Physics-Informed Deep Learning Neural Network Solution to the Neutron Diffusion Model

Mohamed H. Elhareef[1], Zeyun Wu[1], and Yu Ma[2]

[1]Virginia Commonwealth University, Richmond VA, USA
[2]Sun Yat-sen University, Zhuhai, Guangdong, P.R. China

VCU College of Engineering

M&C 2021 - The International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering
Virtual Talk, October 5th 2021

Outline

- PINN Framework
- Burgers' Equation
- Loosely Coupled Reactor Model (LCRM)
- LCRM Results
- Future Work and Conclusions

Outline

- **PINN Framework**
- Burgers' Equation
- Loosely Coupled Reactor Model (LCRM)
- LCRM Results
- Future Work & Conclusions

Physics-Informed Neural Networks (PINN)

- **Solves two classes of problems:**
  1. **Data-driven solution of PDEs (Forward approach)**
  2. **Data-driven discovery of PDEs (Inverse approach)**
- **Provides framework for integrating observed data with theoretical models.**
- **Successfully applied to various domains:**

  Fluids, Quantum Mechanics, Power Systems, etc.

Ref.: M. Raissi et al., "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, **378**, pp. 686-707 (2019).

Forward PINN Framework

The PINN approach uses the neural networks (NN) model to approximate the solution of PDEs:

- Considering a general non-linear differential operator:

$$F := \mathbb{N}\big(Y(x_1, x_2, \cdots, x_n)\big) = 0$$

- The solution can be approximated to a NN model:

$$Y(x_1, x_2, \cdots, x_n) \cong net\_Y(x_1, x_2, \cdots, x_n)$$

- Automatic differentiation (AD) can be used to differentiate the NN with respect to its input parameters. Thus the PDE model can be constructed as:

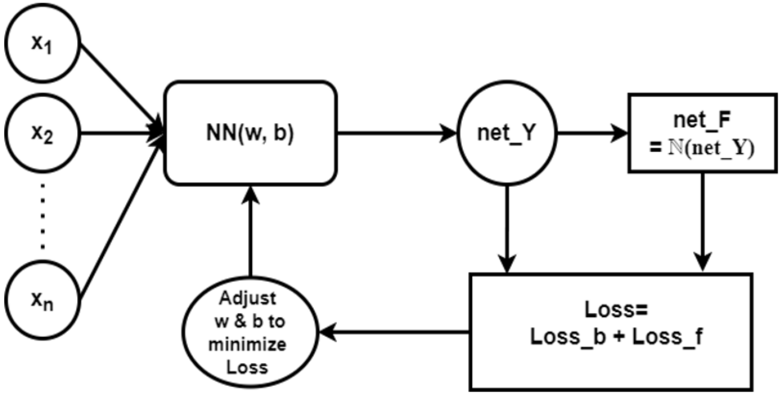
$$net\_F := \mathbb{N}\big(net\_Y(x_1, x_2, \cdots, x_n)\big) \cong 0$$

- The solution can be learnt by satisfying the above equations

Forward PINN Framework (Cont.)

NN Training Procedure:

- Learnable parameters are learnt by minimizing a customized *loss* function.
- Loss function is defined as MSE between:
  1. Predicted values of ***Y*** and the known values (e.g., BCs)
  2. Predicted values of ***F*** and its exact value (0)
- Loss function is evaluated at a set of boundary points N_b and internal points N_f



PINN Training Scheme

$$Loss = \frac{1}{N_b}\sum_{i=1}^{N_b}\left[net\_Y(x_1^i, x_2^i, \ldots x_n^i) - Y(x_1^i, x_2^i, \ldots x_n^i)\right]^2 + \frac{1}{N_f}\sum_{i=1}^{N_f}\left[net\_F(x_1^i, x_2^i, \ldots x_n^i)\right]^2$$

Outline

- PINN Framework
- **Burgers' Equation**
- Loosely Coupled Reactor Model (LCRM)
- LCRM Results
- Future Work & Conclusions

Burgers' Equation

- **Time-dependent one-dimensional (1D) Burgers' equation**

$$F := \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \frac{0.01}{\pi}\frac{\partial^2 u}{\partial x^2} = 0$$

