LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-JRNL-868933

Adaptive Interface-PINNs (AdaI-PINNs) for inverse problems: Determining material properties for heterogeneous systems

D. R. Sarkar, C. Annavarapu, P. Roy

September 3, 2024

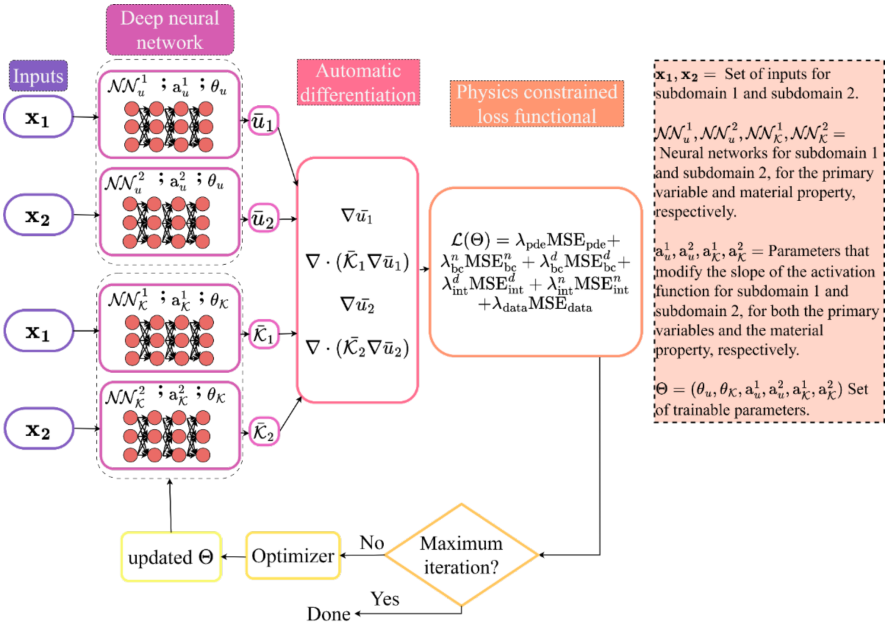
Finite Elements in Analysis and Design

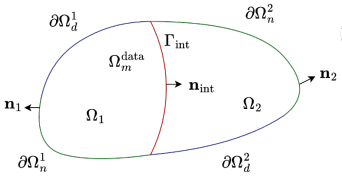
Disclaimer

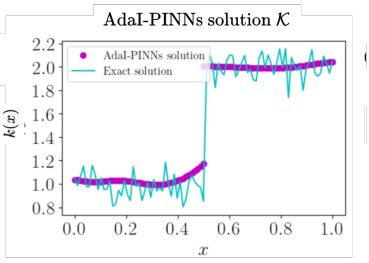
Graphical Abstract

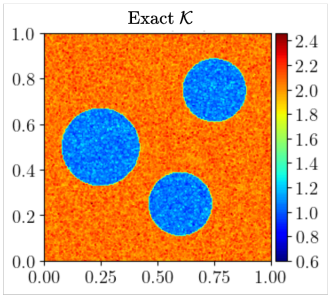
Adaptive Interface-PINNs (AdaI-PINNs) for Inverse Problems: Determining Material Properties for Heterogeneous Systems

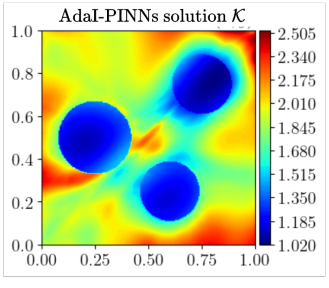
Dibakar Roy Sarkar, Chandrasekhar Annavarapu, Pratanu Roy











Highlights

Adaptive Interface-PINNs (AdaI-PINNs) for Inverse Problems: Determining Material Properties for Heterogeneous Systems

Dibakar Roy Sarkar, Chandrasekhar Annavarapu, Pratanu Roy

- Discontinuous material property determination.
- Adaptive Interface-PINNs framework to solve inverse problem.
- Optimal hyperparameters were obtained with Bayesian optimization using Weights & Biases (WandB).

Adaptive Interface-PINNs (AdaI-PINNs) for Inverse Problems: Determining Material Properties for Heterogeneous Systems

Dibakar Roy Sarkar[a], Chandrasekhar Annavarapu[a], Pratanu Roy[b]

[a]*Indian Institute of Technology Madras, Department of Civil Engineering, Chennai, 600036, Tamil Nadu, India*
[b]*Lawrence Livermore National Laboratory, Atmospheric, Earth and Energy Division, Livermore, 94551, California, United States*

Abstract

We determine spatially varying discontinuous material properties using a domain-decomposition based physics-informed neural networks (PINNs) framework named the Adaptive Interface–PINNs or AdaI–PINNs (Roy et al. arXiv preprint arXiv:2406.04626, 2024). We propose the use of distinct neural networks for the field variables and material properties within each material, utilizing adaptive activation functions. While the neural networks across different materials share the same weights and biases, their activation functions are uniquely tailored using a hyperparameter that influences the slope of the activation function. The proposed framework is tested on several one-dimensional and two-dimensional benchmark examples, and its performance is compared with conventional PINNs and existing domain-decomposition PINNs frameworks, namely, the Multi-domain physics-informed neural network (M-PINN), and the eXtended physics-informed neural networks (XPINNs). The results demonstrate that the proposed approach can determine randomly distributed discontinuous material properties with an L_2 error of $\mathcal{O}(10^{-3})$ for the material property and the root-mean-square error of $\mathcal{O}(10^{-3})$ for the primary variable while the other approaches yield errors that are approximately two orders of magnitude larger (that is, $\mathcal{O}(10^{-1})$). Moreover, the spatial distribution of material properties obtained using the proposed framework is in close agreement with the true distribution, whereas the other approaches fare much worse. Additionally, the proposed approach is approximately 40% faster than its competitors, indicating its potential as a robust alternative for solving inverse problems in heterogeneous materials.

Keywords: Physics-informed neural networks, Inverse problems, Discontinuous material properties, Multi-material diffusion, Diffusion in heterogeneous media, Bayesian search.

1. Introduction

The determination of material properties from observable data is a challenging problem with broad applications in science and engineering. Broadly speaking, these problems are classified as inverse problems, where the goal is to infer underlying parameters from observable data. Indeed, these problems are some of the most challenging problems in computational physics and have broad applications in several fields, including materials science [1], civil engineering [2], biomedical engineering [3] and clinical diagnostics [4, 5].

Inverse problems are often harder to solve than forward problems because they are usually ill-posed. This means we cannot always be sure if a solution exists, or even if it exists whether the solution is unique or not, or how sensitive the solution is to small changes in the data [6]. Traditional approaches to solve inverse problems include optimization-based techniques [7], Bayesian inference [8], and sampling methods [9]. While effective in certain domains, these approaches often face challenges such as high computational costs, the need for extensive domain expertise. These limitations have motivated the search for more efficient and robust alternatives.

Physics-informed neural networks (PINNs), introduced by Lagaris et al. [10] and reviewed by Raissi et al. [11], represent a promising advancement in computational modeling of inverse problems. The authors demonstrated PINNs' effectiveness in solving inverse problems, particularly in determining constant coefficients of partial differential equations (PDEs). A potential advantage of PINNs over traditional methods is their computational efficiency; unlike conventional approaches that require solving a forward model at each iteration, PINNs can tackle inverse problems more directly within the optimization framework, potentially reducing computational costs.

Since their inception, PINNs have been applied to solve inverse problems in several domains. Lu et al. [12] introduced hPINNs to address topology optimization challenges, showcasing their ability to produce smoother designs for problems with non-unique solutions. Cai et al. [13] applied PINNs to solve convection heat transfer problems with unknown thermal boundary conditions. The same research also explored the use of PINNs in two-phase

Stefan problems, successfully identifying the phase transition interface in a one-dimensional domain. Shukla et al. [14] utilized PINNs to compute wave speed in materials using ultrasonic wave-field data, indirectly representing the effect of heterogeneities and material properties in terms of bulk modulus and density. Furthermore, Rasht-Behesht et al. [15] proposed a novel PINN-based approach for solving wave propagation problems and full waveform inversions (FWIs), leveraging recent advances in deep learning methodologies.

For problems involving interfaces that result in discontinuous field variables and material properties, researchers have turned to eXtended PINNs (XPINNs), a generalized form of PINNs introduced by Jagtap et al. [16]. This advanced framework has found applications in various complex scenarios. For instance, Papadopoulos et al. [17] employed XPINNs to determine the Kapitza thermal resistance at interfaces between different phases in multiphase composite materials. Their study highlighted the framework's high accuracy, ease of implementation, and robustness. In a different application, Jagtap et al. [18] utilized XPINNs to address inverse shock wave problems. Their approach incorporated density gradient information, a limited dataset of primitive variables, and global constraints to predict density, velocity, and pressure fields. Building upon this work, Zhang et al. [19] introduced Multi-domain PINNs (M-PINN), which employ a modified loss function to capture discontinuities in field variables and fluxes. Their study showcased M-PINN's capability in solving inverse problems with partially known boundary conditions.

Recently, Sarma et al. [20] introduced Interface PINNs (I-PINNs), a novel PINNs architecture to model elliptic interface problems where the solutions may exhibit strong or weak discontinuities. They demonstrated that I-PINNs is more effective in solving elliptical interface problems than XPINNs and M-PINN. Subsequently, I-PINNs have been used to solve elliptic problems in variational form [21] and poroelastic problems in heterogeneous media [22]. Roy et al. [23, 24] further improved on this architecture by using adaptive activation functions and termed their approach as Adaptive Interface PINNs (AdaI-PINNs). They demonstrated that AdaI-PINNs can outperform the original I-PINNs framework in terms of computational efficiency and accuracy.

Although the I-PINNs and AdaI-PINNs framework have demonstrated significant promise in solving forward elliptic interface problems, there are no studies that investigate their efficacy for inverse problems. In this study, we extend the application of AdaI-PINNs, initially designed for forward prob-

lems, to solve the inverse problem of determining spatially varying material properties from a given data set. This is achieved by modifying the underlying neural network architecture such that a composite neural network framework is used in conjunction with AdaI-PINNs. The composite neural network uses separate neural networks for field variables and material properties within each material. Within each material, these neural networks share the same activation functions, while they are independently trained on all other parameters. Furthermore, the networks across different materials feature distinct activation functions but remain identical in all other parameters as prescribed by the AdaI-PINNs framework. The adaptive activation function is critical, as it evolves to find the optimal set of activation functions during the training process. The proposed framework is tested on one-dimensional and two-dimensional benchmark examples featuring inclusions of various shapes.

The rest of the paper is organized as follows. Section 2 outlines the model problem under consideration, which involves multiple interfaces that divide the domain into distinct subdomains. Section 3 details the AdaI-PINNs framework for addressing the inverse problem, along with the use of the WandB tool for hyperparameter search. Section 4 compares the proposed framework with existing alternatives including M-PINN, XPINNs, and conventional PINNs on several one-dimensional and two-dimensional examples. Finally, Section 6 summarizes our study and highlights potential future research directions.

