^2 \, ds,$$

and the total loss function Loss_{drm} is defined as:

$$\text{Loss}_{\text{drm}} = \alpha \frac{\int_\Omega L\phi \cdot \phi \, dx}{\int_\Omega Q\phi \cdot \phi \, dx} + \beta \int_{\partial\Omega} |B\phi - g|^2 \, ds,$$

where α and β denote the weights of different losses. We chose $\alpha = 1$ and $\beta = 1$ for our experiments.

After the optimal approximation is obtained by solving the optimization problem (35), we obtain the smallest eigenvalue $\frac{\int_\Omega L\phi \cdot \phi \, dx}{\int_\Omega Q\phi \cdot \phi \, dx}$ and the eigenfunction represented by the trained neural network. It should be noted that $L\phi$, $Q\phi$, and $B\phi$ are computed using AD. For the DRM process, refer to Algorithm 2.

Remark 7. We enforced the boundary condition by adding a penalty term (34). However, if the boundary condition is Neumann or Robin, we do not use the penalty term (34) because the boundary condition is incorporated into the Rayleigh quotient based on Green's first identity [65].

IV. Experiments

In this section, we present numerical experiments to compare the applicability and accuracy of GIPMNN, PC-GIPMNN, and DRM for solving the smallest eigenvalue problems in reactor physics. In all experiments below, we chose the Adam optimizer with an initial learning rate of 10^{-3} to minimize the loss function. Furthermore, we trained the neural network with the ResNet architecture on a server equipped with a CentOS 7 system, one Intel Xeon Platinum 8358 2.60-GHz CPU, and one NVIDIA A100 80GB GPU. Unless otherwise specified, the activation function was selected as the tanh function.

A. One-Dimensional Slab Reactor

We consider a one-dimensional case with a simple slab reactor consisting of a domain bounded by vacuum or bare surfaces, as shown in Figure 3 [Figure 3: see original paper]. The length of each slab reactor is 10 cm, consisting of fuel and control rod regions labeled 1, 2, and 3. The control rods are located between 2.2 and 2.5 cm and between 7.5 and 7.8 cm on the x-axis.

As shown in Fig. 3, there are two control rods in the one-dimensional slab reactor. Either device can be withdrawn or inserted. Three scenarios were designed to model the reactor and completely simulate the actions of the control rods. The three situations are labeled F1, F2, and F3 in Table 1 . They indicate that both rods are withdrawn, only the left rod is inserted, and only the right rod is inserted. We also considered three other problems, R1, R2, and R3 in Table 1. Problem R1 is the same as F1, designed to simulate the withdrawal of both control rods. Here, we use R1 to facilitate comparison with R2 and R3. Although problems R2 and R3 denote that all control rods are inserted, problem R2 resembles heavily inserted control rods, and problem R3 resembles slightly inserted control rods.

To generate the necessary data to validate the accuracy of GIPMNN, PC-GIPMNN, and DRM, FreeFEM [67–73] is utilized to solve these problems in Table 1. FreeFEM is a partial differential equation solver for nonlinear multi-physics systems using the finite element method. We choose the number of cells in the x-direction as $N = 1001$ and mesh size $\Delta x = 10^{-3}$. The baseline solution is obtained by FEM, and uniform cells are used to train GIPMNN, PC-GIPMNN, and DRM.

Remark 8. When solving all parameter-dependent problems below, parameters Σ_a, Σ_s, and $\nu\Sigma_f$ are selected based on whether the data point x belongs to a related region. For PC-GIPMNN, data points on the interface are considered to belong to multiple regions simultaneously, allowing us to enforce interface conditions using data points on the interface.

Remark 9. As unsupervised algorithms, our methods use only points to train a neural network without any prior knowledge. Therefore, there is no test set for the proposed algorithm.

1. Using GIPMNN to Solve for Eigenvalue In this one-dimensional example, we select 20 neurons for each hidden layer in ResNet and $N_{\text{epoch}} = 50000$. Without loss of generality, the weights α and β of the different losses are not adjusted to achieve better results; therefore, α and β were set to one.

In Eq. (21), we must solve for ϕ_k using the eigenvalues λ_{k-1} and ϕ_{k-1}. To accelerate the training process and obtain more accurate results using Algorithm 1, we split the iterations in the original Algorithm 1 into inner and outer iterations, which can be observed in Algorithm 3, where N_{inner} and N_{outer} denote the number of inner and outer iterations, respectively.

We chose the ratios of outer to inner iterations as 1:1, 1:10, 1:100, 1:1000, and 1:10000 to investigate the effect of the ratio on the results. Ratio 1:n implies that the outer code is executed once, whereas the inner code is executed n times. For comparison, the total number of iterations was fixed at $N_{\text{total}} = 100000$ and $N_{\text{total}} = N_{\text{inner}} \times N_{\text{outer}}$.

The relative errors of k_{eff} and eigenfunction for different ratios of outer and inner iterations during the training process are shown in Figure 4 [Figure 4: see original paper]. The results of $k_{\text{eff}}^{\text{rel}}$ are shown in the first row, and those of ϕ^{rel} are shown in the second row, where the relative errors of k_{eff} and eigenfunction are calculated by:

$$k_{\text{eff}}^{\text{rel}} = \frac{|k_{\text{eff}}(\text{FEM}) - k_{\text{eff}}(\text{NN})|}{k_{\text{eff}}(\text{FEM})},$$

$$\phi^{\text{rel}} = \frac{|\phi(\text{FEM}) - \phi(\text{NN})|}{|\phi(\text{FEM})|}.$$

As shown in the figures, the best ratio is 1:1 because the relative errors of k_{eff} and eigenfunction for ratio 1:1 are relatively smaller than the others when training the neural network. Convergence worsens when the ratio of outer to inner iterations changes from 1:1 to 1:10000. Therefore, we trained GIPMNN using a ratio of 1:1 to solve 1D and 2D problems. Moreover, we fixed the number of outer iterations $N_{\text{outer}} = 1000$ and retrained the neural network using these ratios. The results are shown in Fig. 5 [Figure 5: see original paper]. Opposite results are observed when compared with the results in Fig. 4. The ratio 1:1 is the worst ratio because increases in inner iterations lead to a better approximation of the eigenfunction in the next outer iteration, which is consistent with the results of the inverse power method.

