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, Richmond VA, USA

International Conference on Physics of Reactors (PHYSOR 2022)
Pittsburgh PA, May 17th, 2022

VCU College of Engineering

*PINN - Physics Informed Neural Network

Outline

- Background Introduction
- Forward PINN Framework
- PINN for eigenvalue problems
- Numerical Examples
- Conclusions
- Future Work

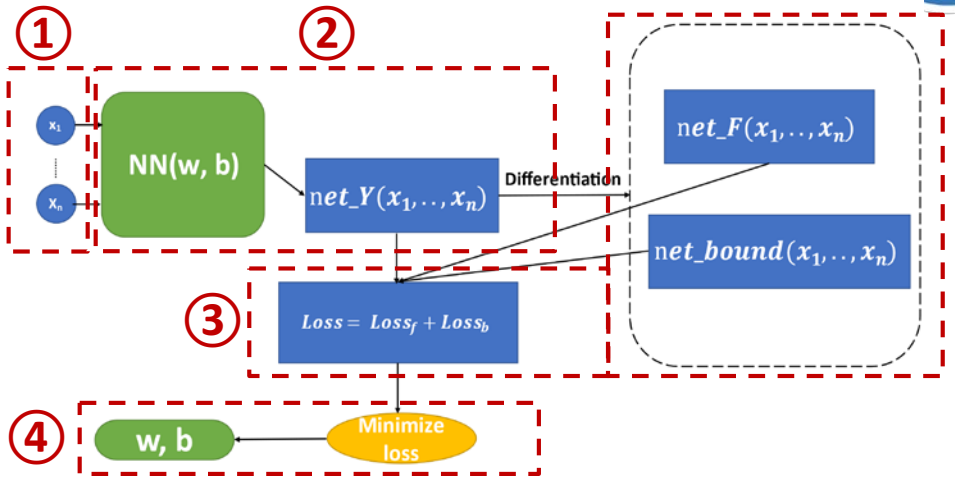
Background

- PINN directly tackles PDEs without prior assumptions or simplifications.
- PINN takes advantages of deep learning technique and greatly simplifies numerical implementation for PDE solvers
- Forward PINN has been picked up in a variety of engineering discipline applications, but rarely seen in nuclear engineering applications
- Preliminary success has been established for PINN in fixed-source diffusion models (previous work)
- This one extends PINN methods to *k*-eigenvlaue diffusion models

Forward PINN Framework

PINN training algorithm:

1. Select training points
2. Evaluate the NN predictions at the set of training points
3. Calculate the deviation between predictions and target values through the loss function
4. Minimize the loss function



PINN Training Scheme

Important note: we do not have direct target values for the predictions, but we know these predictions must satisfy the PDE and the boundary conditions (B.C.).

Explanation Example

The 2D fixed-source diffusion model:

$$-\nabla \cdot D\nabla\phi(x,y) + \Sigma_a\phi(x,y) - S = 0$$

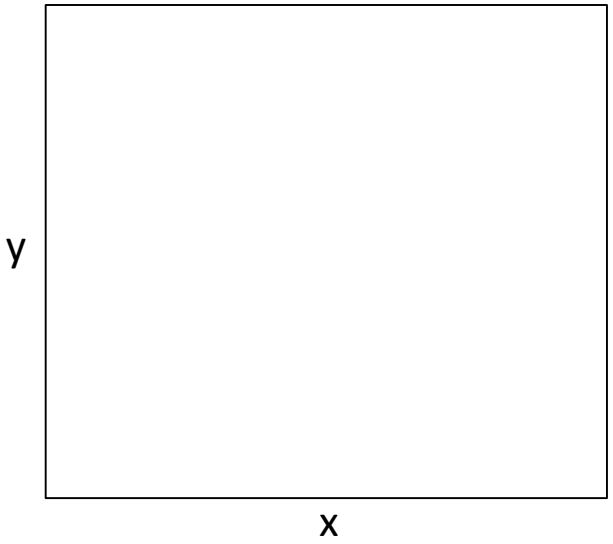
Subject to the following B.C.:

Bottom: $\phi(x,0)$= a

Left: $\phi(0,y)$= a

Top: $\frac{d\phi}{dy} = 0$ (Reflective)

Right: $\frac{d\phi}{dx} + 0.5\phi$=0 (Zero-incoming flux)



Explanation Example: Main PDE

Approximating the solution by NN:

$$\boldsymbol{\phi}(x, y) \approx \boldsymbol{net}_{\boldsymbol{\phi}}(x, y)$$

Evaluate the residual of diffusion model:

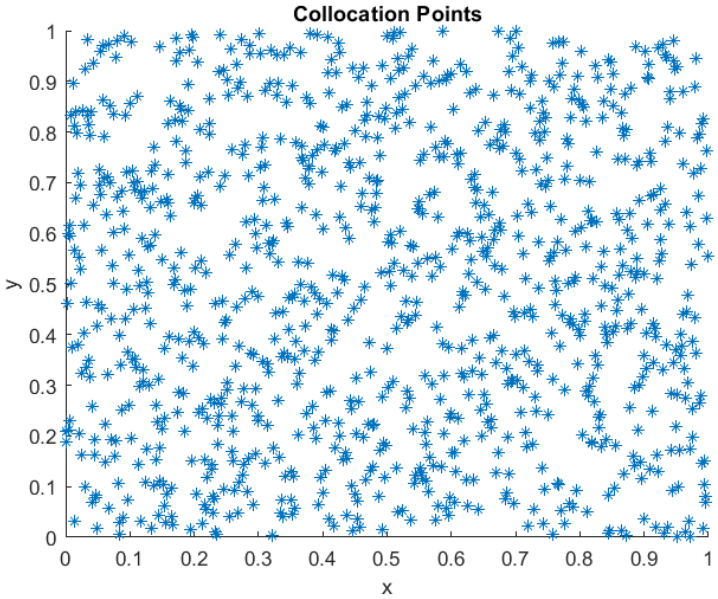
$$\boldsymbol{net}_{\boldsymbol{F}}(x, y) := -\boldsymbol{\nabla} \cdot \boldsymbol{D}\boldsymbol{\nabla}\boldsymbol{net}_{\boldsymbol{\phi}}(x, y) + \boldsymbol{\Sigma}_{\boldsymbol{a}}\boldsymbol{net}_{\boldsymbol{\phi}}(x, y) - \boldsymbol{S}$$

Set the target values:

$$\boldsymbol{net}_{\boldsymbol{F}}(x, y) = \mathbf{0}$$

Define the loss function:

$$\boldsymbol{Loss}_{\boldsymbol{PDE}} = \sum_{i=1}^{N_f} \left(\boldsymbol{net}_{\boldsymbol{F}}(x_i, y_i)\right)^2$$



Explanation Example: Use of BC's

Left & Bottom:

Target values: $net_{\phi}(x, y) = a$

Loss: $Loss_{L\&B} = \sum_{j=1}^{N_{L\&B}} \left(net_{\phi}(x_j, y_j) - a\right)^2$

Right: $\mathbf{net_{BR}(1, y)} = \frac{d\phi}{dx} + 0.5\phi$

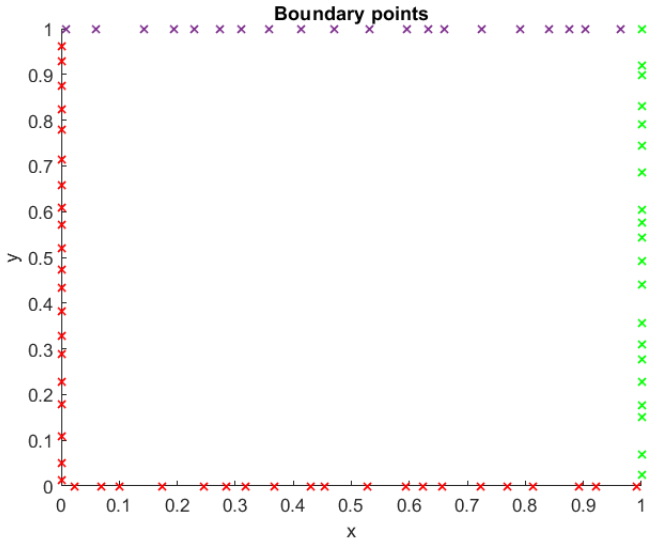
Target values: $\mathbf{net_{BR}(1, y) = 0}$