2. Model Problem

We consider Poisson's equation within a domain Ω, which is divided into two non-overlapping subdomains, Ω_1, and Ω_2 separated by a fixed material interface Γ_{int}. The external boundary, $\partial\Omega$, is partitioned into two disjoint sets denoted by $\partial\Omega_1$, and $\partial\Omega_2$, where their union constitutes the entire external boundary, i.e., $\partial\Omega = \overline{\partial\Omega_1 \cup \partial\Omega_2}$, and their intersections results in a null set, i.e., $\partial\Omega_1 \cap \partial\Omega_2 = \phi$. The interfaces are characterized by the normal vector $\mathbf{n}_{\text{int}}$ that points outward from the interface Γ_{int}, and the external boundaries $\partial\Omega_1$ and $\partial\Omega_2$ are associated with normal vectors $\mathbf{n}_1$ and $\mathbf{n_2}$, directed outward from each subdomain. Figure 1 illustrates a schematic of the problem domain.

The strong form of the boundary value problem in each subdomain Ω_1 and Ω_2 is defined as follows. Given the scalar fields f_n that may be discontinuous across an interface, boundary data $(\Lambda_n^d, \Lambda_n^n)$ and primary variable data Λ_n^{data},

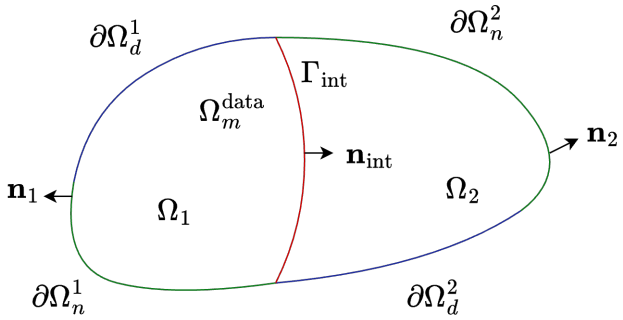


Figure 1: Schematic of the problem domain with two sub-domains Ω_1 and Ω_2 separated by an interface Γ_{int}. The vectors $\mathbf{n}_1$, $\mathbf{n}_2$, and $\mathbf{n}_{\text{int}}$ represent the normal vectors to the boundaries of Ω_1, Ω_2, and the interface Γ_{int}, respectively. The region Ω_m^{data} shows the interior domain from which data are collected. The blue boundaries $\partial\Omega_d^m$ and red boundaries $\partial\Omega_n^m$ denote regions where Dirichlet and Neumann boundary conditions are applied, respectively.

find $\mathcal{K}_n$ such that for $n = 1$ and $n = 2$

$$\begin{aligned} \nabla \cdot (\mathcal{K}_n \nabla u_n) &= f_n \quad \text{in } \Omega_n, \\ u_n &= \Lambda_n^{\text{data}} \quad \text{in } \Omega_n^{\text{data}}, \\ u_n &= \Lambda_n^d \quad \text{on } \partial\Omega_n^d, \\ \mathcal{K}_n \nabla u_n \cdot \mathbf{n}_n &= \Lambda_n^n \quad \text{on } \partial\Omega_n^n, , \\ [\![u]\!] &= 0 \quad \text{on } \Gamma_{\text{int}}, \\ [\![\mathcal{K}\nabla u]\!] \cdot \mathbf{n}_{\text{int}} &= 0 \quad \text{on } \Gamma_{\text{int}}, \end{aligned} \tag{1}$$

where ∇ denotes the gradient operator. The disjoint subsets $\partial\Omega_n^d$ and $\partial\Omega_n^n$ are defined such that $\partial\Omega_n = \overline{\partial\Omega_n^d \cup \partial\Omega_n^n}$, representing the boundaries where Dirichlet and Neumann conditions are applied, respectively. $\Omega_n^{\text{data}} \subset \Omega_n$ refers to the region where the primary variable u_n is known. At the interfaces, $[\![\odot]\!]$ represents the jump in the quantity $\odot$. It is assumed that both the primary variable and the flux are continuous across the interface, with no jumps. The unknown parameter $\mathcal{K}_n$ must be determined using the governing PDEs and the available primary variable data.

3. Methodology

3.1. AdaI-PINNs for material property inversion

Consider a neural network with depth D, comprising an input layer, $D-1$ hidden layers, and an output layer. In the k^{th} hidden layer, there are N_k neurons. Each hidden layer receives input $z^{k-1} \in \mathbb{R}^{N_{k-1}}$ from the preceding layer and applies an affine transformation described by

$$\mathcal{L}_k(z^{k-1}) = w^k z^{k-1} + b^k, \tag{2}$$

where the weights $w^k \in \mathbb{R}^{N_k \times N_{k-1}}$ and biases $b^k \in \mathbb{R}^{N_k}$ for the k^{th} layer are sampled independently and identically distributed (iid). An adaptive nonlinear activation function, $\sigma(na\mathcal{L}_k(z^{k-1}))$, is then applied element-wise to the transformed vector before passing it to the subsequent layer. Here, a is a hyperparameter to be optimized alongside the weights and biases, and n is a scaling factor that facilitates faster convergence to the global minimum. The activation function becomes the identity function after the output layer. Consequently, the final representation of the neural network can be expressed as the composition

$$u_\Theta(z) = (\mathcal{L}_D \circ \sigma \circ na\mathcal{L}_{D-1} \circ \ldots \circ \sigma \circ na\mathcal{L}_1)(z), \tag{3}$$

where $\circ$ denotes the composition operator, $\Theta = \{w^k, b^k, a\}_{k=1}^{D}$ represents the trainable parameters of the network, u is the network output, and $z^0 = z$ is the input. A set of m neural networks is employed to approximate the primary variable $\overline{u}_m$ across m subdomains. While these neural networks share the same weights and biases (w_u^k, b_u^k), their activation functions vary by the slope determined by the hyperparameter a_m. Similarly, distinct neural networks are used to approximate the material property $\overline{\mathcal{K}}$, and these are integrated into a composite neural network framework to construct a loss functional $\mathcal{L}(\Theta)$ as represented in the Figure 2.

The loss functional $\mathcal{L}(\Theta)$ quantifies the discrepancy between the neural network's approximation and the governing partial differential equation, the boundary conditions, interface conditions, and primary variable data, by calculating the mean squared error at a set of randomly chosen collocation points. The parameters $\Theta = \{w_u^k, b_u^k, w_{\mathcal{K}}^k, b_{\mathcal{K}}^k, a_i\}_{k,i=1}^{D,m}$ are iteratively adjusted to minimize the loss functional until it falls below a specified tolerance. The loss functional $\mathcal{L}(\Theta)$, for solving the equations (1), is defined as

$$\begin{aligned}\mathcal{L}(\Theta) =& \lambda_{\text{pde}}\text{MSE}_{\text{pde}} + \lambda_{\text{bc}}^n\text{MSE}_{\text{bc}}^n + \lambda_{\text{bc}}^d\text{MSE}_{\text{bc}}^d + \\ & \lambda_{\text{int}}^n\text{MSE}_{\text{int}}^n + \lambda_{\text{int}}^d\text{MSE}_{\text{int}}^d + \lambda_{\text{data}}\text{MSE}_{\text{data}}.\end{aligned} \tag{4}$$

In this context, $\mathrm{MSE}_{\mathrm{pde}}, \mathrm{MSE}^n_{\mathrm{bc}}, \mathrm{MSE}^d_{\mathrm{bc}}, \mathrm{MSE}^n_{\mathrm{int}}$, $\mathrm{MSE}^d_{\mathrm{int}}$, and $\mathrm{MSE}_{\mathrm{data}}$ represent the mean-squared errors calculated for the governing PDE, Neumann boundary conditions, Dirichlet boundary conditions, interface coupling conditions, and primary variable data, respectively. The coefficients $\lambda_{\mathrm{pde}}, \lambda^n_{\mathrm{bc}}, \lambda^d_{\mathrm{bc}}, \lambda^n_{\mathrm{int}}$, λ^d_{int}, and λ_{data} are introduced to weight these residuals relative to one another within the loss functional. These mean-squared residuals are determined as follows:

$$\mathrm{MSE}_{\mathrm{pde}} = \frac{1}{N_{\mathrm{pde}}} \sum_{i=i}^{N_{pde}} \left(\sum_{m=1}^{2} \left[\nabla \cdot (\overline{\mathcal{K}}_{mi} \nabla \overline{u}_{mi}) - f_{mi} \right]^2 \right), \tag{5a}$$

$$\mathrm{MSE}^n_{\mathrm{bc}} = \frac{1}{N^n_{\mathrm{bc}}} \sum_{i=1}^{N^n_{\mathrm{bc}}} \left(\sum_{m=1}^{2} \left[\mathcal{K}_{mi} \nabla \overline{u}_{mi} \cdot \mathbf{n_m} - \Lambda^n_{mi} \right]^2 \right), \tag{5b}$$

$$\mathrm{MSE}^d_{\mathrm{bc}} = \frac{1}{N^d_{\mathrm{bc}}} \sum_{i=1}^{N^d_{\mathrm{bc}}} \left(\sum_{m=1}^{2} \left[\overline{u}_{mi} - \Lambda^d_{mi} \right]^2 \right), \tag{5c}$$

$$\mathrm{MSE}^n_{\mathrm{int}} = \frac{1}{N_{\mathrm{int}}} \left(\sum_{i=1}^{N_{\mathrm{int}}} \left[[\![\overline{\mathcal{K}} \nabla \overline{u}]\!]_i \cdot \mathbf{n}_{\mathrm{int}} - q_i \right]^2 \right), \tag{5d}$$

$$\mathrm{MSE}^d_{\mathrm{int}} = \frac{1}{N_{\mathrm{int}}} \left(\sum_{i=1}^{N_{\mathrm{int}}} \left[[\![\overline{u}]\!]_i - p_i \right]^2 \right), \tag{5e}$$

$$\mathrm{MSE}_{\mathrm{data}} = \frac{1}{N_{\mathrm{data}}} \sum_{i=1}^{N_{\mathrm{data}}} \left(\sum_{m=1}^{2} \left[\overline{u}_{mi} - \Lambda^{\mathrm{data}}_{mi} \right]^2 \right). \tag{5f}$$

Here, N_{pde} denotes the number of collocation points used to evaluate the residuals of the PDEs. Similarly, N^n_{bc} and N^d_{bc} correspond to the number of points where Neumann and Dirichlet boundary conditions are enforced, respectively. N^n_{int} refers to the points where the flux discontinuity is specified, and N^d_{int} indicates the points where the discontinuity in the field variable is defined. Finally, N_{data} represents the points where the primary variable data are available.