2. Using PC-GIPMNN to Solve for Eigenvalue We divided the slab reactor into five parts from left to right. The output layer included five neurons, and five functions were defined in different subdomains. Here, u, w, and q denote functions defined in Region 1, while v and p are those defined in Regions 2 and 3, respectively.

As shown in Figure 3, the four interface points are denoted as $x_{i1} = 2.2$, $x_{i2} = 2.5$, $x_{i3} = 7.5$, and $x_{i4} = 7.8$. The interface conditions (29) and (30) are implemented as loss functions defined in (38) and (39) for the 1D example:

$$\text{Loss}_{i1} = |u(x_{i1}) - v(x_{i1})|^2 + |v(x_{i2}) - w(x_{i2})|^2 + |w(x_{i3}) - p(x_{i3})|^2 + |p(x_{i4}) - q(x_{i4})|^2,$$

$$\begin{aligned}\text{Loss}_{i2} = {} & |(-D_1 u_x(x_{i1})) - (-D_2 v_x(x_{i1}))|^2 + |(-D_2 v_x(x_{i2})) - (-D_1 w_x(x_{i2}))|^2 \\ & + |(-D_1 w_x(x_{i3})) - (-D_3 p_x(x_{i3}))|^2 + |(-D_3 p_x(x_{i4})) - (-D_1 q_x(x_{i4}))|^2.\end{aligned}$$

The eigenfunction can be expressed as:

$$\phi = ul_1 + vl_2 + wl_3 + pl_4 + ql_5,$$

where l_1, l_2, l_3, l_4, and l_5 are indicator functions that are 1 or 0 based on whether the input x is inside or outside the subdomain.

3. Using DRM to Solve for Eigenvalue The configuration of DRM is identical to that of GIPMNN. Given that it is a variational formulation and the boundary condition is incorporated into the Rayleigh quotient, we do not use a penalty term to enforce the boundary condition. The loss function Loss_{drm} is defined as:

$$\text{Loss}_{\text{drm}} = \frac{\text{Length}_{\text{slab}} \sum_{i=1}^{N} D|\nabla\phi(x_i)|^2 + \frac{1}{2}(\phi(x_l)^2 + \phi(x_r)^2)}{\text{Length}_{\text{slab}} \sum_{i=1}^{N} \nu\Sigma_f \phi(x_i)^2} + \frac{\text{Length}_{\text{slab}} \sum_{i=1}^{N} \Sigma_a \phi(x_i)^2}{\text{Length}_{\text{slab}} \sum_{i=1}^{N} \nu\Sigma_f \phi(x_i)^2},$$

where $\text{Length}_{\text{slab}}$ denotes the length of the slab reactor, x_l and x_r denote points representing the positions of the endpoints of the slab reactor, and N denotes the number of points used to approximate the integral. Therefore, the smallest eigenvalue λ can be obtained after the neural network converges.

4. Results The results for the one-dimensional example are listed in Tables 2, 3, and 6. The values of k_{eff} and their relative errors are listed in Table 2 . For problems F1 and R1, which have continuous coefficients, the results obtained by GIPMNN are better than those obtained by PC-GIPMNN and DRM. For problem F2, the relative error of k_{eff} obtained by DRM is better than those obtained by other methods. For other problems with discontinuous coefficients, the results obtained using PC-GIPMNN are better than those obtained by GIPMNN and DRM. The relative error of k_{eff} computed by PC-GIPMNN is approximately 10^{-5}, which is lower than 10^{-4} as computed by DRM. The relative errors in the eigenfunctions are listed in Table 3 . For all problems, PC-GIPMNN can attain better results than GIPMNN and DRM.

In Fig. 6 [Figure 6: see original paper], the eigenfunctions of GIPMNN, PC-GIPMNN, and DRM are compared with those of FEM. Specifically, we follow the conventional normalization process in nuclear reactor physics [1], where the normalization constant is generally computed to make the average reactor power equal to unity; thus, the eigenfunctions are normalized by:

$$\phi_{\text{norm}} = \frac{\phi(x_i)}{\sum_{i=1}^{N} \phi(x_i)}.$$

The figures in the first row show results for problems F1, F2, and F3, and those in the second row show results for problems R1, R2, and R3. In each figure, we plot the eigenfunction obtained by FEM and compare the relative errors of the eigenfunctions computed by GIPMNN, PC-GIPMNN, and DRM. Evidently, the results obtained by PC-GIPMNN are better than those obtained by the other methods, which is consistent with the results in Table 3.

As reported in a previous study [49], IPMNN can attain more accurate eigenvalues compared to DRM when linear eigenvalue problems without interfaces are considered. Therefore, IPMNN and GIPMNN are suitable for determining the eigenfunction of a problem in strong form, while DRM is similar to FEM in that it is applicable for finding the eigenfunction of a problem in weak form. Consequently, DRM performs better than GIPMNN in this one-dimensional example with discontinuous coefficients. However, given that interface conditions are well implemented in PC-GIPMNN, it successfully learns the eigenvalue and eigenfunction and achieves better results than GIPMNN and DRM, as shown in Tables 2 and 3.

Remark 10. Specifically, DRM is applicable for determining the eigenfunction of a problem in weak form, which implies that the eigenfunction exhibits

low regularity. In contrast, as shown in the implementation of GIPMNN, it is necessary for the eigenfunction obtained from GIPMNN to exhibit high regularity. Therefore, the learned eigenfunction is in strong form. Hence, GIPMNN is unable to obtain accurate values at the interface. However, PC-GIPMNN does not require the eigenfunction to exhibit higher regularity at the interface but instead guarantees continuity and physical constraints by enforcing interface conditions. Therefore, PC-GIPMNN successfully learns the eigenvalue and eigenfunction and obtains better results compared to GIPMNN and DRM.

