MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

AB-PINNs: Adaptive-Basis Physics-Informed Neural Networks for Residual-Driven Domain Decomposition

Botvinick-Greenhouse, Jonah; Ali, Wael H.; Benosman, Mouhacine; Mowlavi, Saviz

TR2026-116 August 27, 2026

Abstract

We introduce adaptive-basis physics-informed neural networks (AB-PINNs), an adaptive domain decomposition framework for PINNs in which learnable subdomains dynamically evolve during training to align with intrinsic features of the unknown solution. Local networks capture fine-scale features within each adaptive subdomain, while a global network learns large-scale solution structures. Furthermore, drawing inspiration from classical adaptive mesh refinement, we also modify the domain decomposition on-the-fly throughout training by introducing new subdomains in regions of high residual loss, thereby providing additional expressive power where needed. Our flexible approach to domain decomposition is well-suited for multiscale problems, as different subdomains can learn to capture different scales of the underlying solution. Moreover, the ability to introduce new subdomains during training helps prevent convergence to unwanted local minima and can reduce the need for extensive hyperparameter tuning compared to static domain decomposition approaches. Throughout, we present comprehensive numerical results demonstrating the rapid convergence of AB-PINNs compared with standard PINNs and existing, static PINN-based domain decompositions when solving multiscale differential equations.

Machine Learning: Science and Technology 2026

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

AB-PINNs: Adaptive-Basis Physics-Informed Neural Networks for Residual-Driven Domain Decomposition

Jonah Botvinick-Greenhouse[†,*]
Center for Applied Mathematics
Cornell University
Ithaca, NY, 14850
jrb482@cornell.edu

Wael H. Ali
Mitsubishi Electric Research Laboratories
Cambridge, MA, 02139
wali@merl.com

Mouhacine Benosman
Amazon Robotics
North Reading, MA, 01864
m_benosman@ieee.org

Saviz Mowlavi[*]
Mitsubishi Electric Research Laboratories
Cambridge, MA, 02139
mowlavi@merl.com

Abstract

We introduce adaptive-basis physics-informed neural networks (AB-PINNs), an adaptive domain decomposition framework for PINNs in which learnable subdomains dynamically evolve during training to align with intrinsic features of the unknown solution. Local networks capture fine-scale features within each adaptive subdomain, while a global network learns large-scale solution structures. Furthermore, drawing inspiration from classical adaptive mesh refinement, we also modify the domain decomposition on-the-fly throughout training by introducing new subdomains in regions of high residual loss, thereby providing additional expressive power where needed. Our flexible approach to domain decomposition is well-suited for multiscale problems, as different subdomains can learn to capture different scales of the underlying solution. Moreover, the ability to introduce new subdomains during training helps prevent convergence to unwanted local minima and can reduce the need for extensive hyperparameter tuning compared to static domain decomposition approaches. Throughout, we present comprehensive numerical results demonstrating the rapid convergence of AB-PINNs compared with standard PINNs and existing, static PINN-based domain decompositions when solving multiscale differential equations.

Keywords Physics-informed neural networks · Domain decomposition · Partial differential equations

1 Introduction

Forward and inverse problems related to partial differential equations (PDEs) play a crucial role in many physical and biological applications, including climate modeling, medical imaging, and heat transfer, to name a few. While classical techniques for solving PDEs, e.g., the finite difference and finite element methods [1], have led to remarkable scientific advances, these approaches also require careful setup for inverse modeling via adjoint-based optimization [2], are nontrivial to adapt to domains with complex geometry, rely on a fixed mesh, and struggle in high dimensions.

In recent years, physics-informed neural networks (PINNs) [3] have emerged as an alternative approach capable of overcoming some of these challenges. PINNs represent the unknown PDE solution via a deep neural network and are trained to minimize a residual loss which enforces the PDE constraint throughout the problem domain. Thus, PINNs are mesh-free, can be easily deployed over domains with complex geometry, and directly generalize to both parametric forward and inverse problems. Notably, one can simply incorporate a data-matching term in the loss function to solve inverse problems with PINNs, while classical adjoint-based optimization typically requires many evaluations of a costly

[†]This work was partially completed during an internship at Mitsubishi Electric Research Laboratories (MERL)
[*]Corresponding authors

forward model. Since their introduction, PINNs have successfully modeled various physical phenomena, including heat transfer [4], wave propagation [5], and fluid flows [6].