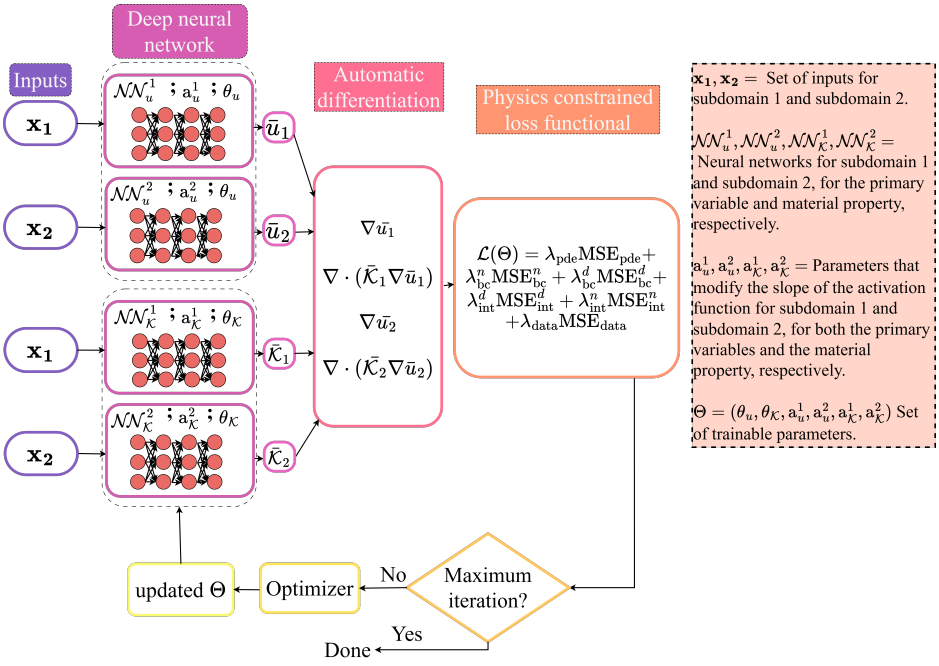


Figure 2: Schematic of an AdaI-PINNs architecture for the inverse problem, with one interface separating the domain into two subdomains

3.2. WandB: Hyperparameter search

Weights & Biases (WandB) is a widely utilized online platform in the deep learning community, offering tools for tracking and managing hyperparameters during model training runs. A key feature of WandB is Sweeps, which enables systematic optimization of model hyperparameters. The hyperparameter search space is predefined and stored in a Sweeps configuration dictionary. A Sweep may involve multiple runs based on this configuration, with hyperparameter sampling performed via grid search, random search, or Bayesian optimization. In our study, we employed Bayesian optimization to effectively navigate the predefined hyperparameter space and determine the optimal set of hyperparameters. While hyperparameter tuning in deep learning is typically focused on maximizing accuracy on a validation dataset, our approach involved minimizing the relative L_2 error between the exact solution and the neural network approximation, evaluated at specific points within the domain.

4. Numerical Examples

In this section, the proposed framework is applied to approximate solutions of two-dimensional (2D) Poisson's equations with discontinuous spatially varying coefficients. The results are compared with the numerical solution obtained using the finite element method in FreeFEM++. A uniform grid of collocation points is generated to solve all the problems. The use of different activation functions across the interface in the AdaI-PINNs architecture facilitates an accurate approximation of the local character of the solution near the interface even with a uniform grid of collocation points (see also, Sarma et al. [20] and Roy et al. [23]. The number of sampling points (both collocation and the locations where the primary variable is known) are determined through trial and error, by gradually increasing their number until AdaI-PINNs produced satisfactory results. For all the problems, weights and biases are initialized using the Glorot scheme [25] to improve convergence behavior. A learning rate of 10^{-3} is adopted throughout this study. At arbitrary points in the domain, the field variable is assumed to be known and is obtained from the finite element solution obtained using the open-source finite element simulator FreeFEM++ [26]. The approximation accuracy is measured through the root-mean-square error (RMSE) of the numerical solution evaluated against the FEM solution. Proposed AdaI-PINNs for the inverse problem were implemented as Python jupyter notebooks in Kaggle Notebooks[1], and the JAX library [27] was used to perform automatic differentiation [28]. All simulations were performed on an Nvidia Tesla T4 GPU in Kaggle. Adam optimizer was used for training the AdaI-PINNs model. We conducted a cost comparison between AdaI-PINNs, M-PINN, XPINNs, and PINNs using a metric called relative cost $\mathcal{C}_t$, defined as the ratio of the execution time of each method to that of the AdaI-PINNs framework.

4.1. 1D Diffusion with one interface

For the first problem, we consider the model problem discussed in Equations (1) defined over a 1D computational domain, $\Omega = \{0 \leq x \leq 1\}$. The location of the material interface is specified at $\Gamma_{\text{int}} = x = 0.5$, such that $\Omega_1 = \{0 \leq x \leq 0.5\}$ and $\Omega_2 = \{0.5 \leq x \leq 1\}$. The governing equation and boundary conditions for both subdomains Ω_m (for $m = 1$ and 2) are given

[1] https://www.kaggle.com/

as:

$$\begin{aligned} \frac{d}{dx}\left(\mathcal{K}_m \frac{du_m}{dx}\right) &= -10 \quad \text{in } \Omega_m, \quad m = 1, 2, \\ u_1 &= 0 \quad \text{at } x = 0, \\ \mathcal{K}_2 \frac{du_2}{dx} &= 8 \quad \text{at } x = 1, \\ [\![u]\!] &= 0, \quad \text{at } x = 0.5 \\ [\![\mathcal{K}\frac{du}{dx}]\!] &= 0 \quad \text{at } x = 0.5. \end{aligned} \tag{6}$$

The material property $\mathcal{K}$ follows a normal distribution across the domain, characterized by distinct parameters in two subdomains. In subdomain Ω_1, $\mathcal{K}$ has a mean of 1 and a standard deviation of 0.1, while in subdomain Ω_2, it has a mean of 2 with the same standard deviation of 0.1. This spatial distribution of $\mathcal{K}$ is illustrated in Figure 3(b).

The solution to Equation (6), obtained using the finite element method (FEM) implemented in FreeFEM++ [26], is presented in Figure 3(a). This solution incorporates the specified $\mathcal{K}$ distribution, boundary conditions, and interface conditions.

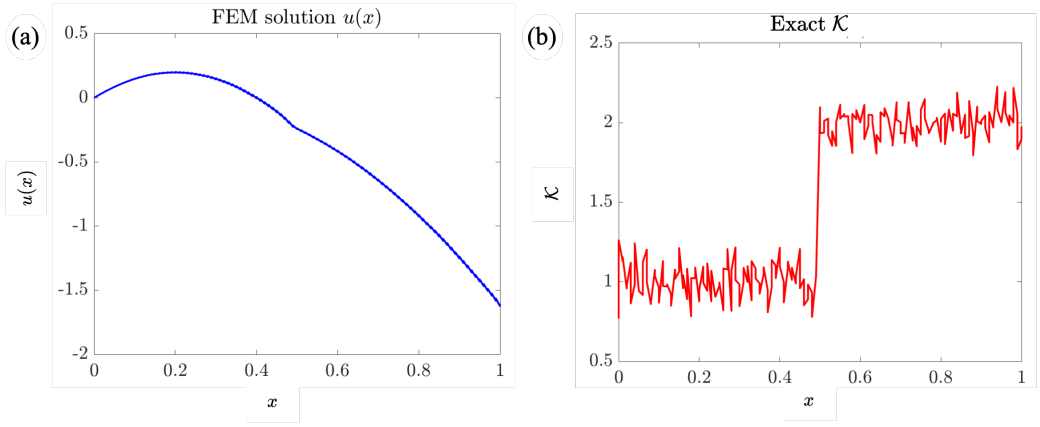


Figure 3: (a) FEM solution based on the specified $\mathcal{K}$ distribution, boundary conditions, and interface conditions. (b) Spatial distribution of $\mathcal{K}$ across the domain.

To simulate known data points, we consider the primary variable to be exactly known (without noise) at 20 distinct locations within the computational domain. These values are extracted from the FEM solution. The

sampling points are uniformly distributed within each subdomain, ensuring that no point coincides with the interface between Ω_1 and Ω_2. The spatial arrangement of these data points is depicted in Figure 4.

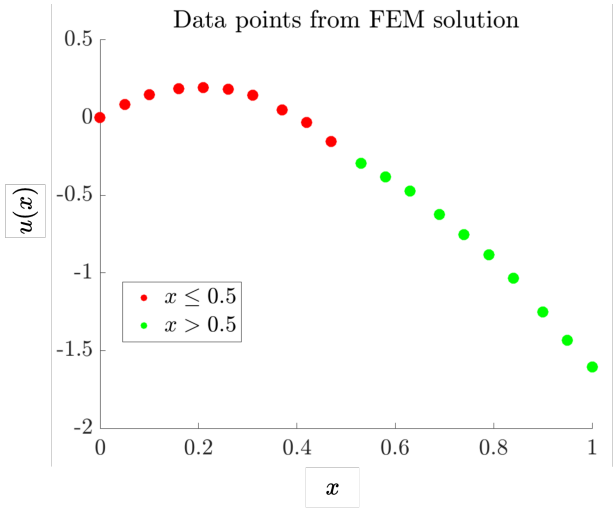


Figure 4: Distribution of training data from FEM representing the primary variable u(x) for a heterogeneous problem with an interface at x = 0.5. The red data points represent data in Ω_1 and green data points represent data in Ω_2

A Bayesian hyperparameter search was conducted using Weights & Biases (WandB) to determine the optimal parameters for the AdaI-PINNs framework. The search space for this optimization is detailed in Table 1. A total of 50 runs were performed, to minimize the L_2 norm of $\mathcal{K}$.

The optimal hyperparameters derived from this search were as follows: a scaling factor n of 6, and initial values for the adaptive activation parameters a_u^1, a_u^2, $a_{\mathcal{K}}^1$, and $a_{\mathcal{K}}^2$ set to 0.5, 1, 0.5, and 1, respectively. The network architecture that yielded the best L_2 error consisted of 3 hidden layers with 20 neurons each. The weight for the PDE loss (λ_{pde}) was set to 10, while the weight for the data loss (λ_{data}) was 50. All other weights were set to unity. The problem was solved using 40 collocation points, evenly distributed with 20 points in each domain. The hyperbolic tangent function was employed as the activation function.