Loss: $Loss_R = \sum_{l=1}^{N_R} \left(\mathbf{net_{BR}}(1, y_l)\right)^2$

Top: $\mathbf{net_{BT}(1, y)} = \frac{d\phi}{dy}$

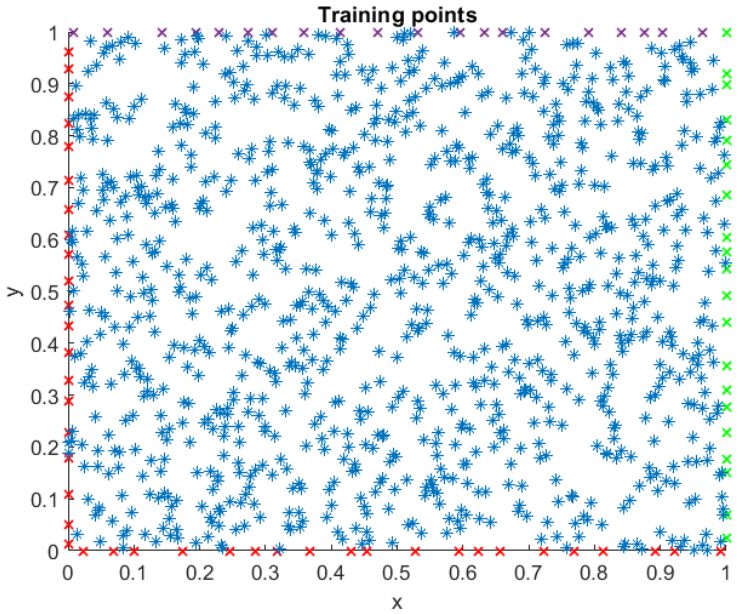
Target values: $\mathbf{net_{BT}(x, 1) = 0}$

Loss: $Loss_T = \sum_{m=1}^{N_T} \left(\mathbf{net_{BT}}(x_m, 1)\right)^2$



$$Loss_{BC's} = Loss_{L\&B} + Loss_R + Loss_T$$

Explanation Example: Put Things Together



$$\text{Loss} = \sum_{i=1}^{N_f}\left(net_F(x_i, y_i)\right)^2 + \sum_{j=1}^{N_{L\&B}}\left(net_\phi(x_j, y_j) - a\right)^2 + \sum_{l=1}^{N_R}\left(\text{net}_{\text{BR}}(1, y_l)\right)^2 + \sum_{m=1}^{N_T}\left(\text{net}_{\text{BT}}(x_m, 1)\right)^2$$

Neutron Balance
(i.e., physics principle)

Boundary Conditions

k-eigenvalue Problems

The 2D *k*-eigenvalue diffusion model:

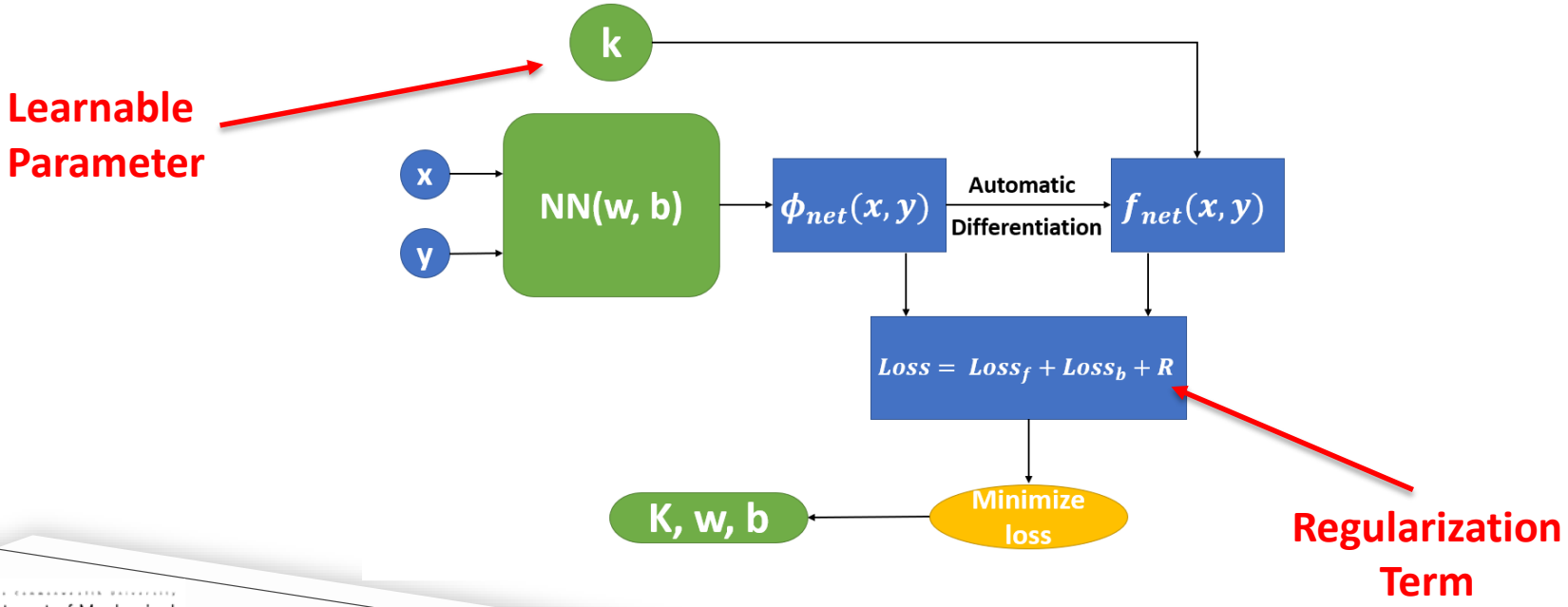
$$F := \left[\frac{\partial}{\partial x}\left(D\frac{\partial \phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(D\frac{\partial \phi}{\partial y}\right)\right] - \Sigma_a(x,y)\phi(x,y) + \frac{1}{k}\nu\Sigma_f(x,y)\phi(x,y) = 0$$