B. Two-Dimensional Reactor

As shown in Figure 7 [Figure 7: see original paper], a two-dimensional reactor is modeled in a square-shaped domain with 90-cm sides. The reactor is surrounded by a neutron reflector with graphite material, which implies that a Robin boundary condition is applied. The main bulk of the reactor corresponds to the fuel region. Within its central region, four control rods can be inserted or withdrawn. When the control rods are withdrawn, materials in the regions are replaced with water, corresponding to common practice in many reactor designs. Table 4 lists five different materials used in the mock reactor. The two types of fuels in Table 4 are designed to simulate different fuel materials: Fuel type 1 defines the standard fuel used in most problems, and Fuel type 2 defines an adjusted fuel composition different from Fuel type 1. Fuel type 2 in problem R7 is used to test whether our methods can be affected by different fuel types.

As shown in Table 5 , there are 12 problems for validating the accuracy of the proposed method. Five full models, labeled F1–F5, were proposed to simulate the reactor with all control rods removed and then with only one control rod inserted in the control rod regions. As previously discussed, when a control rod is removed, the material in the region is replaced with water; therefore, W is used to denote water in Table 5. Seven other reactor configurations, denoted by R1–R7, were proposed to simulate cases where more control rods were inserted. Problems R1 and R2 are equivalent to full model problems F1 and F2. Problems R3–R6 utilized different combinations of inserted and withdrawn control rods. It should be noted that in problem R7, the material configuration differs from that of other problems: the fuel type was replaced with Fuel type 2, and the control rods were assumed to be partially inserted, implying that the materials in the regions correspond to a mix of control rod and water materials.

In the two-dimensional case, we use FreeFEM to solve the problems listed in Table 5. We chose uniform grids with $\Delta x = \frac{1}{90}$ and $\Delta y = \frac{1}{90}$. We trained GIPMNN, PC-GIPMNN, and DRM with $N_x = 91$ and $N_y = 91$.

1. Using GIPMNN and PC-GIPMNN to Solve for Eigenvalue The number of points $N = N_x N_y$ is used to calculate $\text{Loss}_{\text{gipmnn}}$, and the number of points $N_b = 2(N_x - 2) + 2(N_y - 2) + 4$ is used to calculate Loss_b. The number of neurons is 20 for each hidden layer in ResNet, and $N_{\text{epoch}} = 500000$ for both GIPMNN and PC-GIPMNN. Without loss of generality, α and β are set to one.

As mentioned previously, we use the optimal ratio of outer to inner iterations, which is 1:1.

The 2D reactor is divided into six parts, as shown in Figure 7. The output layer includes six neurons, and six functions are defined in different subdomains and labeled as u, v, w, r, p, and q, where u, v, w, and r denote functions defined in CR1, CR2, CR3, and CR4, and p and q denote functions defined for Fuel and Graphite, respectively. S_{CR1}, S_{CR2}, S_{CR3}, S_{CR4}, and S_{GF} denote different datasets at different interfaces.

For PC-GIPMNN, interface conditions (29) and (30) are implemented as loss functions (43) and (44) in the 2D example:

$$\text{Loss}_{i1} = \frac{1}{|S_{CR1}|}\sum_{x_i\in S_{CR1}}|u(x_i)-p(x_i)|^2+\frac{1}{|S_{CR2}|}\sum_{x_i\in S_{CR2}}|v(x_i)-p(x_i)|^2+\frac{1}{|S_{CR3}|}\sum_{x_i\in S_{CR3}}|w(x_i)-p(x_i)|^2+\frac{1}{|S_{CR4}|}$$

$$\begin{aligned}\text{Loss}_{i2} = &\frac{1}{|S_{CR1}|}\sum_{x_i\in S_{CR1}}|(-D_{CR1}\nabla u(x_i)\cdot\mathbf{n})-(-D_{Fuel}\nabla p(x_i)\cdot\mathbf{n})|^2\\ &+\frac{1}{|S_{CR2}|}\sum_{x_i\in S_{CR2}}|(-D_{CR2}\nabla v(x_i)\cdot\mathbf{n})-(-D_{Fuel}\nabla p(x_i)\cdot\mathbf{n})|^2\\ &+\frac{1}{|S_{CR3}|}\sum_{x_i\in S_{CR3}}|(-D_{CR3}\nabla w(x_i)\cdot\mathbf{n})-(-D_{Fuel}\nabla p(x_i)\cdot\mathbf{n})|^2\\ &+\frac{1}{|S_{CR4}|}\sum_{x_i\in S_{CR4}}|(-D_{CR4}\nabla r(x_i)\cdot\mathbf{n})-(-D_{Fuel}\nabla p(x_i)\cdot\mathbf{n})|^2\\ &+\frac{1}{|S_{GF}|}\sum_{x_i\in S_{GF}}|(-D_{Fuel}\nabla p(x_i)\cdot\mathbf{n})-(-D_{Graphite}\nabla q(x_i)\cdot\mathbf{n})|^2.\end{aligned}$$

The eigenfunction is expressed as:

$$\phi = ul_1 + vl_2 + wl_3 + rl_4 + pl_5 + ql_6,$$

where l_1, l_2, l_3, l_4, l_5, and l_6 denote indicator functions.

Remark 11. In the experiment, it was important to use Eq. (42) instead of the original L^2 norm, which is too small to optimize the neural network. Specifically, the L^2 norm of the eigenfunction corresponds to one if we attempt to find a normalized eigenfunction. If the total number of points N is excessively high, then the value of each component of the eigenvector is excessively low, making it difficult for the neural network to learn the eigenfunction.

2. Using DRM to Solve for Eigenvalue and the Failure of DRM in 2D Experiments Given that the homogeneous Neumann boundary condition is used, the loss function in DRM can be defined as:

$$\text{Loss}_{\text{drm}} = \frac{\text{Area}_{\text{square}} \sum_{i=1}^{N} D|\nabla\phi(x_i)|^2 + \text{Area}_{\text{square}} \sum_{i=1}^{N} \nu\Sigma_f\phi(x_i)^2}{\text{Area}_{\text{square}} \sum_{i=1}^{N} \Sigma_a\phi(x_i)^2},$$

where $\text{Area}_{\text{square}}$ denotes the area of the square domain shown in Fig. 7.

We use $N_{\text{epoch}} = 500000$ in DRM. First, we found that DRM can learn the eigenvalues and eigenfunctions at an early stage of the training process. Subsequently, DRM attempts to learn a smaller eigenvalue after it approaches the true eigenvalue. Finally, DRM successfully learns a smaller eigenvalue that may be close to the true eigenvalue but a terrible eigenfunction that is meaningless and far from the true eigenfunction. This phenomenon is illustrated in Fig. 8 [Figure 8: see original paper].