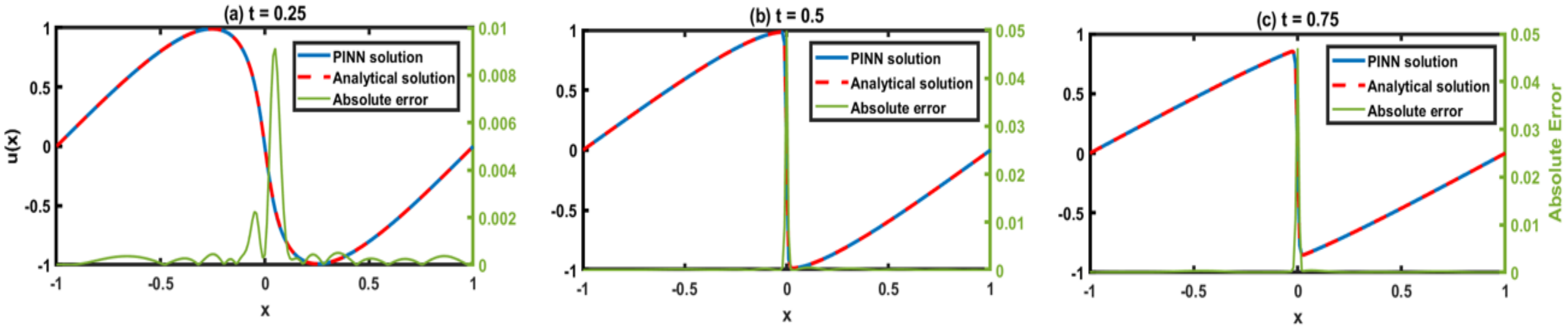
where $x \in [-1, 1]$, $t \in [0, 1]$**, and the boundary and initial conditions are subject to**

$$u(0, x) = -\sin(\pi x)$$
$$u(t, -1) = u(t, 1) = 0$$

PINN Applied to Burgers' Equation

- The same NN structure and optimization algorithm recommended by Raissi was used
- The NN model used in this example has **9** hidden layers and each layer contains **20** neurons
- The hyperbolic tangent sigmoid transfer function is used as the threshold function for each activation connector in the network
- The loss function was minimized using the L-BFGS approach
- Analytic solution was used as reference solution

PINN Solution to Burgers' Equation



A comparison between PINN and analytical $u(t, x)$ at $t = 0.25, 0.5,$ and 0.75, respectively.

Outline

- PINN Framework
- Burgers' Equation
- **Loosely Coupled Reactor Model (LCRM)**
- LCRM Results
- Future Work & Conclusions

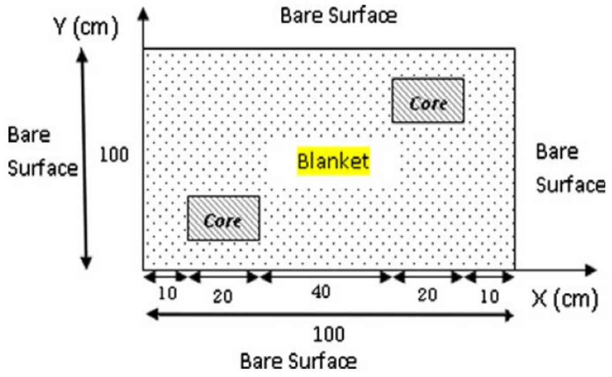
Loosely Coupled Reactor Model (LCRM)

- The model is based on the one-group two-dimensional steady state diffusion equation:

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

- with zero-incoming fluxes are assumed for all boundary surfaces:

At the surface $x = 0$: $$\frac{1}{4}\phi(0,y) - \frac{1}{2}D\frac{d\phi}{dx}\bigg|_{x=0} = 0$$



Geometry of LCRM problem

Region Material	Σ_a (cm^{-1})	Σ_s (cm^{-1})	S (n/cm^3)
Core	0.062158	0.089302	0.01048083
Blanket	0.064256	0.094853	0.00214231

PINN Applied to LCRM

- The solution is approximated to: $\phi(x, y) \approx \boldsymbol{net\_\phi}(\boldsymbol{x}, \boldsymbol{y})$,

  Where $\boldsymbol{net\_\phi}$ is a NN model with a set of learnable parameters ***(w, b).***

- $\boldsymbol{net\_\phi}$ is differentiated according to the PDE model to construct $\boldsymbol{net\_F}$
- Boundary conditions (set of 4 ODEs) are satisfied by differentiating $\boldsymbol{net\_\phi}$ *according to each ODE.*
- *Learnable parameters* **(w, b)** *are learnt by minimizing loss function:*

$$Loss = \frac{1}{N_f}\sum_{i=1}^{N_f}[net\_F(x_i, y_i)]^2 + \frac{1}{N_b}\sum_{i=1}^{N_b}\left[\frac{1}{4}net\_\phi(0, y_i) - \frac{1}{2}D\frac{\partial net\_\phi(0, y_i)}{\partial x}\right]^2 + \cdots$$

More Implementation Details

- LHS strategy was used to generate training points
- The sigmoid transfer function was used for each activation connector in the network
- Adam optimizer was used to minimize the loss function
- A high-order FEM solution was used as a reference solution

PINN Optimization

Mean relative error (%) between PINN prediction and the reference solution for different NN architectures with fixed N_f and N_b (**N_f = 10000** and **N_b = 25**).