The results are shown in Figure 5 (a) and demonstrate that the proposed AdaI-PINNs framework successfully captured the variation of the material property close to mean values, with a relative L_2 error on the order of

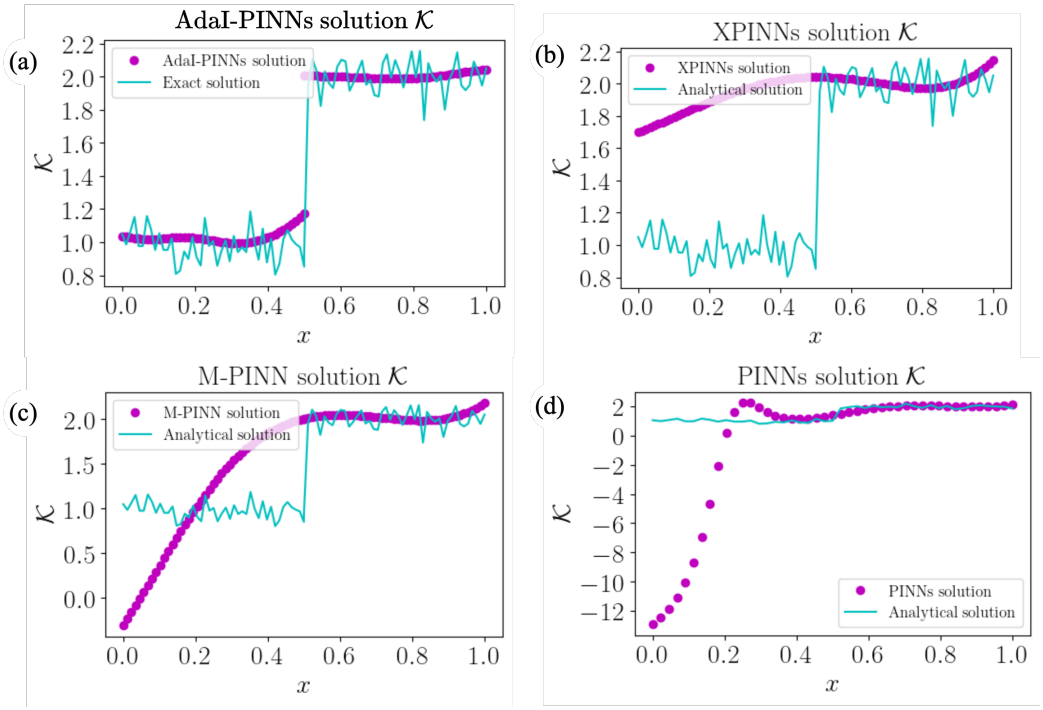


Figure 5: Results for the problem described in Section 4.1: (a) AdaI-PINNs approximation of $\mathcal{K}$. (b) XPINN approximation of $\mathcal{K}$. (c) M-PINN approximation of $\mathcal{K}$. (d) (b) PINNs approximation of $\mathcal{K}$.

$\mathcal{O}(10^{-2})$. Moreover, the framework accurately approximated the field variable u with a root mean square error (RMSE) of $\mathcal{O}(10^{-3})$, while the RMSE for the approximated $\mathcal{K}$ was on the order of $\mathcal{O}(10^{-1})$.

To evaluate the performance of our proposed framework, we conducted a comparative analysis with other established frameworks, namely M-PINN, XPINNs, and PINNs. For these comparisons, we aimed to keep the choice of hyperparameters as close as possible to the optimal configuration found for AdaI-PINNs. Limited grid searches were performed to ensure the best possible approximations for each framework.

For M-PINNs and XPINNs, we maintained the same layer size and width as in AdaI-PINNs. However, unlike the proposed composite network approach, these frameworks utilized neural networks with two outputs corresponding to the variables u and $\mathcal{K}$. PINNs lack domain decomposition and employ a single neural network, so we used a larger architecture consisting

of 5 hidden layers with 20 neurons each. All other parameters were kept consistent with those used in AdaI-PINNs.

The comparative results are presented in Figure 5 and Table 2. Both M-PINN and XPINN showed a significant increase in L_2 error by two orders of magnitude compared to AdaI-PINNs, as depicted in Figure 5 (b) and (c), respectively. The standard PINN approach struggled considerably in capturing the variable $\mathcal{K}$, resulting in poor performance as shown in Figure 5 (d).

The approximation of the primary variables is shown in Figure 6. These results demonstrate that AdaI-PINNs provide a more accurate approximation of the primary variable compared to XPINNs, M-PINN, and PINNs when using the same amount of data. This improvement is due to AdaI-PINNs' ability to effectively capture the variable $\mathcal{K}$. However, while increasing the number of data points may lead to the convergence of u across all methods, it does not necessarily guarantee an accurate approximation of $\mathcal{K}$.

Table 1: Sweep configuration: The search space of the hyperparameters

Hyperparameters	**Values**
Activation function	sin, tanh, swish
Collocation points	40, 60, 80
Hidden size of u_1, u_2, $\mathcal{K}_1$ & $\mathcal{K}_2$	3, 4, 5
Number of neuron for u_1, u_2, $\mathcal{K}_1$ & $\mathcal{K}_2$	20, 50, 100, 200
# of iterations	70000, 90000, 120000
n (scaling factor)	1, 2, 4, 6, 8
λ_{pde}, λ_{bc}^{d}, λ_{bc}^{n}, λ_{int}^{n}, λ_{int}^{d}, λ_{data}	1, 10, 50, 100, 150, 200

4.2. 2D Diffusion with straight interface

Next, we consider the Poisson equation over a 2D computational domain, $\Omega = (0,1) \times (0,1)$, featuring a material interface at $\Gamma_{\text{int}} = \{\mathbf{x} : y = x\}$, which divides Ω into two subdomains: $\Omega_1 = \{\mathbf{x} : y \leq x\}$ and $\Omega_2 = \{\mathbf{x} : y > x\}$. The governing equations, along with the boundary and interface conditions,

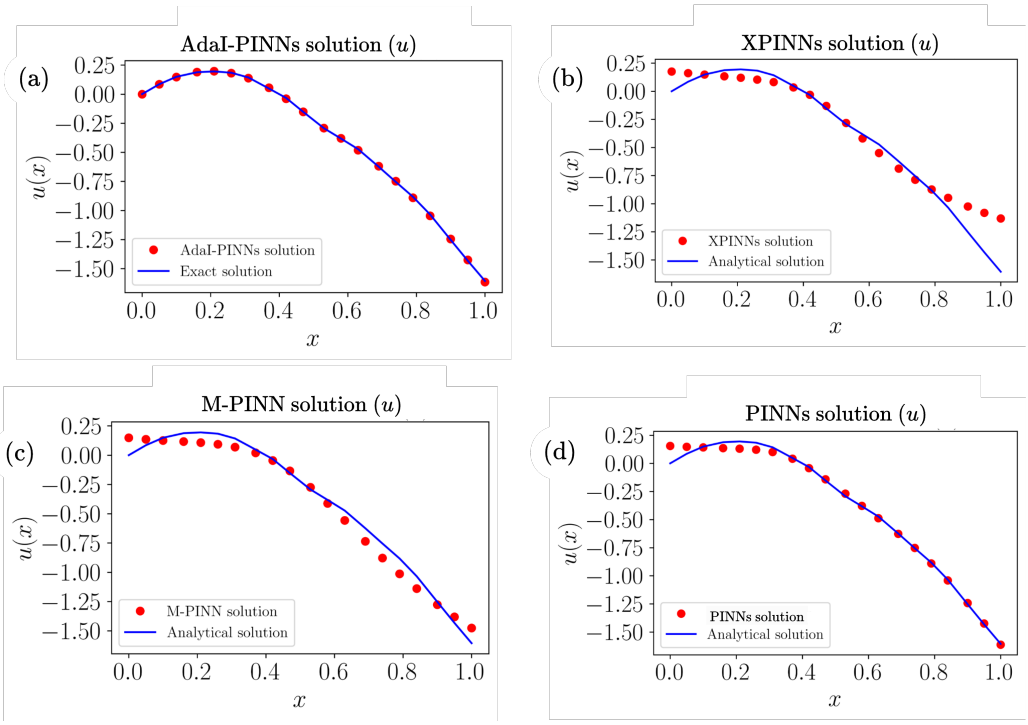


Figure 6: Results for the problem described in Section 4.1: (a) AdaI-PINNs approximation of u. (b) XPINN approximation of u. (c) M-PINN approximation of u. (d) (b) PINNs approximation of u.

are given by:

$$\begin{aligned} \nabla \cdot (\mathcal{K}_m \nabla u_m) &= -10 \qquad \text{in } \Omega_m \quad \text{for m} = 1,2, \\ u_1 &= 1 \qquad \text{on } \partial\Omega_1^d, \\ u_2 &= 1 \qquad \text{on } \partial\Omega_2^d, \\ [\![u]\!] &= 0, \qquad \text{on } \Gamma_{\text{int}}, \\ [\![\mathcal{K}\nabla u]\!] \cdot \mathbf{n}_{\text{int}} &= 0 \qquad \text{on } \Gamma_{\text{int}}. \end{aligned} \tag{7}$$

The boundaries where Dirichlet conditions are specified are denoted by $\partial\Omega_1^d$ and $\partial\Omega_2^d$. The material property $\mathcal{K}$ is characterized by a normal distribution. In the region Ω_1, it has a mean value of 1 and a standard deviation of 0.1. In Ω_2, the mean is 2, while the standard deviation remains 0.1. This distribution is visually represented in Figure 7. The same figure also displays the solution to Equations (7), which was computed using the FreeFEM++ software.

Table 2: Results of AdaI-PINNs, M-PINN, XPINNs and PINNs frameworks for problem shown in Section 4.1

Metrics	**AdaI-PINNs**	**M-PINN**	**XPINNs**	**PINNs**
Relative L_2 error on $\mathcal{K}$	9.92×10^{-2}	$5.23 \times 10^{+0}$	$4.05 \times 10^{+0}$	$2.14 \times 10^{+1}$
RMSE on u, $\mathcal{K}$	$\mathcal{O}(10^{-3})$, 1.55×10^{-1}	$\mathcal{O}(10^{-1})$, 8.65×10^{-1}	$\mathcal{O}(10^{-1})$, 6.70×10^{-1}	$\mathcal{O}(10^{-1})$, 4.95×10^{-1}
MSE on $[\![u]\!]$, $[\![\mathcal{K}\nabla u]\!] \cdot \mathbf{n}_{\text{int}}$	1.81×10^{-4}, 8.44×10^{-5}	1.27×10^{-4}, 8.87×10^{-3}	2.10×10^{-3}, 6.58×10^{-3}	9.44×10^{-3}, 3.03×10^{-2}
Maximum absolute error on u, $\mathcal{K}$	8.25×10^{-2}, 3.54×10^{-1}	1.72×10^{-1}, $1.87 \times 10^{+0}$	5.20×10^{-1}, $1.18 \times 10^{+0}$	1.31×10^{-1}, $1.15 \times 10^{+1}$
Relative cost $\mathcal{C}_t$	1	1.23	1.04	0.93
Training time (s)	138.11	169.87	143.63	128.71
# of parameters for NN	1,802	1,804	1,804	1764

For our problem, data points were sampled from the FEM solution on a uniform 44×44 grid, as shown in Figure 8. Similar to the last problem, a Bayesian hyperparameter optimization was conducted using WandB, with the search space outlined in Table 1. The only modification to the search space was the number of collocation points, which were set to 400, 625, 900, 1225, and 1600.