As listed in Table 6 , although the neural network in DRM fails to learn the eigenfunction, the eigenvalue is close to the true value. This phenomenon may be caused by minimization of the Rayleigh quotient. As mentioned in [49], the loss approaches zero, whereas the eigenvalue may not reach zero. In Figure 8, we observe that the failure of DRM in the 2D experiments occurs after $N_{\text{epoch}} \geq 60000$. Therefore, we select $N_{\text{epoch}} = 50000$ to train DRM again and record the results. In a previous study [40], the stopping criteria of the training process for PINN were investigated. In future studies, we will follow the technique discussed in [40] to determine the stopping criteria for DRM. In the next section, we compare the results of DRM trained with $N_{\text{epoch}} = 50000$ with those of GIPMNN and PC-GIPMNN.

Remark 12. The numerical results in Fig. 8 are not due to hardware problems or code errors but can be attributed to the nature of the neural network. The eigenvalue was approximated by constructing a Rayleigh quotient in DRM, and the eigenvalue is treated as a loss function to optimize the neural network. However, this mechanism of minimizing the eigenvalue leads to overfitting of the neural network, as the network always attempts to find a point where the loss function tends toward zero.

3. Results Similar to the results of the one-dimensional case, the relative errors of k_{eff} and eigenfunction ϕ are shown in Tables 7 and 8. It can be observed that the relative errors for k_{eff} obtained via all methods are small, and the results for DRM are trained with $N_{\text{epoch}} = 50000$.

For problems F1, F2, F4, and R7, the relative error of k_{eff} obtained by DRM was smaller than that obtained by GIPMNN, except for problems F3, F5, R3, R4, R5, and R6. However, although the relative error of k_{eff} obtained by GIPMNN is small, the relative error of ϕ simulated by GIPMNN is larger than that obtained by DRM. It is observed that $k_{\text{eff}}^{\text{rel}}$ obtained by PC-GIPMNN is smaller

than that obtained by GIPMNN and DRM for all problems. Furthermore, the relative errors of ϕ computed by PC-GIPMNN are smaller than those obtained by GIPMNN and DRM for all problems. Therefore, PC-GIPMNN can successfully learn eigenvalues and eigenfunctions.

In Figs. 9 [Figure 9: see original paper] and 10 [Figure 10: see original paper], the eigenfunctions computed by FEM are shown in the first column, and the relative errors of the eigenfunctions obtained by GIPMNN, PC-GIPMNN, and DRM are shown in the other columns for different problems. It is observed that the relative errors of the eigenfunction computed by PC-GIPMNN and DRM are smaller than those obtained by GIPMNN, which failed to learn some details. Compared to the eigenfunctions computed by FEM, the results obtained by PC-GIPMNN are the best among the three methods.

2D IAEA Benchmark Problem

We also considered the classical 2D IAEA benchmark problem reported in the study by Yang et al. [40], which was modeled using two-dimensional and two-group diffusion equations. Here, the one-group neutron diffusion equation defined in Eq. (4) is considered, and multigroup problems will be addressed in our future study. Its geometry is shown in Fig. 11 [Figure 11: see original paper]. The main bulk of the reactor consists of two fuel regions, labeled 1 and 2, representing two types of fuel materials. Within its central region, there are four control rods, all labeled as 3. The last region, labeled 4, is composed of water. Cross-sectional data for the 2D IAEA benchmark problem are presented in Table 9 . Only one quarter of the reactor is shown in this figure because the rest can be inferred by symmetry along the x- and y-axes.

Therefore, this 2D IAEA benchmark problem is confined to the two types of boundary conditions defined in Eq. (7). The problem uses the Neumann boundary condition on the x- and y-axes and the Robin boundary condition on the other boundaries.

In this two-dimensional case, we used FreeFEM to solve the 2D IAEA benchmark problem with the parameters listed in Table 9. We selected uniform grids with $\Delta x = \frac{1}{170}$ and $\Delta y = \frac{1}{170}$. We trained GIPMNN, PC-GIPMNN, and DRM with $N_x = 171$ and $N_y = 171$.

1. Using GIPMNN and PC-GIPMNN to Solve for Eigenvalue The number of points $N = N_x N_y$ is used to calculate $\text{Loss}_{\text{gipmnn}}$. The number of points N_{Nb} on the x- and y-axes and N_{Rb} on the other boundaries were used to calculate Loss_{Nb} and Loss_{Rb}, which enforce the Neumann and Robin boundary conditions. The number of neurons is 20 for each hidden layer in ResNet, and $N_{\text{epoch}} = 500000$ for GIPMNN and PC-GIPMNN. As mentioned previously, we use the optimal ratio of outer to inner iterations, which is 1:1.

The 2D reactor is divided into seven parts, as shown in Fig. 11. The output layer has seven neurons, and seven functions are defined in different subdomains

and labeled as u, v, w, r, p, q, and h, where u, v, w, and r denote functions defined in the control rods and p, q, and h are functions defined in the fuel and water regions, respectively. S_{up}, S_{vp}, S_{wp}, S_{rp}, S_{rq}, S_{pq}, and S_{qh} denote different datasets at different interfaces.

For PC-GIPMNN, interface conditions (29) and (30) are implemented as loss functions (47) and (48) in the 2D example:

$$\begin{aligned}\text{Loss}_{i1} = &\frac{1}{|S_{up}|}\sum_{x_i\in S_{up}}|u(x_i)-p(x_i)|^2+\frac{1}{|S_{vp}|}\sum_{x_i\in S_{vp}}|v(x_i)-p(x_i)|^2\\&+\frac{1}{|S_{wp}|}\sum_{x_i\in S_{wp}}|w(x_i)-p(x_i)|^2+\frac{1}{|S_{rp}|}\sum_{x_i\in S_{rp}}|r(x_i)-p(x_i)|^2\\&+\frac{1}{|S_{rq}|}\sum_{x_i\in S_{rq}}|r(x_i)-q(x_i)|^2+\frac{1}{|S_{pq}|}\sum_{x_i\in S_{pq}}|p(x_i)-q(x_i)|^2+\frac{1}{|S_{qh}|}\sum_{x_i\in S_{qh}}|q(x_i)-h(x_i)|^2,\end{aligned}$$