While classical numerical schemes come with convergence guarantees, PINNs rely on neural network training of a nonconvex loss function. Though some theory for PINNs has been developed [7, 8], in general no performance guarantees are available, and it is well-known that PINNs often struggle to converge when the underlying problem exhibits high-frequency and multiscale behavior. Inspired by classical techniques for solving multiscale PDEs [9], one promising direction for simplifying the PINN optimization involves performing domain decomposition. The core idea of domain decomposition is to partition the spatio-temporal domain into many smaller subdomains and to solve a relatively simple problem within each subdomain, while also ensuring consistent interface conditions between subdomains are satisfied. Several PINN-based domain decomposition methodologies have already been developed [10, 11, 12, 13, 14, 15, 16]; see [17] for a comprehensive review.

Many of these methods involve training a separate neural network within each subdomain and enforcing interface conditions as a soft constraint, which requires careful construction of the loss function and can result in discontinuities of the learned PDE solution. Instead, the finite-basis physics-informed neural network (FBPINN) automatically enforces interface conditions by a suitable PINN solution ansatz which defines subdomains through compactly-supported window functions, each multiplied with a subnetwork responsible for learning the solution within the corresponding subdomain [18, 19]. This method simplifies the training loss since it automatically ensures continuity of the predicted PDE solution. FBPINNs have shown significant promise for learning the solution of high-frequency and multiscale PDEs where conventional PINN approaches struggle. Due to individual subdomain normalization, each subnetwork is capable of representing a local solution using relatively few parameters. Moreover, the evaluation of all subnetworks can be vectorized, which makes the approach highly efficient at learning the unknown PDE solution.

In the FBPINN framework, the domain decomposition is determined before training begins, is fixed throughout training, and the only tunable parameters are the subnetworks' weights and biases. While adaptive mesh refinement has played a critical role in classical numerical methods for solving PDEs [20, 21, 22], there has not yet been extensive work on extending these ideas to PINN-based domain decomposition methodologies. For multiscale problems with local and sharp gradients, it is unlikely that a uniform domain decomposition is optimal for learning the underlying PDE solution. In particular, a decomposition which refines itself in areas of high PDE residual, i.e., where the solution is challenging to learn, may lead to faster convergence.

While the implementation of adaptive mesh strategies for classical numerical methods can be complex and often requires numerous forward solves at different mesh resolutions, the simple nature of PINN training and flexibility of the FBPINN solution ansatz (8) opens the door for dynamic PINN-based domain decomposition strategies which adapt throughout training. Motivated by this observation, we propose adaptive-basis physics-informed neural networks (AB-PINNs), which have the following key features.

1. **Local adaptivity of subdomains:** Each subdomain is parameterized as the support of a radial basis function (RBF) composed with a tunable affine transformation. Thus, during training the center and spread of each subdomain evolve to fit the geometry of the learned solution; see Section 3.1 for further discussion.
2. **Residual-based addition of new subdomains:** During training, we introduce new adaptive subdomains on-the-fly in regions of high PDE residual. This provides additional expressive power in regions where the solution is challenging to represent and prevents the PINN from getting stuck at a local minima; see Section 3.2 for further discussion.
3. **Global network:** Our model also makes use of a global network defined over the entire problem domain. The global network helps learn the low frequency behavior of the PDE and ensures that the solution can still be learned in regions of the domain where subnetworks are not defined.