Why this is different?

- Parametric equation (unknown *k*)
- Homogenous (Direct minimization of ***F*** *results in* $\phi(x,y) = 0$)

Customize PINN for *k*-eigen Problems

Add one free learnable parameter in PINN to approximate *k*



Additional Regulation for the Loss Function

$$\text{Loss} = \underbrace{\text{Loss}_f}_{\text{Neutron Balance (i.e., physics principle)}} + \underbrace{\text{Loss}_b}_{\text{Boundary Conditions}} + \underbrace{R}_{\text{Regularization}}$$

Regularization term:

$$R = [\text{Fission rate} - C]^2$$

Here *C* is a user provided parameter, and the fission rate is

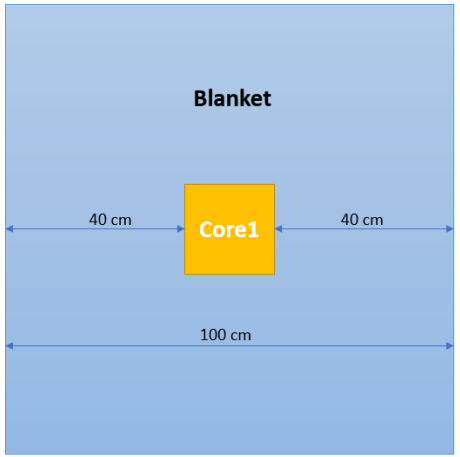
$$\text{Fission rate} \equiv \sum_{i=1}^{N_f} \Sigma_f \phi_{net}(x_i, y_i)$$

One-Group 2D Examples

Material Properties

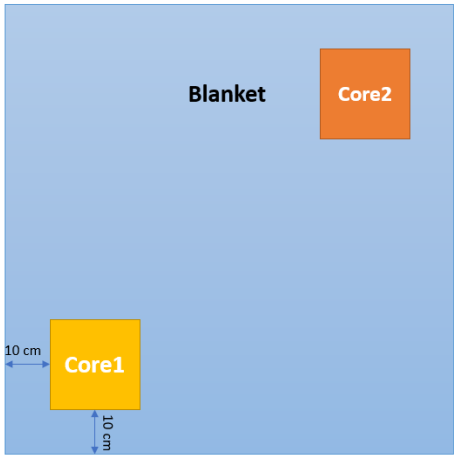
Region Material	Σ_a (cm^{-1})	D (cm)	Σ_f (cm^{-1})
Core 1	0.062158	2.2008	0.107622
Core 2	0.062158	2.2008	0.102622
Blanket	0.064256	2.0950	0.0

Example 1



(a)