$$\begin{aligned}\text{Loss}_{i2} = &\frac{1}{|S_{up}|}\sum_{x_i\in S_{up}}|(-D_3\nabla u(x_i)\cdot\mathbf{n})-(-D_2\nabla p(x_i)\cdot\mathbf{n})|^2\\&+\frac{1}{|S_{vp}|}\sum_{x_i\in S_{vp}}|(-D_3\nabla v(x_i)\cdot\mathbf{n})-(-D_2\nabla p(x_i)\cdot\mathbf{n})|^2\\&+\frac{1}{|S_{wp}|}\sum_{x_i\in S_{wp}}|(-D_3\nabla w(x_i)\cdot\mathbf{n})-(-D_2\nabla p(x_i)\cdot\mathbf{n})|^2\\&+\frac{1}{|S_{rp}|}\sum_{x_i\in S_{rp}}|(-D_3\nabla r(x_i)\cdot\mathbf{n})-(-D_2\nabla p(x_i)\cdot\mathbf{n})|^2\\&+\frac{1}{|S_{rq}|}\sum_{x_i\in S_{rq}}|(-D_3\nabla r(x_i)\cdot\mathbf{n})-(-D_1\nabla q(x_i)\cdot\mathbf{n})|^2\\&+\frac{1}{|S_{pq}|}\sum_{x_i\in S_{pq}}|(-D_2\nabla p(x_i)\cdot\mathbf{n})-(-D_1\nabla q(x_i)\cdot\mathbf{n})|^2\\&+\frac{1}{|S_{qh}|}\sum_{x_i\in S_{qh}}|(-D_1\nabla q(x_i)\cdot\mathbf{n})-(-D_4\nabla h(x_i)\cdot\mathbf{n})|^2.\end{aligned}$$

The eigenfunction can be expressed as:

$$\phi = ul_1 + vl_2 + wl_3 + rl_4 + pl_5 + ql_6 + hl_7,$$

where $l_1, l_2, l_3, l_4, l_5, l_6$, and l_7 are indicator functions.

2. Using DRM to Solve for Eigenvalue Given that the homogeneous Neumann boundary condition is used, the loss function in DRM omits the impact of the Neumann boundary condition and focuses on the Robin boundary condition. The loss function is defined as:

$$\text{Loss}_{\text{drm}} = \frac{\sum_{i=1}^{N} D|\nabla\phi(x_i)|^2 + \text{Length}\sum_{i=1}^{N_{Rb}} \frac{1}{2}\phi(x_i)^2}{\sum_{i=1}^{N} \nu\Sigma_f\phi(x_i)^2} + \frac{\text{Area}\sum_{i=1}^{N} \Sigma_a\phi(x_i)^2}{\sum_{i=1}^{N} \nu\Sigma_f\phi(x_i)^2},$$

where Area denotes the area of all regions, Length indicates the length of the boundaries other than the x- and y-axes in Fig. 11, and N_{Rb} denotes the number of points on the Robin boundary.

3. Results As discussed above, we also trained DRM with $N_{\text{epoch}} = 500000$ and found that DRM failed to learn the eigenfunction again. In this case, DRM attains a good $k_{\text{eff}} = 0.9750$, but the relative errors of k_{eff} and ϕ are 6.4023×10^{-3} and 0.9557, respectively. Therefore, we retrained DRM with $N_{\text{epoch}} = 50000$ and stored the best results.

The relative errors of k_{eff} and eigenfunction ϕ are shown in Tables 10 and 11 . It was observed that all three methods obtained good results, and the relative errors of k_{eff} for DRM were small. However, DRM's ability to learn the eigenfunction was worse than that of GIPMNN and PC-GIPMNN, and the relative errors of ϕ were the largest. Thus, it can be concluded that PC-GIPMNN successfully learned the eigenvalues and eigenfunctions. The same conclusion can be drawn from the graphs in Fig. 12 [Figure 12: see original paper].

For the 1D slab reactor, the computation time of FEM is 1.25 s, and the training times of DRM, GIPMNN, and PC-GIPMNN are 4788.37 s, 10614.57 s, and 16052.16 s, respectively. For the 2D reactor, the computation time of FEM is 3.64 s, and the training times of DRM, GIPMNN, and PC-GIPMNN are 5827.91 s, 18444.39 s, and 108072.41 s, respectively. For the 2D IAEA benchmark, the computation time of FEM is 5.22 s, and the training times of DRM, GIPMNN, and PC-GIPMNN are 29352.83 s, 64546.74 s, and 137812.92 s, respectively.

Although PC-GIPMNN outperformed DRM and GIPMNN, it required significantly more training time to obtain accurate results. Compared with FEM, neural network methods require excessive time. However, neural networks are an emerging method, and we believe they will achieve better results in the near future.

V. Conclusion

In this study, we proposed two methods, GIPMNN and PC-GIPMNN, to solve generalized K-eigenvalue problems in nuclear reactor physics. We also conducted a comprehensive study of GIPMNN, PC-GIPMNN, and DRM. GIPMNN follows

the main idea of the inverse power method to find the smallest eigenvalue. PC-GIPMNN enforces interface conditions through multiple neurons in the output layer. The concept of DRM is to define the Rayleigh quotient function and form an optimization problem. Unlike DRM, which solves for the smallest eigenvalue by directly minimizing the eigenvalue (Rayleigh quotient), GIPMNN and PC-GIPMNN attain the smallest eigenvalue using an iterative method. All methods use neural networks to represent functions, and derivatives are implemented using AD. Finally, we applied these three methods to problems in reactor physics.

Three numerical experiments were conducted to verify the applicability and accuracy of GIPMNN, PC-GIPMNN, and DRM. In the first 1D example, we used inner and outer iterations for simulation. According to our tests, the best ratio of outer to inner iterations was 1:1. Furthermore, we compared the results of GIPMNN, PC-GIPMNN, and DRM with those of FEM. For continuous problems, the solution learned by GIPMNN was more accurate than those learned by DRM and PC-GIPMNN. For interface problems, the eigenvalue and eigenfunction learned by PC-GIPMNN were better than those learned by DRM and GIPMNN, due to the interface conditions implemented in the loss function of PC-GIPMNN.