The hyperparameter search yielded an optimal network architecture for both u and k consisting of 5 hidden layers with 50 neurons each. This configuration was implemented for the AdaI-PINNs, M-PINN, and XPINNs frameworks. For the standard PINN approach, a slightly larger architecture of 6 hidden layers with 50 neurons each was utilized. Across all frameworks, we employed 400 collocation points and set λ_{data} to 100, with all other weights fixed at 1. The hyperbolic tangent function was chosen as the activation function for all frameworks, and each model was trained for 120,000 itera-

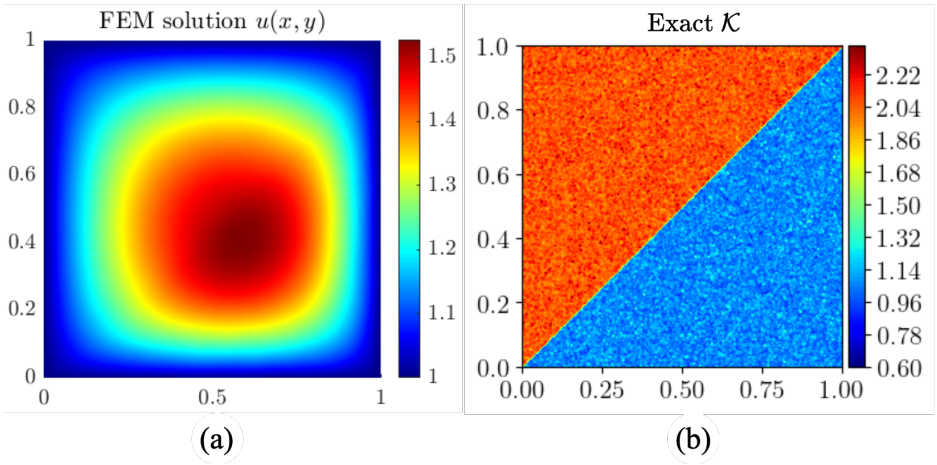


Figure 7: (a) FEM solution based on the specified $\mathcal{K}$, boundary conditions, and interface conditions. (b) Spatial distribution of $\mathcal{K}$ across the domain.

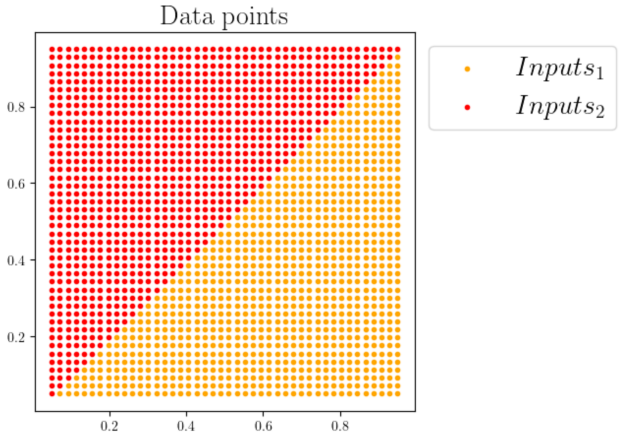


Figure 8: Uniform grid with 1,936 data points where the primary variable u is collected.

tions.

The proposed AdaI-PINNs framework demonstrated superior performance in approximating $\mathcal{K}$, achieving a relative L_2 error on the order of $\mathcal{O}(10^{-2})$. As illustrated in Figure 9 (a) and (b), the maximum absolute error of 0.84 was observed at the interface, indicating a high degree of accuracy even in challenging regions of the domain.

In comparison, both the M-PINN and XPINN frameworks were able to

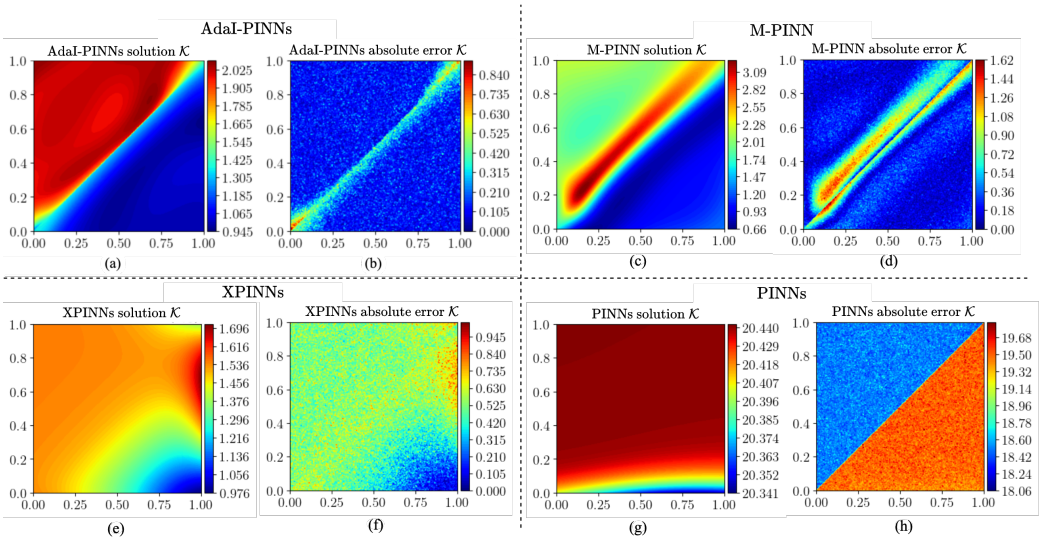


Figure 9: Results for problem shown in Section 4.2: (a) AdaI-PINNs approximation of $\mathcal{K}$. (b) Absolute error in the AdaI-PINNs approximation. (c) M-PINN approximation of $\mathcal{K}$. (d) Absolute error in the M-PINN approximation. (e) XPINNs approximation of $\mathcal{K}$. (f) Absolute error in the XPINNs approximation. (g) PINNs approximation of $\mathcal{K}$. (h) Absolute error in the PINNs approximation.

approximate $\mathcal{K}$ with relative L_2 errors in the order of $\mathcal{O}(10^{-1})$. The contour plots from Figures 9(a), 9(c), 9(e), and 9(g) show that the spatial variation of $\mathcal{K}$ is accurately captured only by AdaI-PINNs. The standard PINNs approach struggled significantly in this task. Its inability to accurately approximate the discontinuous primary variable u led to a consequent failure in approximating $\mathcal{K}$, highlighting the limitations of this method for problems involving discontinuities.

A comparison of the performance metrics for each framework is presented in Table 3. This table provides detailed information on the Root Mean Square Error (RMSE), computational cost, and the number of parameters for each method. Notably, the AdaI-PINNs framework not only outperforms the other methods in terms of accuracy but also demonstrates superior efficiency in terms of relative computational cost $\mathcal{C}_t$.

The approximation accuracy of the primary variables provides further insight into each framework's performance, as illustrated in Figure 10. AdaI-PINNs achieves the highest accuracy in approximating u with an RMSE of $\mathcal{O}(10^{-3})$, while M-PINN follows with an RMSE of $\mathcal{O}(10^{-2})$. The superior

Table 3: Results of AdaI-PINNs, M-PINN, XPINNs and PINNs framework for problem shown in Section 4.2

Metrics	**AdaI-PINNs**	**M-PINN**	**XPINNs**	**PINNs**
Relative L_2 error on $\mathcal{K}$	9.92×10^{-2}	2.49×10^{-1}	2.90×10^{-1}	$1.19 \times 10^{+1}$
RMSE on u, $\mathcal{K}$	$\mathcal{O}(10^{-3})$, 1.55×10^{-1}	$\mathcal{O}(10^{-2})$, 3.95×10^{-1}	$\mathcal{O}(10^{-2})$, 4.60×10^{-1}	$\mathcal{O}(10^{-1})$, $1.89 \times 10^{+1}$
MSE on $[\![u]\!]$, $[\![\mathcal{K}\nabla u]\!] \cdot \mathbf{n}_{\text{int}}$	1.11×10^{-5}, 3.83×10^{-5}	2.52×10^{-4}, 2.99×10^{-6}	5.50×10^{-1}, 2.19×10^{-2}	1.26×10^{-1}, $2.40 \times 10^{+0}$
Maximum absolute error on u, $\mathcal{K}$	1.55×10^{-2}, 8.79×10^{-1}	4.20×10^{-2}, $1.62 \times 10^{+0}$	1.38×10^{-1}, 9.45×10^{-1}	3.31×10^{-1}, $1.97 \times 10^{+2}$
Relative cost $\mathcal{C}_t$	1	1.34	1.06	0.88
Training time (s)	702.40	941.21	744.54	618.11
# of parameters for NN	20,802	20,804	20,804	12,954

performance of these frameworks extends to their handling of interface conditions, as evidenced by the low interface MSE shown in Table 3. In contrast, XPINNs exhibits larger errors at the interface due to its limitations in capturing interface conditions accurately. PINNs, lacking any interface recognition capability, fails to satisfy the interface conditions altogether.

4.3. 2D Diffusion with a circular interface

Next, we consider the Poisson equation over a 2D computational domain, $\Omega = (0,1) \times (0,1)$, featuring a material interface at $\Gamma_{\text{int}} = \{\mathbf{x} : (x-0.5)^2 + (y-0.5)^2 = 0.05\}$, which divides Ω into two subdomains: $\Omega_1 = \{\mathbf{x} : (x-0.5)^2 + (y-0.5)^2 \leq 0.05\}$ and $\Omega_2 = \{\mathbf{x} : (x-0.5)^2 + (y-0.5)^2 > 0.05\}$. The governing equations, along with the boundary and interface conditions, are

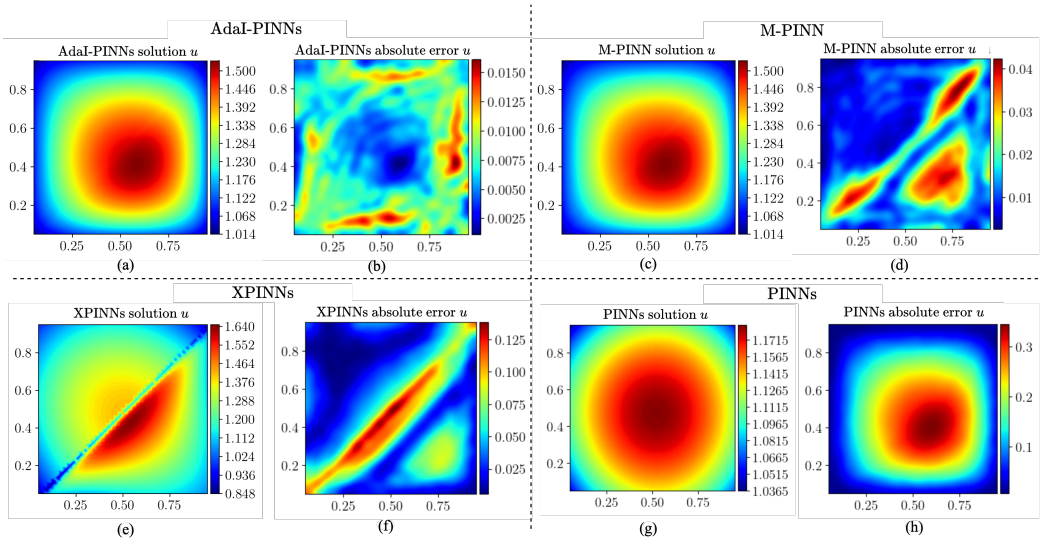


Figure 10: Results for problem shown in Section 4.2: (a) AdaI-PINNs approximation of u. (b) Absolute error in the AdaI-PINNs approximation. (c) M-PINN approximation of u. (d) Absolute error in the M-PINN approximation. (e) XPINNs approximation of u. (f) Absolute error in the XPINNs approximation. (g) PINNs approximation of u. (h) Absolute error in the PINNs approximation.

given by:

$$\begin{aligned} \nabla \cdot (\mathcal{K}_m \nabla u_m) &= -10 \qquad \text{in } \Omega_m \quad \text{for m} = 1{,}2, \\ u_2 &= 1 \qquad \text{on } \partial\Omega_2^d, \\ [\![u]\!] &= 0 \qquad \text{on } \Gamma_{\text{int}}, \\ [\![\mathcal{K}\nabla u]\!] \cdot \mathbf{n}_{\text{int}} &= 0 \qquad \text{on } \Gamma_{\text{int}}, \end{aligned} \tag{8}$$

where $\partial\Omega_2^d$ represents the boundary sections with prescribed Dirichlet conditions. The material property $\mathcal{K}$ follows a normal distribution, with a mean of 1 and standard deviation of 0.1 in subdomain Ω_1, and a mean of 2 with the same standard deviation in subdomain Ω_2, as illustrated in Figure 11. The solution to Equations (8) obtained using FreeFEM++ is shown in Figure 11.