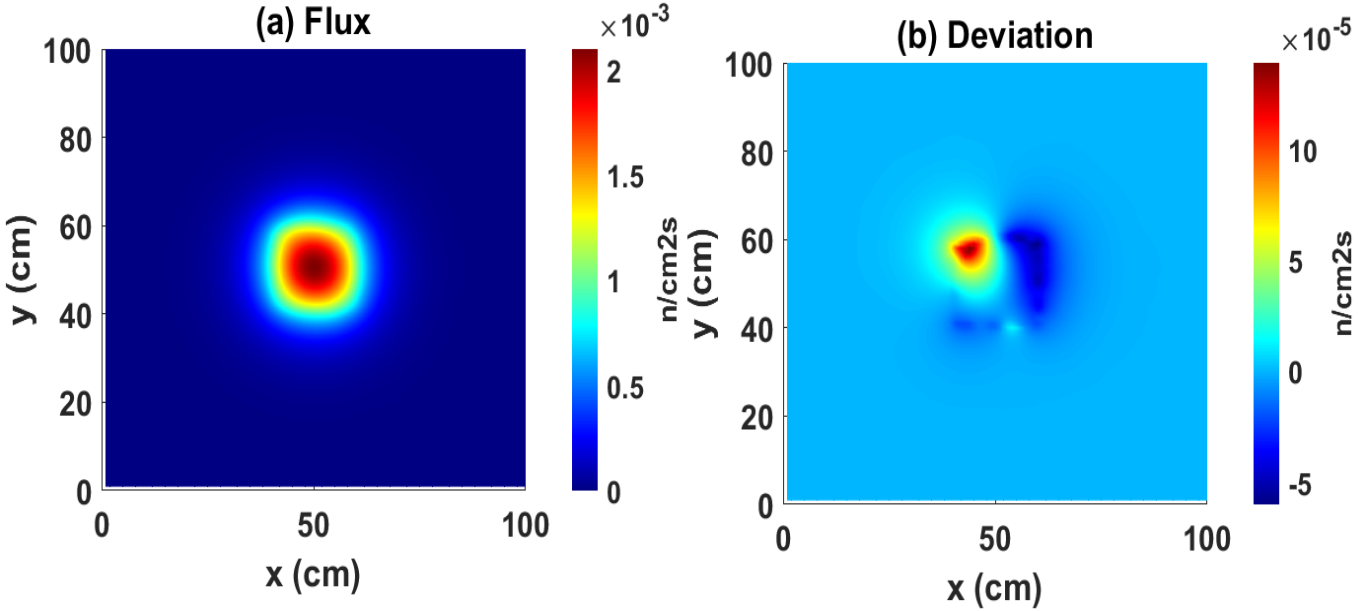
Example 2

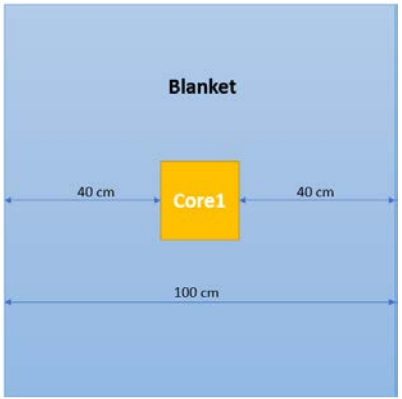


(b)