In the 2D examples, we observed the failure of DRM in the 2D experiments. DRM can learn the eigenvalue and eigenfunction at the early stage of the training process, but the results degrade when N_{epoch} increases. Therefore, we selected $N_{\text{epoch}} = 50000$ to train DRM and compared the results obtained by GIPMNN, PC-GIPMNN, and DRM with those obtained by FEM. The results show that PC-GIPMNN outperforms GIPMNN and DRM for both eigenvalue and eigenfunction results. Moreover, GIPMNN and PC-GIPMNN are more stable than DRM.

Although good results were obtained, several aspects need to be examined in the future. First, given that the architecture of a neural network significantly influences the accuracy of our methods, it is important to design a universal network architecture to achieve high accuracy. For example, MLP is widely used in the current field of solving PDEs using neural networks, and ResNet [60] is effective in improving the convergence rate and may even obtain better results than MLP. Recently, a transformer [74] was used for operator learning and achieved better results. Sifan Wang et al. [62] investigated the effect of MLP on operator learning and proposed a modified MLP [61], which is a new architecture that improves accuracy. Although GIPMNN and PC-GIPMNN are more stable than DRM and PC-GIPMNN is more accurate than DRM, they require a large number of iterations and long training times to achieve good results. Therefore, improving convergence and reducing training time will be investigated in our future work. Furthermore, the failure of DRM in 2D experiments on eigenvalue problems should be studied and clarified. Finally, for interface problems, a suitable sampling algorithm can facilitate the training process and provide better approximations. Future studies could also involve

implementing the proposed networks in emerging reactor digital twins [75–77] as core solvers.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] A. Hébert, *Applied Reactor Physics*, 3rd Edition, (Presses internationales Polytechnique, 2020)
[2] T.B. Fowler, D.R. Vondy, *Nuclear reactor core analysis code: Citation.* (Jan 1969) https://www.osti.gov/biblio/4772428
[3] A. Hébert, Development of the nodal collocation method for solving the neutron diffusion equation. *Annals Nucl. Energy* 14, 527–541 (1987). doi:https://doi.org/10.1016/0306-4549(87)90074-0
[4] L.A. Semenza, E.E. Lewis, E.C. Rossow, The application of the finite element method to the multigroup neutron diffusion equation. *Nucl. Sci. Eng.* 47, 302–310 (1972). doi:10.13182/NSE72-A22416
[5] A. Hébert, Application of a dual variational formulation to finite element reactor calculations. *Annals Nucl. Energy* 20, 823–845 (1993). doi:https://doi.org/10.1016/0306-4549(93)90076-2
[6] L. Wanai, G. Helin, Z. Chunyu, Solution of neutron diffusion problems by discontinuous Galerkin finite element method with consideration of discontinuity factors. *J. Nucl. Eng. Radiat. Sci.* 9, 031503 (2023). doi:10.1115/1.4055379
[7] R. Lawrence, Progress in nodal methods for the solution of the neutron diffusion and transport equations. *Prog. Nucl. Energy* 17, 271–301 (1986). doi:https://doi.org/10.1016/0149-1970(86)90034-X
[8] K.S. Smith, *An analytic nodal method for solving the two-group, multidimensional, static and transient neutron diffusion equation.* (Mar 1979) http://hdl.handle.net/1721.1/15979
[9] P. An, Y. Ma, P. Xiao, et al., Development and design validation of 3D nuclear reactor corca. *Nucl. Technol.* (2019). doi:https://doi.org/10.1016/j.net.2019.05.015
[10] A. Kuz'min, Iterative methods for solving nonlinear problems of nuclear reactor criticality. *Phys. At. Nucl.* 75, 1551–1556 (2012). doi:10.1134/S1063778812130042
[11] W.M. Stacey, *Nuclear Reactor Physics*, (John Wiley & Sons, 2007). doi:10.1002/9783527611041
[12] S. Marguet, *The Physics of Nuclear Reactors*, (Springer, 2018). doi:https://doi.org/10.1007/978-3-319-59560-3
[13] K. Tang, X. Wan, C. Yang, Das-pinns: A deep adaptive sampling method for solving high-dimensional partial differential equations. *J. Comput. Phys.* 476, 111868 (2023). doi:https://doi.org/10.1016/j.jcp.2022.111868
[14] J.P. Argaud, B. Bouriquet, P. Erhard, et al., Data assimilation in nuclear power plant core, (Springer, 2010). doi:10.1007/978-3-642-12110-4_{61}
[15] H. Gong, Y. Yu, Q. Li, et al., An inverse-distance-based fitting term for

3D-var data assimilation in nuclear core simulation. *Annals Nucl. Energy* 141, 107346 (2020). doi:https://doi.org/10.1016/j.anucene.2020.107346
[16] S. Cheng, J. Chen, C. Anastasiou, et al., Generalised latent assimilation in heterogeneous reduced spaces with machine learning surrogate models, Vol. 94, (Springer, 2023). doi:10.1007/s10915-022-02059-4
[17] S. Cheng, D. Lucor, J.P. Argaud, Observation data assimilation for dynamical systems. *J. Comput. Sci.* (2021). doi:https://doi.org/10.1016/j.jocs.2021.101405
[18] S. Riva, C. Introini, S. Lorenzi, et al., Hybrid data assimilation methods, part i: Numerical comparison between geim and pbdw. *Annals Nucl. Energy* 190, 109864 (2023). doi:https://doi.org/10.1016/j.anucene.2023.109864
[19] S. Riva, C. Introini, S. Lorenzi, et al., Hybrid data assimilation methods, part ii: Application to the dynasty experimental facility. *Annals Nucl. Energy* 190, 109863 (2023). doi:https://doi.org/10.1016/j.anucene.2023.109863
[20] L.C. Evans, *Partial Differential Equations.* (2010) https://bookstore.ams.org/gsm-19-r/
[21] J. Han, A. Jentzen, et al., Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Commun. Math. Stat.* 5, 349–380 (2017). doi:https://doi.org/10.1007/s40304-017-0117-6
[22] J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning. *Proc. Natl. Acad. Sci.* (2018). doi:https://doi.org/10.1073/pnas.1718942115
[23] J. Sirignano, K. Spiliopoulos, Dgm: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* 375, 1339–1364 (2018). doi:https://doi.org/10.1016/j.jcp.2018.08.029
[24] W. E, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.* 6, 1–12 (2018). doi:10.1007/s40304-018-0127-z
[25] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* 378, 686–707 (2019). doi:https://doi.org/10.1016/j.jcp.2018.10.045
[26] X. Jiang, D. Wang, Q. Fan, et al., Physics-informed neural network for nonlinear dynamics in fiber optics. *Laser & Photonics Rev.* 16, 2100483 (2022). doi:https://doi.org/10.1002/lpor.202100483
[27] D.W. Abueidda, S. Koric, E. Guleryuz, et al., Enhanced physics-informed neural networks for hyperelasticity. *Int. J. Numer. Methods Eng.* 124, 1585–1601 (2023). doi:https://doi.org/10.1002/nme.7176
[28] D.W. Abueidda, Q. Lu, S. Koric, Meshless physics-informed deep learning method for three-dimensional solid mechanics. *Int. J. Numer. Methods Eng.* 122, 7182–7201 (2021). doi:https://doi.org/10.1002/nme.6828
[29] R. Laubscher, Simulation of multi-species flow and heat transfer using physics-informed neural networks. *Phys. Fluids* 33, 087101 (2021). doi:https://doi.org/10.1063/5.0058529
[30] S. Cai, Z. Wang, S. Wang, et al., Physics-informed neural networks for heat transfer problems. *J. Heat Transf.* 143, 060801 (2021). doi:10.1115/1.4050542