From the FEM solution, data points were extracted on a 44×44 uniform grid, as illustrated in Figure 12, which shows the data points used for the inverse problem. The inverse problem was solved using both AdaI-PINNs and M-PINN, and the results were compared.

Once again, we employed a neural network architecture derived from hyperparameter search using WandB in this study. The network used to ap-

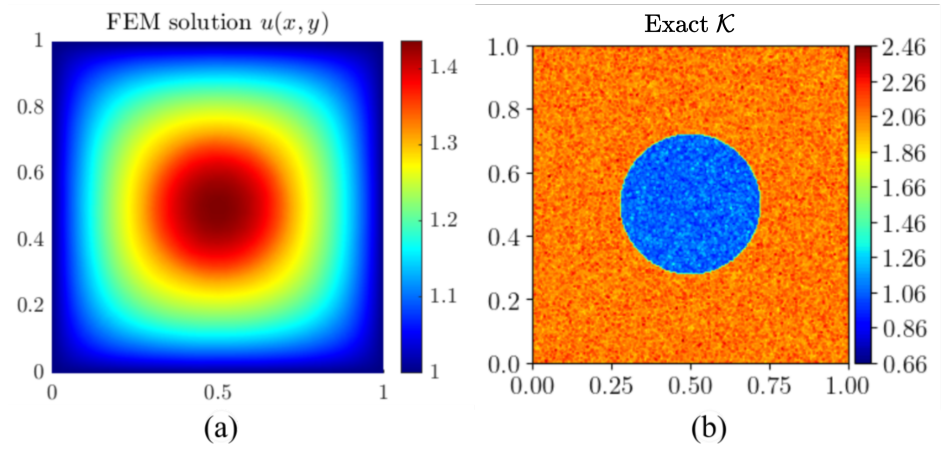


Figure 11: (a) FEM solution based on the specified $\mathcal{K}$, boundary conditions, and interface conditions. (b) Spatial distribution of $\mathcal{K}$ across the domain.

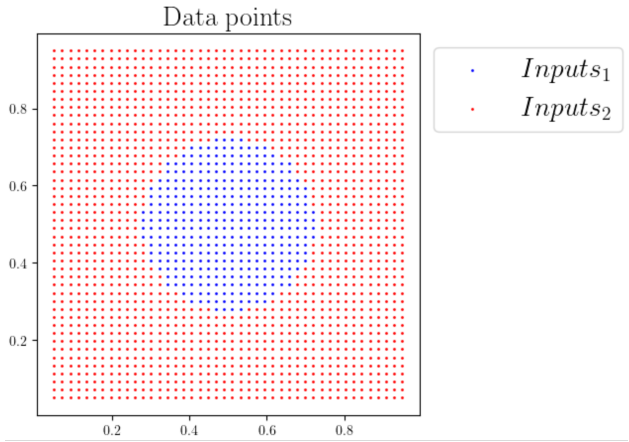


Figure 12: Uniform grid with 1,936 data points where the primary variable u is collected.

proximate u consists of 5 hidden layers with 100 neurons each, while the network for $\mathcal{K}$ is composed of 5 hidden layers with 200 neurons each. This configuration was determined to be optimal for the problem at hand. The scaling factor n was set to 2, and the initial adaptive activation parameters a_u^1, a_u^2, $a_{\mathcal{K}}^1$, and $a_{\mathcal{K}}^2$ were initialized to 0.1, 1, 0.1, and 1, respectively. The weight for the PDE loss term, λ_{pde}, was set to 10, while all other loss terms were weighted at 150.

For comparative purposes, we maintained the same hyperparameters for

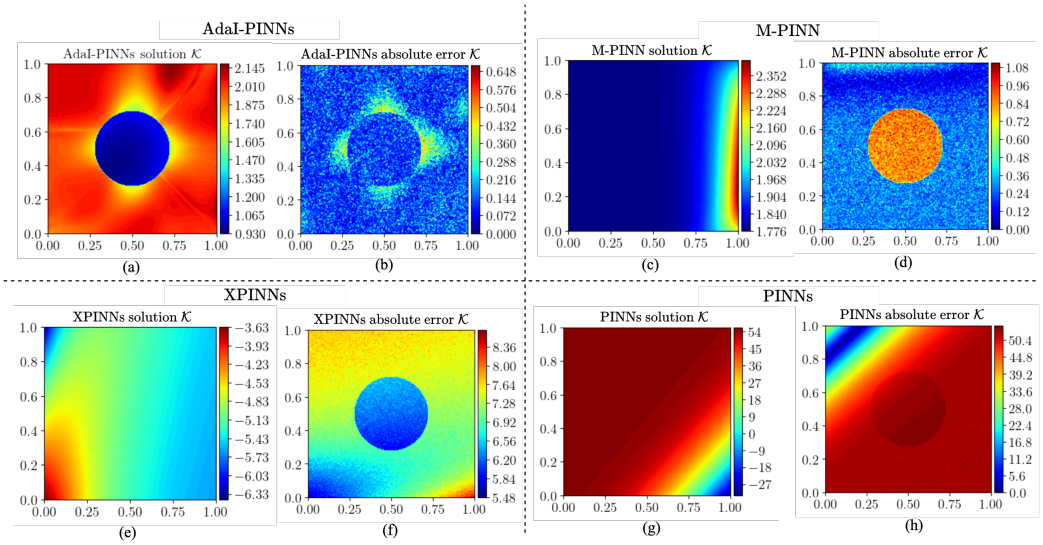


Figure 13: Results for problem shown in Section 4.3: (a) AdaI-PINNs approximation of $\mathcal{K}$. (b) Absolute error in the AdaI-PINNs approximation. (c) M-PINN approximation of $\mathcal{K}$. (d) Absolute error in the M-PINN approximation. (e) XPINNs approximation of $\mathcal{K}$. (f) Absolute error in the XPINNs approximation. (g) PINNs approximation of $\mathcal{K}$. (h) Absolute error in the PINNs approximation.

the M-PINNs and XPINNs frameworks. However, for the standard PINNs approach, we used a network with 5 hidden layers and 200 neurons each, to account for the lack of separate networks for u and $\mathcal{K}$.

In the case of a circular inclusion problem, the proposed AdaI-PINNs framework demonstrated superior performance compared to other frameworks. As shown in Table 4, there is a clear trend of increasing relative L_2 error as we move from left to right across different frameworks, with AdaI-PINNs exhibiting the lowest error.

The advantages of the proposed composite neural network approach become particularly evident in this type of problem. Figure 13 (a) and (b) illustrate the approximation of $\mathcal{K}$ using the proposed framework. Clearly, only the proposed framework is able to capture the correct spatial variation of the material properties. Notably, the maximum absolute error in the approximation of $\mathcal{K}$ by AdaI-PINNs was observed to be 0.648 at the interface, which is a significant improvement over other methods. These results highlight an important characteristic of our proposed framework: its performance advantage becomes more pronounced as problem complexity increases.

Table 4: Results of AdaI-PINNs and M-PINN framework for problem shown in Section 4.3

Metrics	**AdaI-PINNs**	**M-PINN**	**XPINNs**	**PINNs**
Relative L_2 error on $\mathcal{K}$	6.733×10^{-2}	1.970×10^{-1}	$3.712 \times 10^{+0}$	$2.622 \times 10^{+1}$
RMSE on u, $\mathcal{K}$	$\mathcal{O}(10^{-3})$, 1.304×10^{-1}	$\mathcal{O}(10^{-2})$, 3.711×10^{-1}	$\mathcal{O}(10^{-1})$, $6.990 \times 10^{+0}$	$\mathcal{O}(10^{-1})$, $4.938 \times 10^{+1}$
MSE on $[\![u]\!]$, $[\![\mathcal{K}\nabla u]\!] \cdot \mathbf{n}_{\text{int}}$	3.18×10^{-6}, 4.38×10^{-5}	6.36×10^{-5}, 1.37×10^{-5}	4.70×10^{-3}, 6.44×10^{-2}	5.72×10^{-1}, $3.54 \times 10^{+0}$
Maximum absolute error on u, $\mathcal{K}$	1.10×10^{-2}, 6.48×10^{-1}	1.71×10^{-2}, $1.08 \times 10^{+0}$	1.32×10^{-1}, $8.74 \times 10^{+0}$	1.17×10^{-1}, $5.24 \times 10^{+1}$
Relative cost $\mathcal{C}_t$	1	1.70	1.45	0.923
Training time (s)	1051.26	1787.14	1524.327	970.31
# of parameters for NN	2,02,402	2,02,404	2,02,404	1,61,604

The superior performance extends to the approximation of primary variables, as demonstrated in Figure 14. While all methods show maximum errors on the order of $\mathcal{O}(10^{-1})$ due to the discontinuous material property, our framework consistently achieves the lowest error at the interfaces compared to other approaches.

4.4. 2D Diffusion with multiple circular interfaces

For the last problem, we consider the Poisson equation over a two-dimensional computational domain, $\Omega = (0, 1) \times (0, 1)$, featuring three distinct material interfaces. These interfaces are defined as:

$$\Gamma_{\text{int}_1} = \{\mathbf{x} \in \mathbb{R}^2 : (x - 0.75)^2 + (y - 0.75)^2 = 0.02\}, \tag{9}$$

$$\Gamma_{\text{int}_2} = \{\mathbf{x} \in \mathbb{R}^2 : (x - 0.25)^2 + (y - 0.5)^2 = 0.03\}, \tag{10}$$

$$\Gamma_{\text{int}_3} = \{\mathbf{x} \in \mathbb{R}^2 : (x - 0.6)^2 + (y - 0.25)^2 = 0.02\}. \tag{11}$$

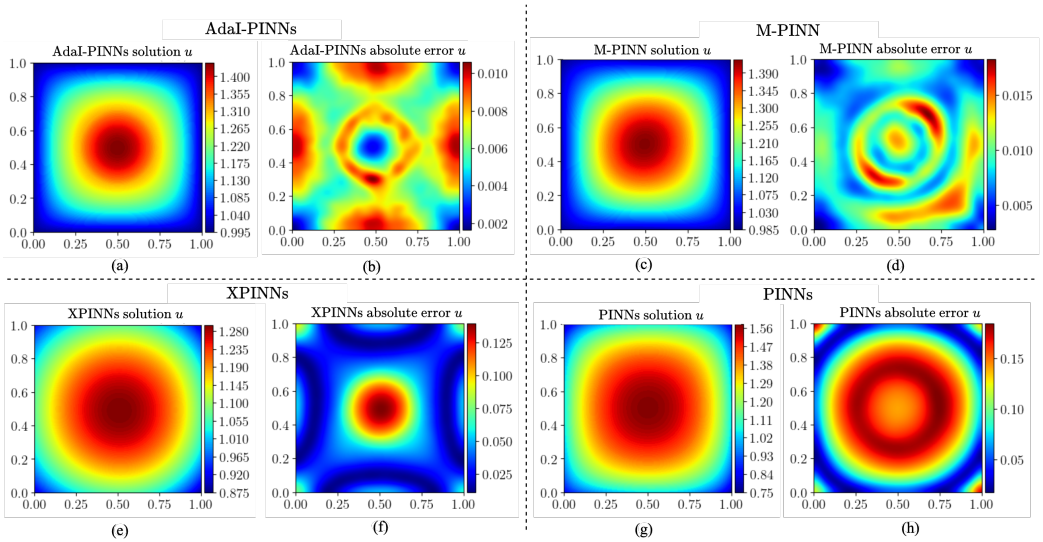


Figure 14: Results for problem shown in Section 4.3: (a) AdaI-PINNs approximation of u. (b) Absolute error in the AdaI-PINNs approximation. (c) M-PINN approximation of u. (d) Absolute error in the M-PINN approximation. (e) XPINNs approximation of u. (f) Absolute error in the XPINNs approximation. (g) PINNs approximation of u. (h) Absolute error in the PINNs approximation.