All zero-incoming flux B.C. assumed

Example 1 Results



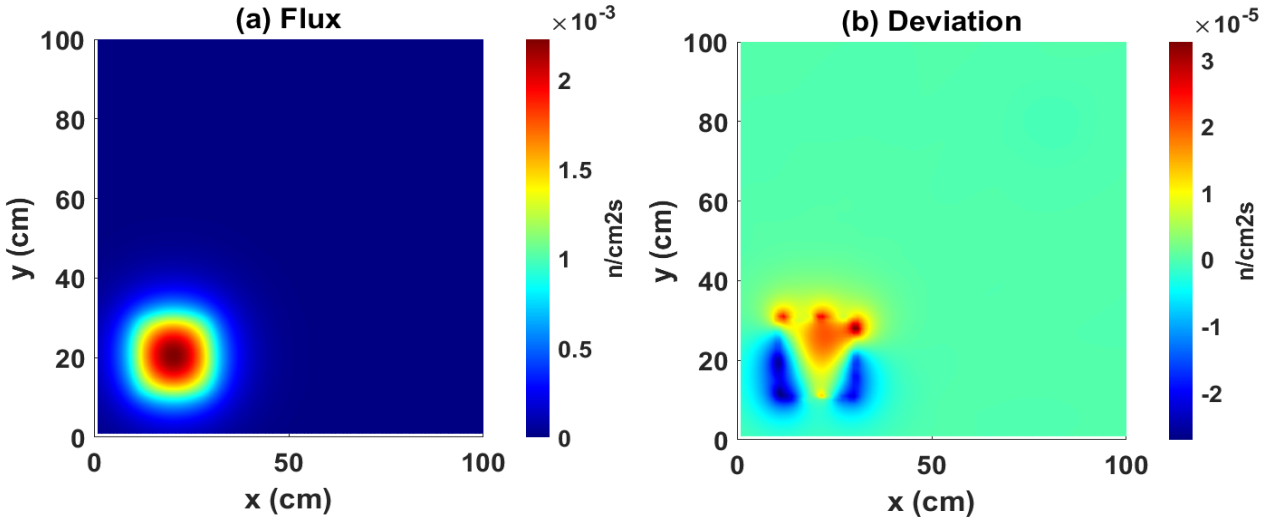


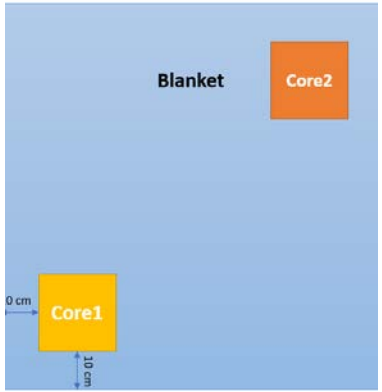
Predicted $k = 0.96266$

Reference $k = 0.96395$

$\%Err = 0.13\%$ (129 pcm)

Example 2 Results





Predicted $k = 0.95894$

Reference $k = 0.96321$

$\%Err = 0.44\%$ **(427 pcm)**

Two-Group Diffusion Model

$$\begin{cases} F_1 := -\nabla(D_1\nabla\phi_1) + \Sigma_{r,1}\phi_1 - \dfrac{1}{k}\boldsymbol{\nu\Sigma_{f,2}\phi_2} = 0 \\ F_2 := -\nabla(D_2\nabla\phi_2) + \Sigma_{a,2}\phi_2 - \Sigma_{s,1\to2}\phi_1 = 0 \end{cases}$$

- **What's the difference?**
  - **Joint learning task**
  - **Generally multi-scale optimization problem**

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

$$\mathbf{Loss} = \sum_{j=1}^{N_b} \left| \begin{matrix} \boldsymbol{F_{T1}}\left(\boldsymbol{x_j^T}, \boldsymbol{y_j^T}\right) \\ \boldsymbol{F_{T2}}(\boldsymbol{x_j^T}\boldsymbol{y_j^T}) \end{matrix} \right|^2 + \sum_{i=1}^{N_f} \left| \begin{matrix} \boldsymbol{F_1}\left(\boldsymbol{x_i^f}, \boldsymbol{y_i^f}\right) \\ \boldsymbol{F_2}\left(\boldsymbol{x_i^f}, \boldsymbol{y_i^f}\right) \end{matrix} \right|^2 + \cdots + \quad \mathbf{R}$$

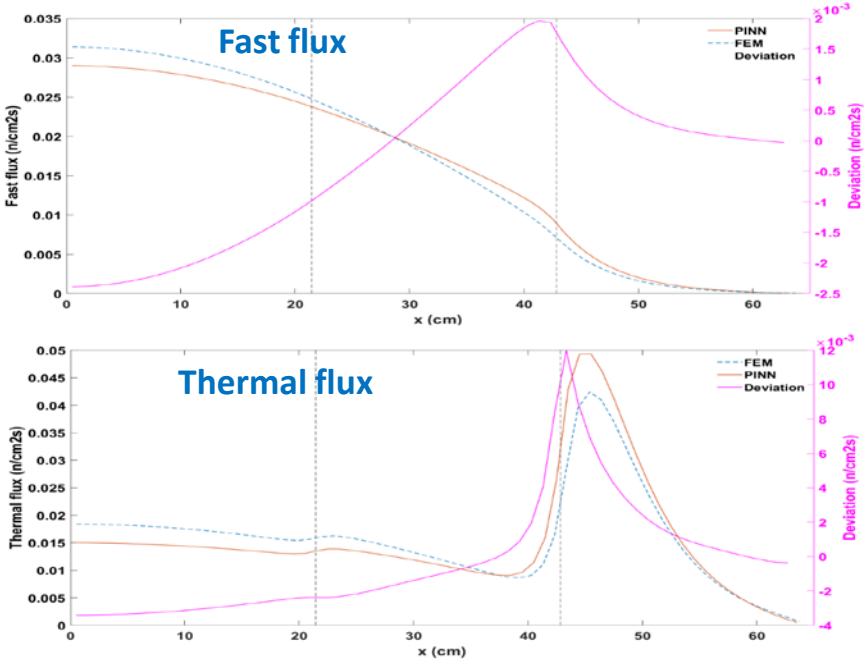
Boundary Conditions | Neutron Balance | Regularization

