TPINN: An improved architecture for distributed physics informed neural networks

Sreehari M, Balaji Srinivasan

Indian Institute of Technology Madras, India

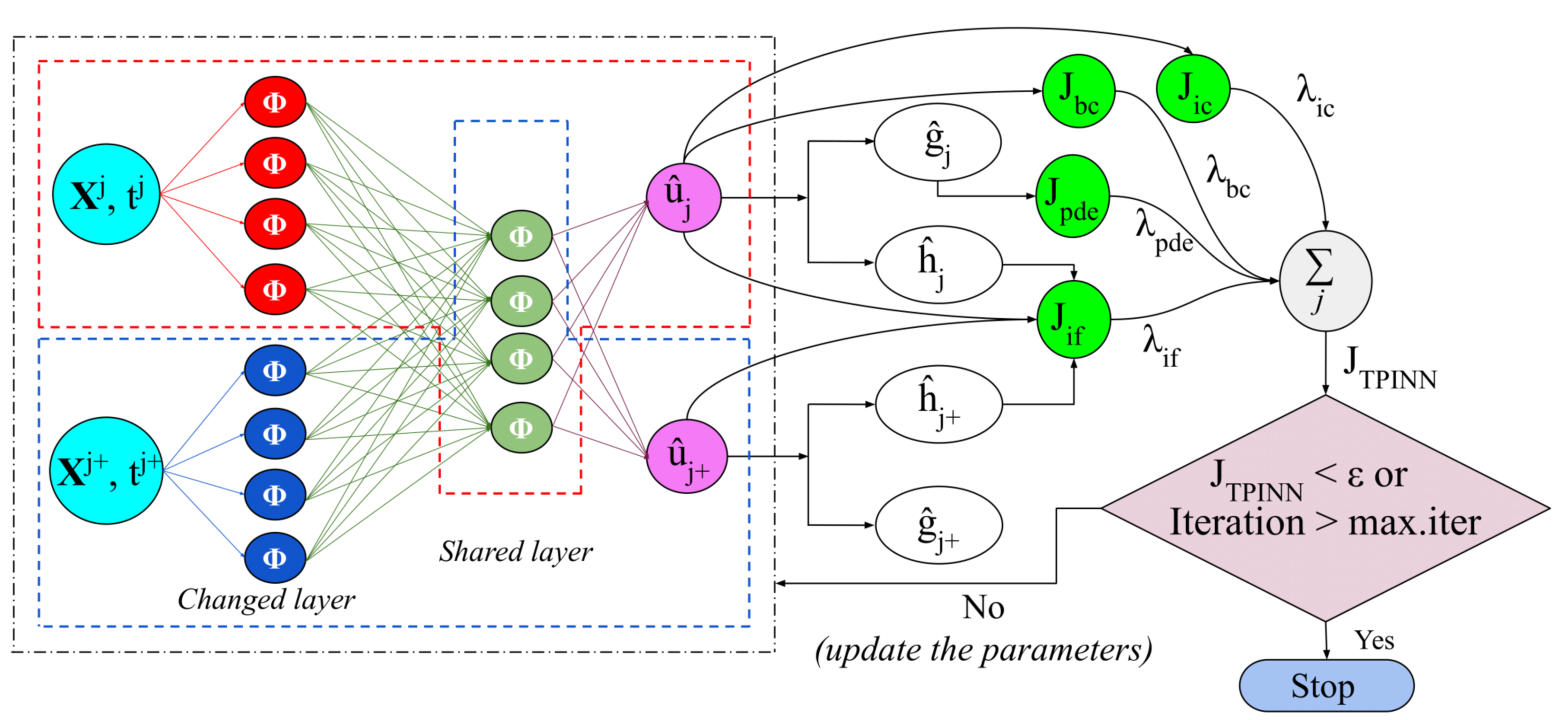
Introduction

- Approximate solutions to PDEs using ANNs is gaining attention because of efficient methods like Physics Informed Neural Network (PINN) [1].
- PINNs, unlike typical numerical methods, use neural networks as a basis for the solution.
- PINNs are elegant but still lag behind traditional methods for some forward and inverse problems.

Problem Setup

- PINN cannot be employed in its native form when the PDE changes its form or when there is a discontinuity in the parameters of PDE across different non-overlapping sub-domains.
- Employing separate PINNs for each subdomain with interface conditions embedded into the loss function is a possible solution (DPINN) [2]. However, DPINN demands high computational power and memory usage.
- We propose Transfer Physics Informed Neural Network (TPINN), an improved architecture for DPINN that demands less computational power and memory requirements without compromising prediction accuracy.

TPINN Architecture



TPINN with first hidden layer change

- In TPINN, one or more layers of PINN are changed across different sub-domains keeping the remaining layer(s) same.
- *Problem specific interface constraints* are embedded in the loss function.
- Sharing of layers results in a reduction of the total no. of parameters without affecting the representation capability.