These interfaces partition the domain Ω into four subdomains: $\Omega_1, \Omega_2, \Omega_3$, and Ω_4. Specifically, Ω_1, Ω_2, and Ω_3 represent the subdomains enclosed by the interfaces $\Gamma_{\text{int}_1}, \Gamma_{\text{int}_2}$, and Γ_{int_3}, respectively, while Ω_4 denotes the region external to all three interfaces.

The problem is governed by the Poisson equation with heterogeneous coefficients, subject to appropriate boundary and interface conditions. The system of equations is given as:

$$\begin{aligned} \nabla \cdot (\mathcal{K}_m \nabla u_m) &= -10 \qquad \text{in } \Omega_m, \quad m = 1,2,3,4, \\ u_2 &= 1 \qquad \text{on } \partial\Omega_4^d, \\ [\![u]\!] &= 0 \qquad \text{on } \Gamma_{\text{int}_i}, \quad i = 1,2,3, \\ [\![\mathcal{K}\nabla u]\!] \cdot \mathbf{n}_{\text{int}_i} &= 0 \qquad \text{on } \Gamma_{\text{int}_i}, \quad i = 1,2,3, \end{aligned} \tag{12}$$

where $\partial\Omega_4^d$ represents the boundary sections with prescribed Dirichlet conditions, $[\![\cdot]\!]$ denotes the jump operator across interfaces, and $\mathbf{n}_{\text{int}_i}$ is the unit normal vector to the interface Γ_{int_i}.

The material property $\mathcal{K}$ is modeled as a spatially varying random field following a normal distribution:

$$\mathcal{K}_m \sim \mathcal{N}(\mu_m, \sigma^2), \quad m = 1, 2, 3, 4, \tag{13}$$

where μ_m is the mean value in subdomain Ω_m, and σ is the standard deviation. In subdomain Ω_1,Ω_2 and Ω_3, we have $\mu_1 = 1$ and $\sigma = 0.1$, while in subdomain Ω_4, $\mu_2 = 2$ with the same standard deviation. The spatial distribution of $\mathcal{K}$ across the domain is illustrated in Figure 15(b).

The solution to the system of equations (12) was obtained using the finite element method implemented in FreeFEM++. The resulting solution field is shown in Figure 15(a). From the FEM solution, data points were extracted on a 44×44 uniform grid, as illustrated in Figure 16, which shows the data points used for the inverse problem.

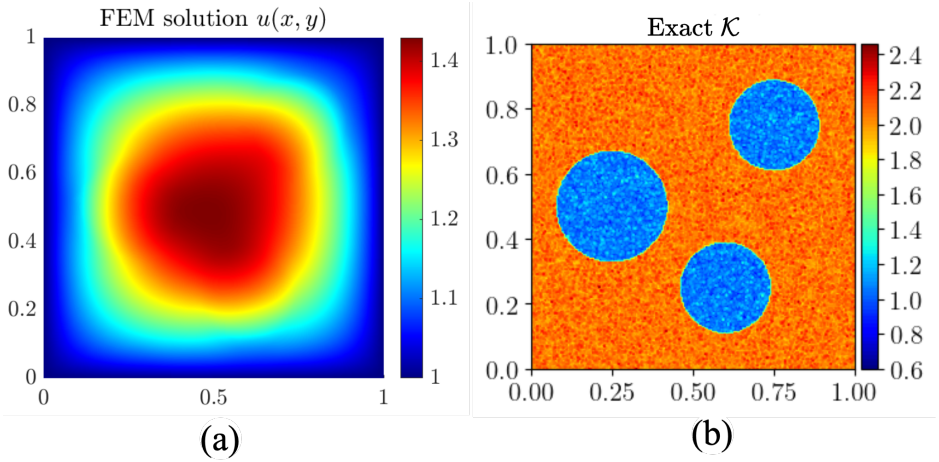


Figure 15: Numerical results: (a) FEM solution $u(\mathbf{x})$ based on the specified $\mathcal{K}$, boundary conditions, and interface conditions. (b) Spatial distribution of the material property $\mathcal{K}(\mathbf{x})$ across the domain.

A hyperparameter search yielded an optimal network architecture for the problem at hand. For the approximation of u, we employed a network with 5 hidden layers, each containing 50 neurons. The network for $\mathcal{K}$ approximation consisted of 4 hidden layers with 300 neurons each. This configuration was implemented for the AdaI-PINNs, M-PINN, and XPINNs frameworks. To ensure a fair comparison in terms of network capacity, we adapted the architecture for the standard PINNs approach, using 3 hidden layers with 512 neurons each, thus increasing the number of trainable parameters to a comparable level.

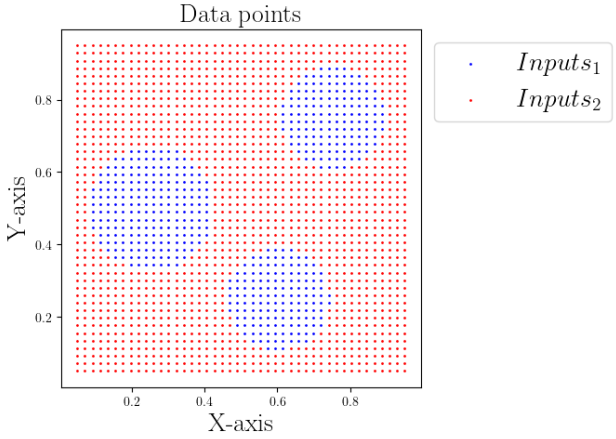


Figure 16: Uniform grid with 1,936 data points where the primary variable u is collected.

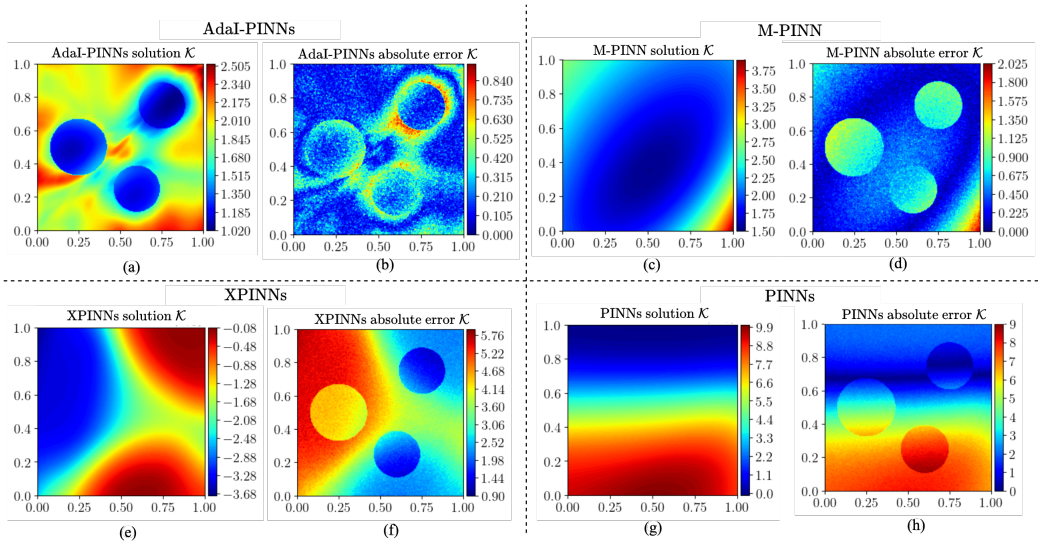


Figure 17: Results for problem shown in Section 4.1: (a) AdaI-PINNs approximation of $\mathcal{K}$. (b) Absolute error in the AdaI-PINNs approximation. (c) M-PINN approximation of $\mathcal{K}$. (d) Absolute error in the M-PINN approximation.

The results from this problem configuration follow a similar trend to our previous experiments, with the AdaI-PINNs framework consistently outperforming other approaches both in terms of approximation accuracy and computational efficiency. This reinforces the robustness of our proposed method across different problem settings.

Table 5: Results of AdaI-PINNs and M-PINN framework for problem shown in Section 4.4

Metrics	**AdaI-PINNs**	**M-PINN**	**XPINNs**	**PINNs**
Relative L_2 error on $\mathcal{K}$	1.319×10^{-1}	2.734×10^{-1}	$2.089 \times 10^{+0}$	$2.589 \times 10^{+0}$
RMSE on u, $\mathcal{K}$	$\mathcal{O}(10^{-3})$, 2.416×10^{-1}	$\mathcal{O}(10^{-2})$, 5.009×10^{-1}	$\mathcal{O}(10^{-1})$, $3.827 \times 10^{+0}$	$\mathcal{O}(10^{-1})$, $4.743 \times 10^{+0}$
MSE on $[\![u]\!]$, $[\![\mathcal{K}\nabla u]\!] \cdot \mathbf{n}_{\text{int}}$	1.09×10^{-4}, 1.66×10^{-4}	7.21×10^{-5}, 5.10×10^{-4}	5.94×10^{-3}, $1.15 \times 10^{+1}$	2.02×10^{-2}, $6.70 \times 10^{+1}$
Maximum absolute error on u, $\mathcal{K}$	1.12×10^{-2}, 8.62×10^{-1}	2.50×10^{-2}, $2.02 \times 10^{+0}$	7.02×10^{-2}, $5.82 \times 10^{+0}$	7.81×10^{-2}, $9.01 \times 10^{+0}$
Relative cost $\mathcal{C}_t$	1	1.53	1.38	1.12
Training time (s)	1000.51	1530.78	1380.70	1120.57
# of parameters for NN	2,82,502	2,82,504	2,82,504	5,27,364

While the error metrics for AdaI-PINNs and M-PINN appear to be of similar order, a closer examination reveals significant differences in their performance. As illustrated in Figure 17, the M-PINN, XPINNs, and conventional PINNs frameworks all struggle to accurately capture the variation of $\mathcal{K}$. By contrast, the spatial distribution of the material properties obtained from the proposed method closely agree with the true variation shown in Figure 15(b).