Two-Group 1D Multi-Region Example

Material Properties

	Material 1	Material 2	Material 3
D_1 [cm]	1.2	1.2	1.2
D_2 [cm]	0.4	0.4	0.2
$\Sigma_{r,1} cm^{-1}]$	0.03	0.03	0.051
$\Sigma_{a,2} cm^{-1}]$	0.3	0.25	0.04
$\Sigma_{s,1\to2} cm^{-1}]$	0.015	0.015	0.05
$\Sigma_{f,1} cm^{-1}]$	0.0075	0.0075	0
$\Sigma_{f,2} cm^{-1}]$	0.45	0.375	0

$\frac{d\phi}{dx}$ I II III $\phi = 0$

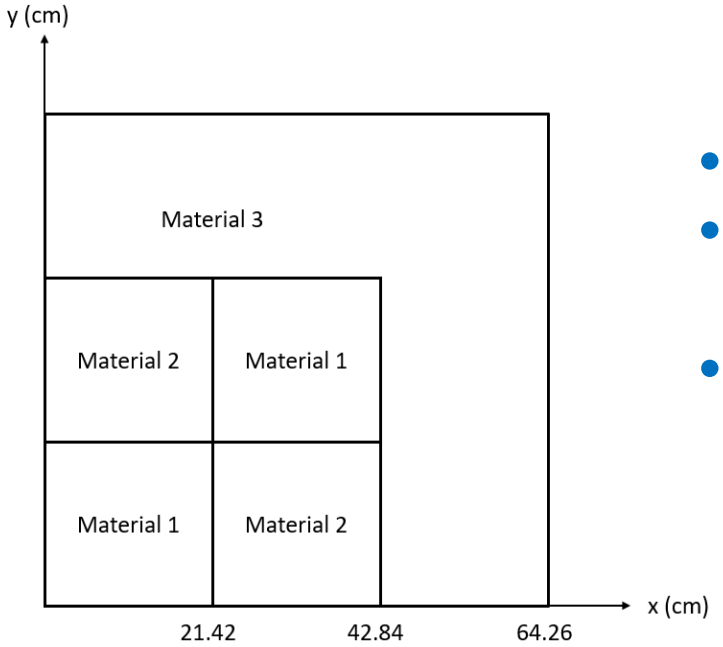


Predicted $k = 0.96764$

Reference $k = 0.96243$

$\%Err = 0.54\%$ **(521 pcm)**

Two-Group 2D Five Region Example



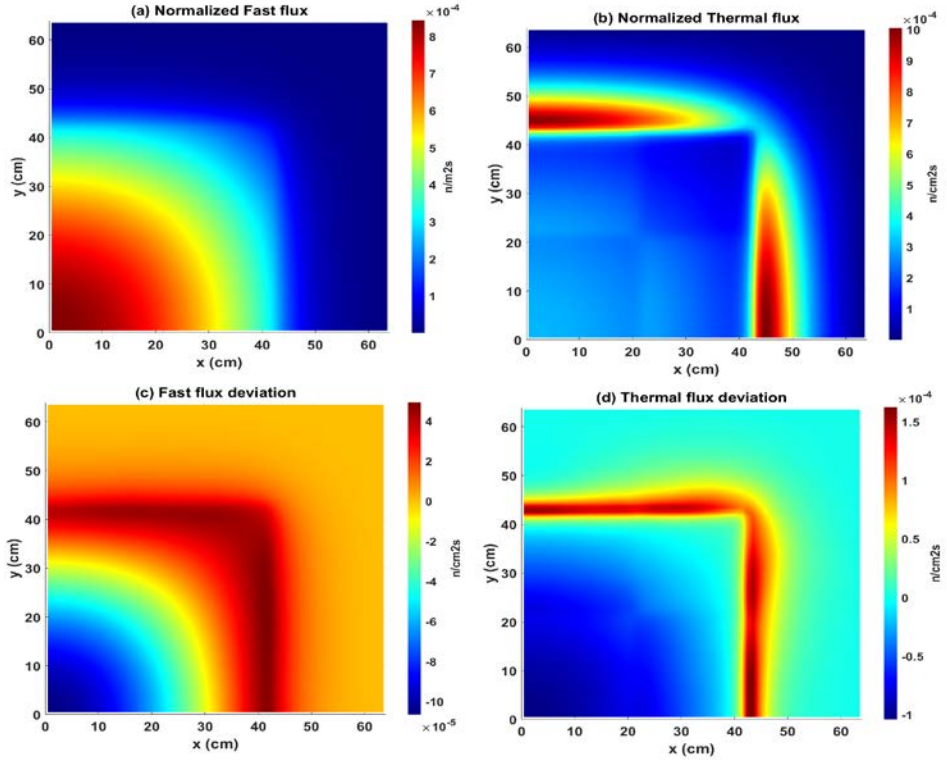
- Reflective BC. (Left and Bottom)
- Zero-flux BC. (Top and Right)