Loss function

- $J_{TPINN} = \lambda_{pde} J_{pde} + \lambda_{bc} J_{bc} + \lambda_{ic} J_{ic} + \lambda_{if} J_{if} + \lambda_{inv} J_{inv}$

- $J_{pde} = \sum_{j=1}^{N_{SD}} \left(\frac{1}{N_{pde}^{j}} \sum_{i=1}^{N_{pde}^{j}} \left| \hat{g}_j \left(\boldsymbol{x_{pde}^{i,j}}, t_{pde}^{i,j} \right) \right|^2 \right)$

- $J_{bc} = \sum_{j=1}^{N_{SD}} \left(\frac{1}{N_{bc}^{j}} \sum_{i=1}^{N_{bc}^{j}} \left| \hat{u}_j \left(\boldsymbol{x_{bc}^{i,j}}, t_{bc}^{i,j} \right) - u_j \left(\boldsymbol{x_{bc}^{i,j}}, t_{bc}^{i,j} \right) \right|^2 \right)$

- $J_{ic} = \sum_{j=1}^{N_{SD}} \left(\frac{1}{N_{ic}^{j}} \sum_{i=1}^{N_{ic}^{j}} \left| \hat{u}_j \left(\boldsymbol{x_{ic}^{i,j}}, t_{ic}^{i,j} \right) - u_j \left(\boldsymbol{x_{ic}^{i,j}}, t_{ic}^{i,j} \right) \right|^2 \right)$

- $J_{if} = \sum_{k=1}^{N_I} \left(\frac{1}{N_{if}^{k}} \sum_{i=1}^{N_{if}^{k}} \left| \hat{u}_{k+} \left(\boldsymbol{x_{if}^{i,k}}, t_{if}^{i,k} \right) - \hat{u}_{k-} \left(\boldsymbol{x_{if}^{i,k}}, t_{if}^{i,k} \right) \right|^2 \right) +$

  $\sum_{k=1}^{N_I} \left(\frac{1}{N_{if}^{k}} \sum_{i=1}^{N_{if}^{k}} \left| \hat{h}_{k+} - \hat{h}_{k-} \right|^2 \right)$

Example: for 1D heat conduction, $h = K \frac{du}{dx}$, K *is thermal conductivity.*

- $J_{inv} = \sum_{j=1}^{N_{SD}} \left(\frac{1}{N_{inv}^{j}} \sum_{i=1}^{N_{inv}^{j}} \left| \mathcal{L}_j \left(\hat{u}_j, \boldsymbol{x_{inv}^{i,j}}, t_{inv}^{i,j} \right) \right|^2 \right)$

Algorithm 1 TPINN algorithm

Step 1: Identify the non-overlapping sub-domains.

Step 2: Decide the architecture of TPINN by fixing the no. of layers (N_L) and no. of neurons in each layer (N_q) for each sub-domain.

Step 3: Decide the set p which contains the layer(s) of the TPINN to be changed across each sub-domain.

Step 4: Construct TPINN for jth sub-domain using Θ_{shared} and Θ_j, $j = 1,2,...,N_{SD}$.

Step 5: Prepare random scattered training data points from all sub-domains.

Step 6: Prepare random scattered training data points from all the interfaces.

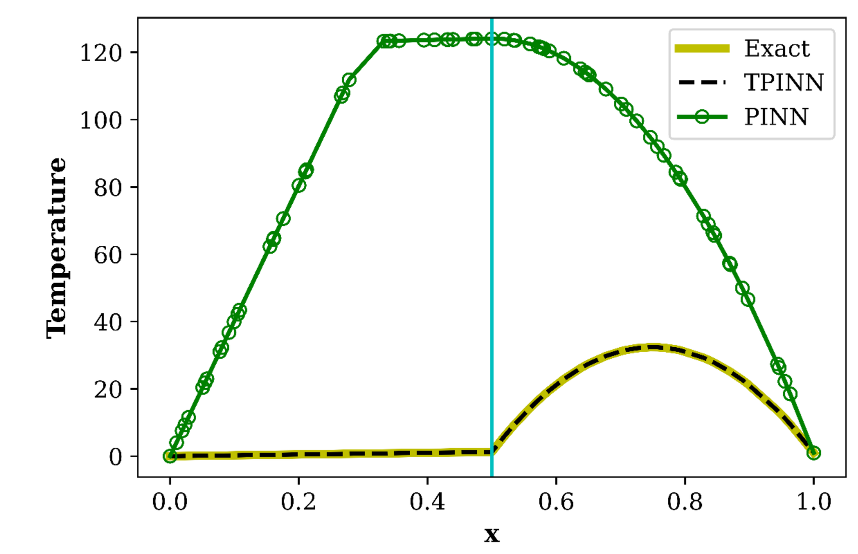
Step 7: Choose proper values for penalty factors.