Layers \ Neurons	10	20	40
2	25.04	11.04	47.69
4	11.24	5.15	1.56
6	2.15	0.79	0.81
8	1.2	0.96	0.73

Mean relative error (%) between PINN prediction and the reference solution for N_f and N_b with fixed NN architecture (**8 hidden layers** and **40 neurons per layer**).

N_b \ N_f	2000	5000	10000
25	1.06	0.72	0.73
50	0.95	1.04	0.72
100	1.39	0.82	0.69
300	1.13	0.76	0.84
1000	0.91	0.74	0.69

PINN Optimization (Cont.)

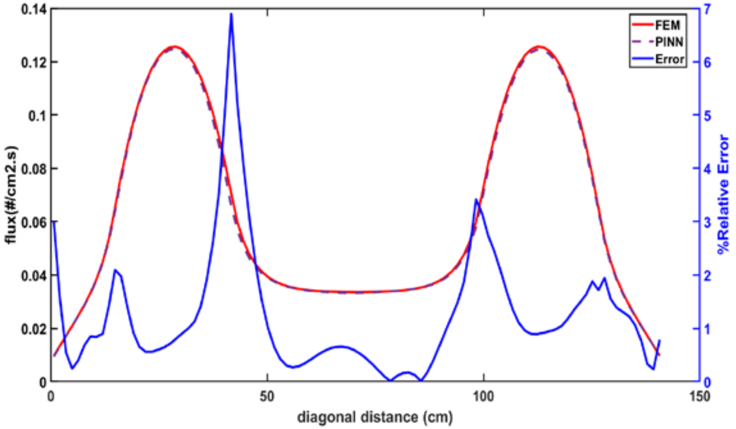
The optimum hyperparameters are shown below with mean percentage relative error of **0.69%** and maximum error of **6.9%** in flux solution

#hidden layers	#neurons/layer	N_f	N_b/surface
8	40	10,000	100

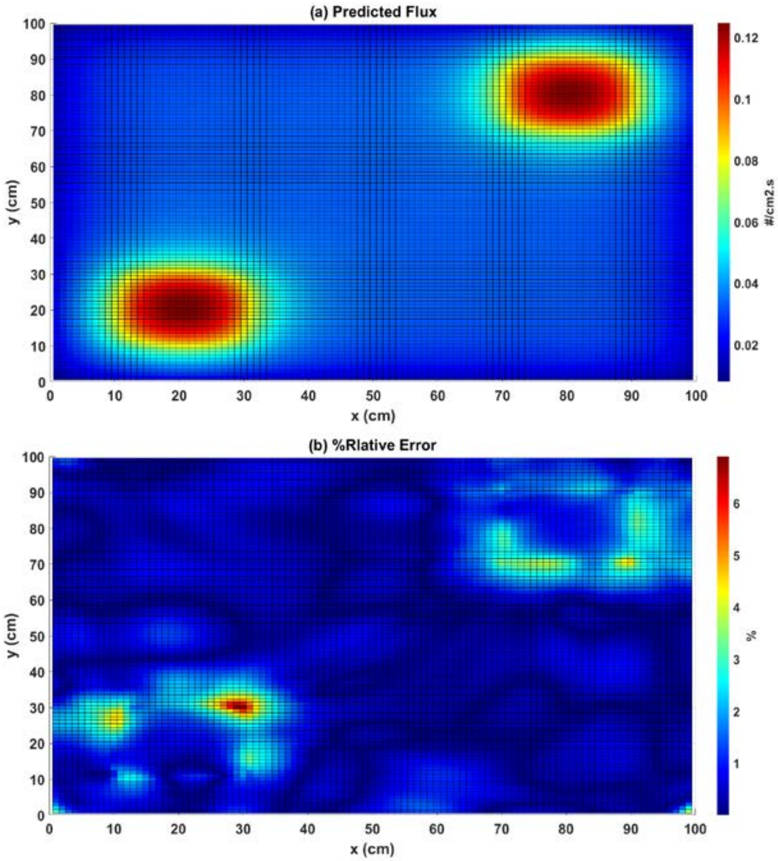
Outline

- PINN Framework
- Burgers' Equation
- Loosely Coupled Reactor Model (LCRM)
- **LCRM Results**
- Future Work & Conclusions