Similar to the above examples, for the case where multiple interfaces are present, AdaI-PINNs showcase better approximation of primary variable as

shown in Figure 18. The MSE values at the interface conditions, presented in Table 5, indicate that both AdaI-PINNs and M-PINN achieve the lowest errors at the interfaces. . However, this distinction is crucial, as it highlights that raw error metrics alone may not fully capture the nuances of model performance, particularly in problems involving complex spatial variations. The ability of AdaI-PINNs to more faithfully represent the underlying physical property distribution underscores its potential for applications where an accurate representation of spatial heterogeneity is critical.

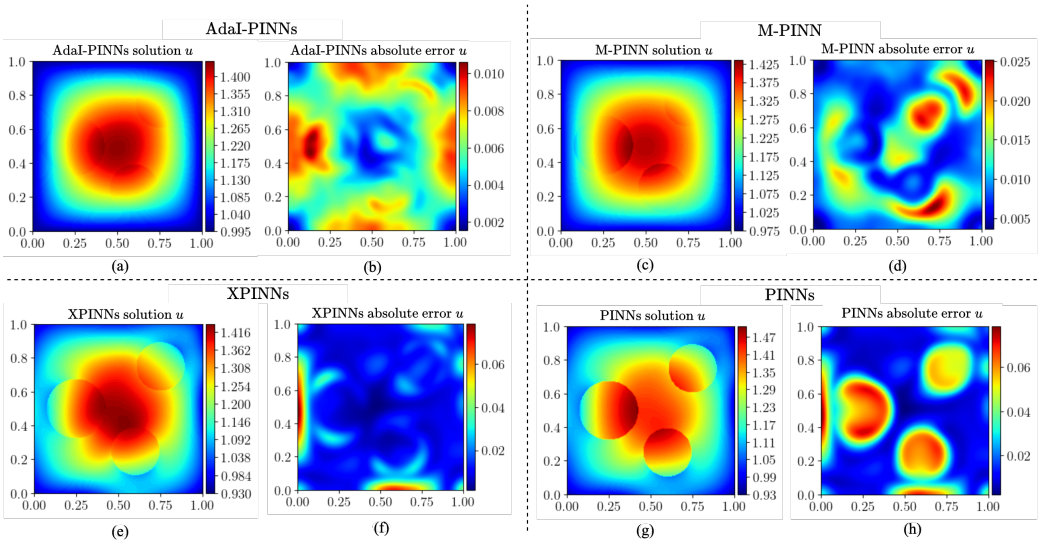


Figure 18: Results for problem shown in Section 4.1: (a) AdaI-PINNs approximation of u. (b) Absolute error in the AdaI-PINNs approximation. (c) M-PINN approximation of u. (d) Absolute error in the M-PINN approximation.

5. Limitations

Although the numerical examples presented in Section 4 show promising results, it is important to highlight the limitations of the presented framework. In principle, the presented framework can be readily extended to three-dimensional problems and for cases involving multiple intersecting interfaces. However, the computational cost of a fully three-dimensional simulation is expected to be much higher and a thorough comparison must be made with the more conventional numerical methods (FEM-based) currently available for

solving inverse problems to justify the use of PINNs for these problems. Additionally, we have studied the problems only for soft constraints on boundary and interface conditions. Applying hard constraints at the boundaries and interfaces may further improve the accuracy of AdaI-PINNs method (see e.g., [29]). For transient problems, involving moving interfaces, the framework potentially remains effective when both the interface location and its dynamic evolution are known *a priori*. However, because the interface evolution will likely be governed by another partial differential equation which depends on the gradients of the primary variable, one would end up with coupled system of PDEs. As such, a composite neural network framework must be used in conjunction with AdaI-PINNs for such problems (see, for e.g., Roy et al. [22]). When the interface geometry is unknown, significant challenges arise in accurately identifying the interface shape, which can lead to considerable errors in enforcing interface conditions. Clearly, these topics are beyond the scope of the current manuscript, but are important extensions that must be considered in future research.

6. Summary

We present an innovative approach to solving inverse problems in heterogeneous media, specifically aimed at deducing spatially varying material properties that are randomly distributed within a domain. Our method extends the AdaI-PINNs framework, employing composite neural networks to approximate both primary variables and material properties across different subdomains. The key features of our approach include two sets of parameters for approximating primary variables and material properties, and distinct neural networks characterized by adaptive activation functions, allowing for variation in function slopes.

The proposed framework is compared with the state-of-the-art M-PINN, XPINNs and the traditional PINNs framework for solving one-dimensional and two-dimensional benchmark numerical examples. The proposed framework consistently outperforms its competitors in terms of both approximation accuracy and the computational cost. In particular, the proposed framework yields L_2 errors for the material property in the order of $\mathcal{O}(10^{-2})$, and root mean square errors for the primary variable in the order of $\mathcal{O}(10^{-3})$. Furthermore, the exact spatial distribution of the material properties obtained using the proposed approach is found to be in close agreement with the true

distribution, whereas MPINN, XPINNs and conventional PINNs yield significantly worse results.

Going forward, enhancing the framework to achieve more accurate approximations of random functions and extending the approach to solve inverse problems where interface locations are not known *a priori* are of interest.

Acknowledgement

Dr. Pratanu Roy's work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under the Contract DE-AC52-07NA27344. LLNL release number is LLNL-JRNL-868933. Dr. Chandrasekhar Annavarapu gratefully acknowledges the support from ExxonMobil Corporation and IIT Madras to the Subsurface Mechanics and Geo-Energy Laboratory under the grants SP22230020CEEXXU008957 and SB0210856CEMHRD008957.

References

[1] J. Blaber, B. Adair, A. Antoniou, Ncorr: open-source 2d digital image correlation matlab software, Experimental Mechanics 55 (6) (2015) 1105–1122.

[2] S. Sony, S. Laventure, A. Sadhu, A literature review of next-generation smart sensing technology in structural health monitoring. struct control health monit 26 (3): e2321 (2019).

[3] B. F. Kennedy, P. Wijesinghe, D. D. Sampson, The emergence of optical elastography in biomedicine, Nature Photonics 11 (4) (2017) 215–221.

[4] J.-L. Gennisson, T. Deffieux, M. Fink, M. Tanter, Ultrasound elastography: principles and techniques, Diagnostic and interventional imaging 94 (5) (2013) 487–495.

[5] K. J. Parker, M. M. Doyley, D. J. Rubens, Imaging the elastic properties of tissue: the 20 year perspective, Physics in medicine & biology 56 (1) (2010) R1.

[6] R. C. Aster, B. Borchers, C. H. Thurber, Parameter estimation and inverse problems, Elsevier, 2018.

[7] D. Givoli, A tutorial on the adjoint method for inverse problems, Computer Methods in Applied Mechanics and Engineering 380 (2021) 113810.

[8] T. Bui-Thanh, O. Ghattas, J. Martin, G. Stadler, A computational framework for infinite-dimensional bayesian inverse problems part i: The linearized case, with application to global seismic inversion, SIAM Journal on Scientific Computing 35 (6) (2013) A2494–A2523.

[9] J. A. Christen, C. Fox, Markov chain monte carlo using an approximation, Journal of Computational and Graphical statistics 14 (4) (2005) 795–810.

[10] I. Lagaris, A. Likas, D. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks 9 (5) (1998) 987–1000. doi:10.1109/72.712178.

[11] 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, Journal of Computational physics 378 (2019) 686–707.

[12] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S. G. Johnson, Physics-informed neural networks with hard constraints for inverse design, SIAM Journal on Scientific Computing 43 (6) (2021) B1105–B1132.

[13] S. Cai, Z. Wang, S. Wang, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks for heat transfer problems, Journal of Heat Transfer 143 (6) (2021) 060801.

[14] K. Shukla, P. C. Di Leoni, J. Blackshire, D. Sparkman, G. E. Karniadakis, Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks, Journal of Nondestructive Evaluation 39 (2020) 1–20.

[15] M. Rasht-Behesht, C. Huber, K. Shukla, G. E. Karniadakis, Physics-informed neural networks (pinns) for wave propagation and full waveform inversions, Journal of Geophysical Research: Solid Earth 127 (5) (2022) e2021JB023120.

[16] A. D. Jagtap, G. E. Karniadakis, Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Communications in Computational Physics 28 (5) (2020).

[17] L. Papadopoulos, S. Bakalakos, S. Nikolopoulos, I. Kalogeris, V. Papadopoulos, A computational framework for the indirect estimation of interface thermal resistance of composite materials using xpinns, International Journal of Heat and Mass Transfer 200 (2023) 123420.

[18] A. D. Jagtap, Z. Mao, N. Adams, G. E. Karniadakis, Physics-informed neural networks for inverse problems in supersonic flows, Journal of Computational Physics 466 (2022) 111402.

[19] B. Zhang, G. Wu, Y. Gu, X. Wang, F. Wang, Multi-domain physics-informed neural network for solving forward and inverse problems of steady-state heat conduction in multilayer media, Physics of Fluids 34 (11) (2022).

[20] A. K. Sarma, S. Roy, C. Annavarapu, P. Roy, S. Jagannathan, Interface pinns (i-pinns): A physics-informed neural networks framework for interface problems, Computer Methods in Applied Mechanics and Engineering 429 (2024) 117135.

[21] A. Sarma, C. Annavarapu, P. Roy, S. Jagannathan, D. Valiveti, Variational interface physics informed neural networks (vi-pinns) for heterogeneous subsurface systems, in: ARMA US Rock Mechanics/Geomechanics Symposium, ARMA, 2023, pp. ARMA–2023.

[22] S. Roy, C. Annavarapu, P. Roy, D. M. V. and, Physics-informed neural networks for heterogeneous poroelastic media, International Journal for Computational Methods in Engineering Science and Mechanics 26 (2) (2025) 187–207. doi:10.1080/15502287.2024.2440420.
URL https://doi.org/10.1080/15502287.2024.2440420

[23] S. Roy, C. Annavarapu, P. Roy, A. K. Sarma, Adaptive interface-pinns (adai-pinns): An efficient physics-informed neural networks framework for interface problems, Communications in Computational Physics 37 (3) (2025) 603–622. doi:https://doi.org/10.4208/cicp.OA-2024-0131.

[24] S. Roy, D. R. Sarkar, C. Annavarapu, P. Roy, B. Lecampion, D. M. Valiveti, Adaptive interface-pinns (adai-pinns) for transient diffusion: Applications to forward and inverse problems in heterogeneous media, Finite Elements in Analysis and Design 244 (2025) 104305.

[25] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[26] F. Hecht, New development in freefem++, Journal of numerical mathematics 20 (3-4) (2012) 251–266.

[27] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python+NumPy programs (2018).
URL `http://github.com/google/jax`

[28] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, Journal of machine learning research 18 (153) (2018) 1–43.

[29] P. Roy, S. T. Castonguay, Exact enforcement of temporal continuity in sequential physics-informed neural networks, Computer Methods in Applied Mechanics and Engineering 430 (2024) 117197.