- The same material properties from the 1D example was used

Two-Group 2D Example Results

Fast flux

Thermal flux



Predicted $k = 0.93620$

Reference $k = 0.92764$

$\%Err = 0.92\%$ **(856 pcm)**

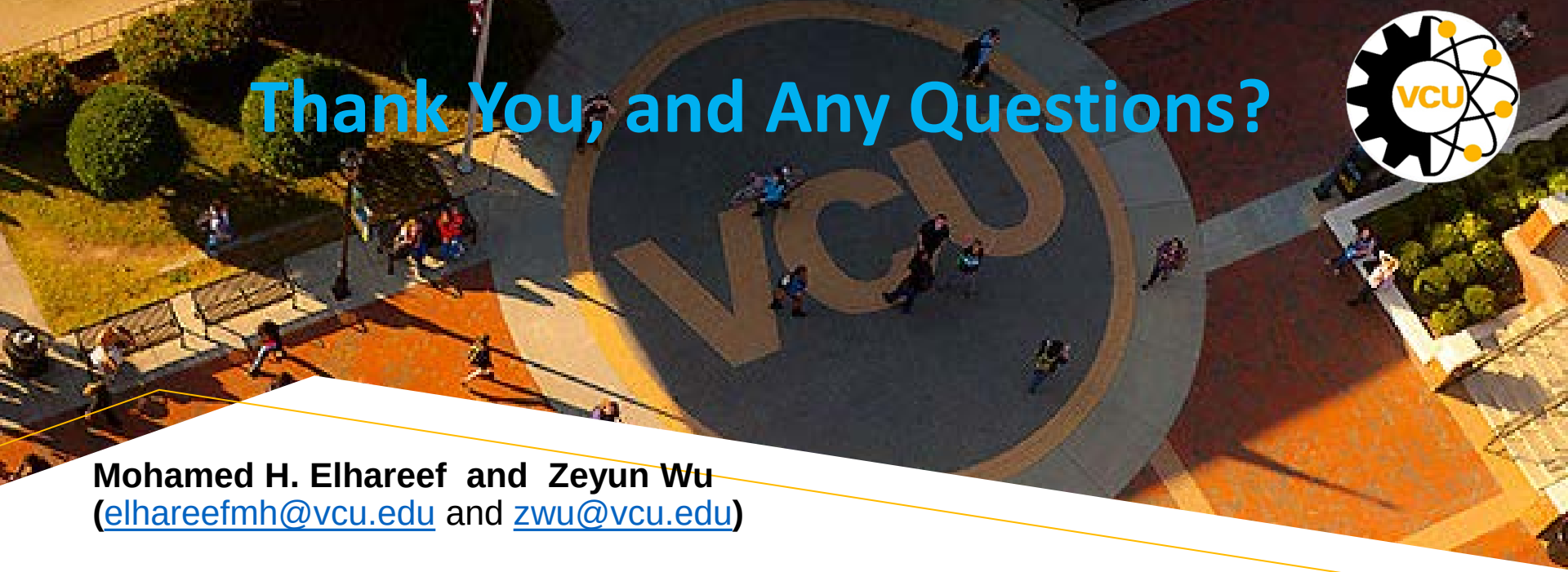
Compared to reference solutions, the max relative difference for fast flux is 8%, for thermal flux is 15% .

Conclusions

- **Advantages:**
  1. Obtain **mesh-free** solutions
  2. No simplifications for material **interfaces** or **B.C.s**
  3. Eliminate the **fission source convergence** problem
  4. Achieve **the same level of accuracy** as conventional methods.
  5. Manpower efforts for the PINN can be **significantly reduced**.
- **Challenges:**
  1. Computational complexity
  2. Applications to higher dimensionality problems
  3. Multi-scale optimization (**Multi-group problems**)

Discussion and Future Work

- Computational accuracy and efficiency
- PINN variations:
  - Physics-Informed Neural Operator (PINO)
  - Parareal Physics-Informed Neural Network (PPINN)
  - Probabilistic PINN
- Future efforts
  - Dominance Ratio
  - Multi-group problems (G>2)



Mohamed H. Elhareef and Zeyun Wu
(elhareefmh@vcu.edu and zwu@vcu.edu**)**

Extension of the PINN Diffusion Model to k-eigenvalue Problems

PHYSOR 2022 Conference, May 15-20, 2022

VCU College of Engineering
Mechanical and Nuclear Engineering