[31] Y. Liu, R. Hu, A. Kraus, et al., Data-driven modeling of coarse mesh turbulence for reactor transient analysis using convolutional recurrent neural networks. *Nucl. Eng. Des.* 390, 111716 (2022). doi:https://doi.org/10.1016/j.nucengdes.2022.111716
[32] Y. Chen, L. Lu, G.E. Karniadakis, et al., Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Opt. Express* 28, 11618–11633 (2020). doi:10.1364/OE.384875
[33] T. Kadeethum, T.M. Jørgensen, H.M. Nick, Physics-informed neural networks for solving inverse problems of nonlinear Biot's equations: Batch training. (Jun 2020) https://onepetro.org/ARMAUSRMS/proceedings-abstract/ARMA20/All-ARMA20/447584
[34] S. Cheng, I. Colin Prentice, Y. Huang, et al., Data-driven surrogate model with latent data assimilation: Application to wildfire forecasting. *J. Comput. Phys.* 464, 111302 (2022). doi:https://doi.org/10.1016/j.jcp.2022.111302
[35] S. Cheng, J. Chen, C. Anastasiou, et al., Generalised latent assimilation in heterogeneous reduced spaces with machine learning surrogate models. *J. Sci. Comput.* 94, 11 (2023). doi:10.1007/s10915-022-02059-4
[36] Y. Gao, M.K. Ng, Wasserstein generative adversarial uncertainty quantification for physics-informed neural networks. *J. Comput. Phys.* 463, 111270 (2022). doi:https://doi.org/10.1016/j.jcp.2022.111270
[37] C. Oszkinat, S.E. Luczak, I. Rosen, Uncertainty quantification in estimating blood alcohol concentration from transdermal alcohol level with physics-informed neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 1–8 (2022). doi:10.1109/TNNLS.2022.3140726
[38] Y. Yang, P. Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks. *J. Comput. Phys.* 394, 136–152 (2019). doi:https://doi.org/10.1016/j.jcp.2019.05.027
[39] Y. Liu, D. Wang, X. Sun, et al., Uncertainty quantification for multiphase-CFD simulations of bubbly flows: A machine learning-based Bayesian approach supported by high-resolution experiments. *Reliab. Eng. Syst. Saf.* 212, 107636 (2021). doi:https://doi.org/10.1016/j.ress.2021.107636
[40] Y. Yang, H. Gong, S. Zhang, et al., A data-enabled physics-informed neural network with comprehensive numerical study on solving neutron diffusion eigenvalue problems. *Annals Nucl. Energy* 183, 109656 (2023). doi:https://doi.org/10.1016/j.anucene.2022.109656
[41] I. Ben-Shaul, L. Bar, N. Sochen, Solving the functional eigen-problem using neural networks. (2020) https://arxiv.org/abs/2007.10205
[42] I. Ben-Shaul, L. Bar, N. Sochen, Deep learning solution operators for differential eigenvalue problems. *Neural Comput.* (2023). doi:http://dx.doi.org/10.1162/neco_a_{01583}
[43] H. Jin, M. Mattheakis, P. Protopapas, Unsupervised neural networks for quantum eigenvalue problems. (2020) https://arxiv.org/abs/2010.05075
[44] H. Jin, M. Mattheakis, P. Protopapas, Physics-informed neural networks for quantum eigenvalue problems. *2022 Int. Jt. Conf. Neural Networks (IJCNN)* 1–8 (2022). doi:10.1109/IJCNN55064.2022.9891944
[45] M.H. Elhareef, Z. Wu, Physics-informed neural network method and