LCRM Results

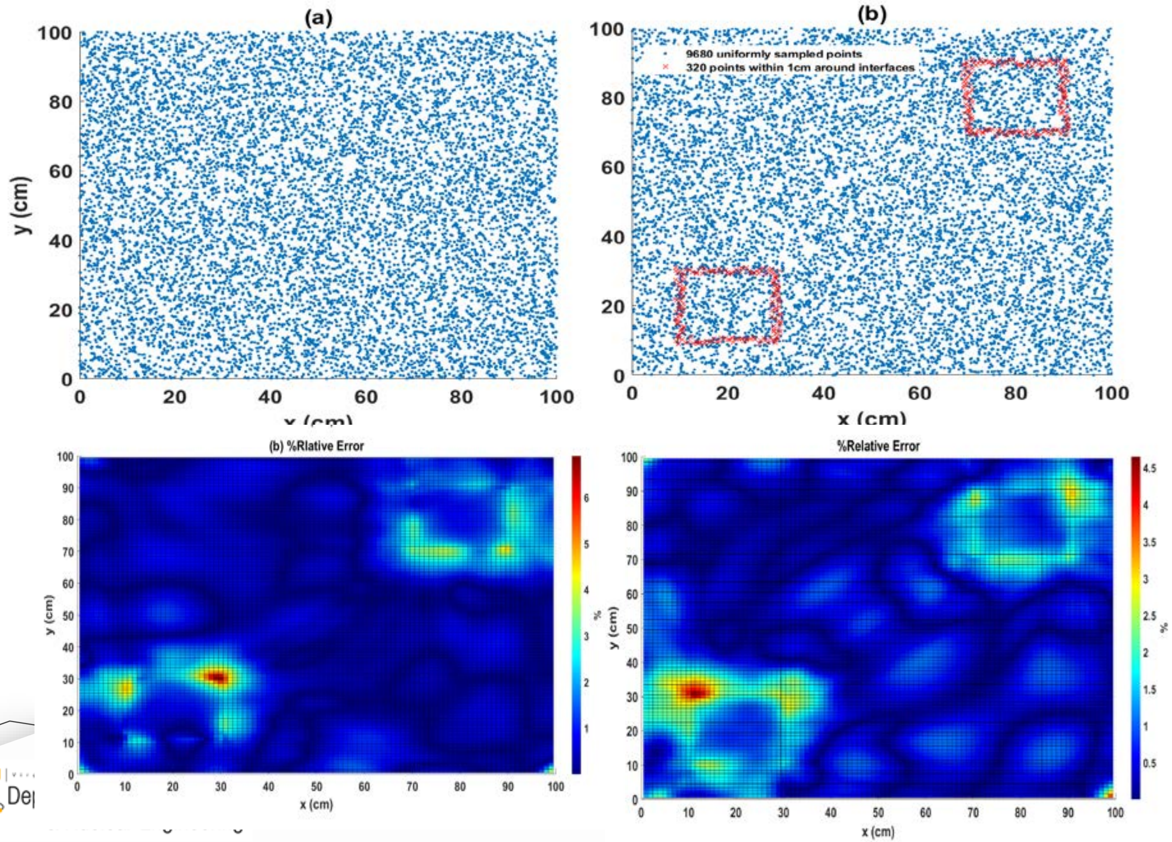


Predicted flux and percentage relative error along the diagonal line of the solution domain.



PINN predicted flux distribution (a) and relative percentage error distribution compared to the FEM solution (b).

Non-uniform training points



Relative percentage error	Uniform data	Non-uniform data
mean	0.69%	0.63%
Std.	0.74	0.59
max	6.9%	4.6%

Outline

- PINN Framework
- Burgers' Equation
- Loosely Coupled Reactor Model (LCRM)
- LCRM Results
- **Future Work & Conclusions**

k-eigenvalue Problems

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

- **Why it's different?**
  - **Parametric equation (unknown *k*)**
  - **Homogenous (Direct minimization of *f* *results in* $\phi(x,y) = 0$**
- ***Proposed approach:***
  - **Additional learnable parameter (approximate *k*)**
  - **Regularization term in the loss function (enforces a pre-defined value for the flux integration)**

Conclusions

- **Advantages:**
  1. Obtain mesh-free solutions
  2. No large amount of training data needed ahead
  3. Achieve the same level of accuracy as conventional methods.
  4. Manpower efforts for the PINN can be significantly reduced.
  5. Easily applied to complex geometries and versatile BCs.
  6. Prior knowledge of high-gradient regions can enhance accuracy
- **Challenges:**
  1. Computational complexity
  2. Applications to higher dimensionality problems.

Thank You!