Step 8: Initialize the parameters of the neural networks constructed in **Step 4** using suitable initialization method. Train these neural networks for the best parameters using the training data prepared in **Step 5** & **Step 6** by minimizing the loss J_{TPINN} using a suitable optimization method. $\Theta^* = \arg\min_{\Theta}(J_{TPINN})$

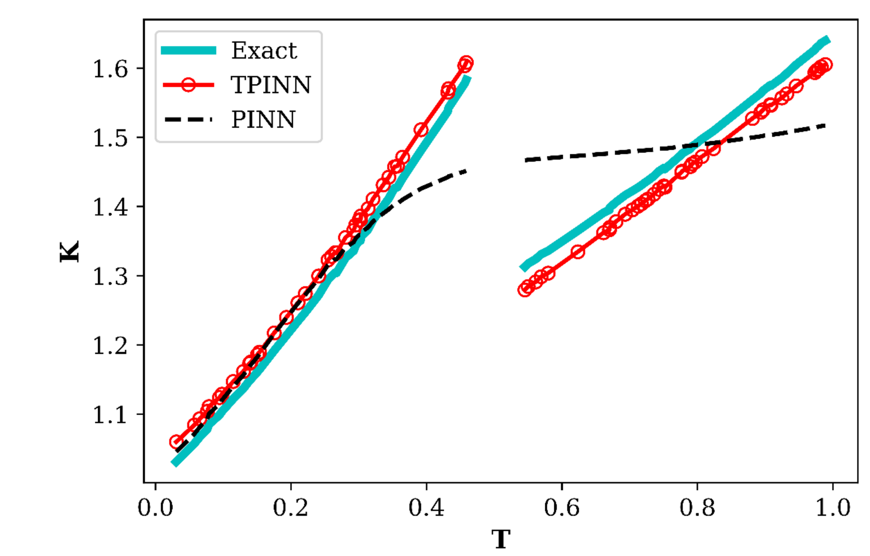
Experiments

- We solve forward and inverse problems of heterogeneous heat conduction. For the inverse problem, the task is to estimate thermal conductivity using noisy temperature and heat flux values.
- Comparison is made with PINN and DPINN for a given depth and total no. of neural network parameters.

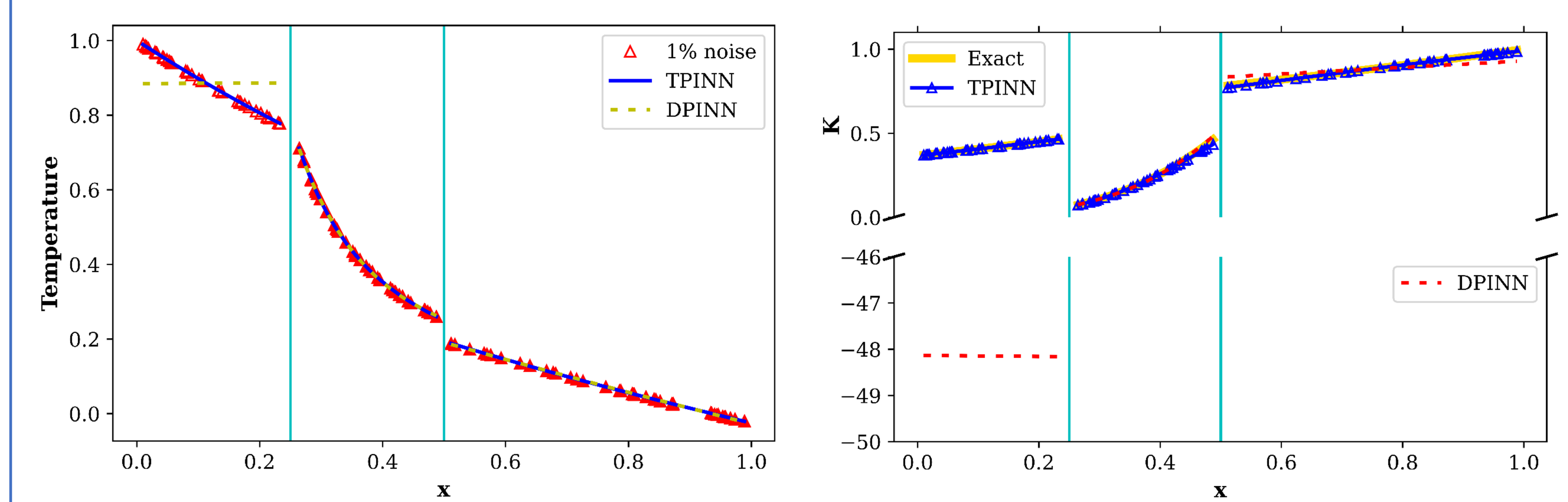
Results



1d steady state heterogeneous heat conduction with heat generation



Inverse problem: 1D steady state heterogeneous heat conduction with temperature dependant thermal conductivity



Inverse problem: 1D steady state heterogeneous heat conduction with spatially varying thermal conductivity

Conclusion

- In this work, we propose a novel architecture for distributed physics informed neural networks.
- Our method performs very well, especially in heterogeneous domains.
- TPINN solves inverse problems in the heterogeneous domain efficiently by fitting a surrogate model compared to the conventional methods that involve the execution of forward and inverse problems iteratively until convergence.

References

1. M. Raissi, P. Perdikaris, G.E. Karniadakis (2018) Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. Journal of Computational Physics 378:686-707.
2. V. Dwivedi, N. Parashar, B. Srinivasan (2020) Distributed learning machines for solving forward and inverse problems in partial differential equations. Neurocomputing. https://doi.org/ 10.1016/j.neucom.2020.09.006