application to nuclear reactor calculations: A pilot study. *Nucl. Sci. Eng.* 197, 1–22 (2022). doi:10.1080/00295639.2022.2123211
[46] V.L. Berdichevsky, Variational principles. *Var. Princ. Continuum Mech.* 3–44 (2009). doi:10.1007/978-3-540-88467-5_1
[47] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, et al., Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.* 1–43 (2018).
[48] J. Han, J. Lu, M. Zhou, Solving high-dimensional eigenvalue problems using deep neural networks: A diffusion Monte Carlo-like approach. *J. Comput. Phys.* 423, 109792 (2020). doi:https://doi.org/10.1016/j.jcp.2020.109792
[49] Q. Yang, Y. Deng, Y. Yang, et al., Neural networks based on power method and inverse power method for solving linear eigenvalue problems. *Comput. Math. Appl.* 147, 14–24 (2023). doi:https://doi.org/10.1016/j.camwa.2023.07.013
[50] W.H. Greub, *Linear Algebra*, Vol. 23, (Springer Science & Business Media, 2013). doi:https://doi.org/10.1007/978-3-
[51] M.H. Elhareef, Z. Wu, Extension of physics-informed neural network method for k-eigenvalue diffusion problems. (2022) https://inis.iaea.org/search/search.aspx?orig_q=RN:54002
[52] A.D. Jagtap, E. Kharazmi, G.E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* 365, 113028 (2020). doi:https://doi.org/10.1016/j.cma.2020.113028
[53] J. Wang, X. Peng, Z. Chen, et al., Surrogate modeling for neutron diffusion problems based on conservative physics-informed neural networks with boundary conditions enforcement. *Annals Nucl. Energy* 176, 109234 (2022). doi:https://doi.org/10.1016/j.anucene.2022.109234
[54] A. Quarteroni, G. Rozza, *Reduced Order Methods for Modeling and Computational Reduction*, Vol. 9, (Springer, 2014). doi:https://doi.org/10.1007/978-3-319-02090-7
[55] J.S. Hesthaven, G. Rozza, B. Stamm, *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, (Springer, 2016). doi:https://doi.org/10.1007/978-3-319-
[56] T. Phillips, C.E. Heaney, P.N. Smith, et al., An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion. *Int. J. Numer. Methods Eng.* 122, 3780–3811 (2021). doi:https://doi.org/10.1002/nme.6681
[57] Z. Wang, Z. Zhang, A mesh-free method for interface problems using the deep learning approach. *J. Comput. Phys.* 400, 108963 (2019). doi:https://doi.org/10.1016/j.jcp.2019.108963
[58] C. He, X. Hu, L. Mu, A mesh-free method using piecewise deep neural network for elliptic interface problems. *J. Comput. Appl. Math.* 412, 114358 (2022). doi:https://doi.org/10.1016/j.cam.2022.114358
[59] W.F. Hu, T.S. Lin, M.C. Lai, A discontinuity-capturing shallow neural network for elliptic interface problems. *J. Comput. Phys.* 114358 (2022). doi:https://doi.org/10.1016/j.jcp.2022.111576
[60] K. He, X. Zhang, S. Ren, et al., Deep residual learning for image recognition. (June 2016) https://openaccess.thecvf.com/content_{{{cvpr}}{{2016}}}/html/He{{{Deep}}{{Residual}}}{{
[61] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient

flow pathologies in physics-informed neural networks. *SIAM J. Sci. Comput.* 43, A3055–A3081 (2021). doi:https://doi.org/10.1137/20M1318043
[62] S. Wang, H. Wang, P. Perdikaris, Improved architectures and training algorithms for deep operator networks. *J. Sci. Comput.* 92, 35 (2022). doi:https://doi.org/10.1007/s10915-022-01881-
[63] L. Lyu, K. Wu, R. Du, et al., Enforcing exact boundary and initial conditions in the deep mixed residual method. (Aug 2020). doi:https://doi.org/10.48550/arXiv.2008.01491
[64] S. Dong, N. Ni, A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. *J. Comput. Phys.* 435, 110242 (2021). doi:https://doi.org/10.1016/j.jcp.2021.110242
[65] G.F. Carrier, C.E. Pearson, *Partial Differential Equations: Theory and Technique*, (Academic Press, 2014)
[66] A. Buchan, C. Pain, F. Fang, et al., A POD reduced-order model for eigenvalue problems with application to reactor physics. *Int. J. Numer. Methods Eng.* 95, 1011–1032 (2013). doi:https://doi.org/10.1002/nme.4533
[67] F. Hecht, New development in FreeFEM++. *J. Numer. Math.* 20, 251–266 (2012). doi:10.1515/jnum-2012-0013
[68] J.P. Argaud, B. Bouriquet, F. De Caso, et al., Sensor placement in nuclear reactors based on the generalized empirical interpolation method. *J. Comput. Phys.* 363, 354–370 (2018). doi:https://doi.org/10.1016/j.jcp.2018.02.050
[69] H. Gong, Y. Yu, Q. Li, Reactor power distribution detection and estimation via a stabilized gappy proper orthogonal decomposition method. *Nucl. Eng. Des.* 370, 110833 (2020). doi:https://doi.org/10.1016/j.nucengdes.2020.110833
[70] H. Gong, Z. Chen, W. Wu, et al., Neutron noise calculation: A comparative study between SP3 theory and diffusion theory. *Annals Nucl. Energy* 156, 108184 (2021). doi:https://doi.org/10.1016/j.anucene.2021.108184
[71] H. Gong, Z. Chen, Y. Maday, et al., Optimal and fast field reconstruction with reduced basis and application to reactor monitoring. *Nucl. Eng. Des.* 377, 111113 (2021). doi:https://doi.org/10.1016/j.nucengdes.2021.111113
[72] J. Argaud, B. Bouriquet, H. Gong, et al., Stabilization of (G)EIM in presence of measurement noise: Application to nuclear reactor physics, (Springer, 2017). doi:https://doi.org/10.1007/978-3-319-65870-4_8
[73] H. Gong, S. Cheng, Z. Chen, et al., Data-enabled physics-informed machine learning for reduced-order modeling digital twin: Application to nuclear reactor physics. *Nucl. Sci. Eng.* 196, 668–693 (2022). doi:https://doi.org/10.1080/00295639.2021.2014752
[74] S. Cao, Choose a transformer: Fourier or Galerkin. *Adv. Neural Inf. Process. Syst.* 34, 24924–24940 (2021).
[75] H. Gong, T. Zhu, Z. Chen, et al., Parameter identification and state estimation for nuclear reactor digital twin. *Annals Nucl. Energy* 180, 109497 (2023). doi:https://doi.org/10.1016/j.anucene.2022.109497
[76] H. Gong, S. Cheng, Z. Chen, et al., An efficient digital twin based on machine learning SVD autoencoder and generalized latent assimilation for nuclear reactor physics. *Annals Nucl. Energy* 179, 109431 (2022). doi:https://doi.org/10.1016/j.anucene.2022.109431

[77] H. Gong, S. Cheng, Z. Chen, et al., Data-enabled physics-informed machine learning for reduced-order modeling digital twin: Application to nuclear reactor physics. *Nucl. Sci. Eng.* 196, 668–693 (2022). doi:10.1080/00295639.2021.2014752

Note: Figure translations are in progress. See original paper for figures.