All three features are novel in the context of PINNs. While various works have studied the use of RBF networks with adaptive basis function parameters for solving PDEs [23, 24, 25, 26], these approaches do not consider the use of subnetworks to improve expressivity of the model within the support of each basis function. Moreover, to the best of our knowledge, our framework is the first to consider the addition of new subdomains during training to target regions of high residual loss, thereby merging classical mesh refinement strategies with the flexibility of deep learning. In static decompositions such as FBPINNs, the optimal number and placement of subdomains are unknown without prior knowledge of the underlying solution, often requiring many trials with different subdomain initializations to obtain an accurate model. In contrast, both the AB-PINN domain decomposition and number of subdomains adapt throughout training, which can significantly reduce time spent on hyperparameter tuning and lead to faster convergence of PDE residuals by several orders of magnitude. Finally, we note that the popular DeepSeekMoE large language model similarly merges global networks capturing common knowledge with individual subnetworks responsible for more specialized tasks [27].

The rest of the paper is structured as follows. Section 2 reviews background material including PINNs, FBPINNs, and related works. Section 3 then introduces our proposed method, AB-PINNs, in full detail. Comprehensive numerical tests which demonstrate the effectiveness of our method are presented in Section 5. We present theoretical properties of AB-PINNs in Section 4. Finally, conclusions follow in Section 7.

2 Background

In this section, we present background material which provides the context and motivation for our proposed method of adaptive domain decomposition in PINNs. In Section 2.1, we introduce the standard PINN framework for solving partial differential equations. Sections 2.2, 2.3, and 2.4 then explore several innovations, e.g., adaptive sampling, domain decompositions, and modified architectures, which have since been used to improve the accuracy and convergence of the standard PINN.

2.1 Physics-Informed Neural Networks (PINNs)

We now introduce the standard PINN framework for solving general partial differential equations of the form

$$\mathcal{D}[u](x) = f(x), \qquad x \in \Omega, \tag{1}$$
$$\mathcal{B}[u](x) = g(x), \qquad x \in \Gamma \subseteq \partial\Omega, \tag{2}$$

where $\Omega \subseteq \mathbb{R}^d$ is a bounded domain, $\partial\Omega$ is the boundary, $\mathcal{D}$ is a differential operator, $\mathcal{B}$ is a boundary operator, f is a forcing function, and g is a boundary function. A PINN represents a solution $u : \Omega \to \mathbb{R}$ of the system (1)-(2) by a neural network $u_\theta : \Omega \to \mathbb{R}$, with tunable parameters $\theta \in \Theta \subseteq \mathbb{R}^p$. The network is then trained via gradient-based methods to reduce a suitable loss that encodes the interior PDE constraint (1), as well as the boundary condition (2). While the original PINN formulation [3] enforces the boundary conditions as a soft constraint incorporated in the loss function, in many situations the boundary condition (2) can be directly enforced as a hard constraint [28, 29, 30], which prevents the need to balance multiple terms in the loss and helps accelerate convergence. By leveraging knowledge of the boundary operator $\mathcal{B}$, in certain situations one may construct a suitable constraining operator Ψ and instead regard $\hat{u}_\theta = \Psi[u_\theta]$ as the ansatz for solving the PDE, which automatically satisfies $\mathcal{B}[\hat{u}_\theta](x) = g(x)$, for all $x \in \Gamma \subseteq \partial\Omega$; see Section 5 for several examples of constraining operators. The resulting optimization problem and physics-informed loss function are then given by

$$\min_{\theta\in\Theta} \mathcal{L}(\theta), \qquad \mathcal{L}(\theta) = \frac{1}{\mathrm{vol}(\Omega)} \int_\Omega \mathcal{R}_\theta(x)\,\mathrm{d}x, \qquad \mathcal{R}_\theta(x) := |\mathcal{D}[\hat{u}_\theta(x)] - f(x)|^2. \tag{3}$$

In (3), $|\cdot|$ denotes the Euclidean 2-norm, and $\mathcal{R}_\theta : \Omega \to \mathbb{R}$ is the so-called *PDE residual*. The residual $\mathcal{R}_\theta(x)$ is computed via automatic differentiation and determines how well the solution ansatz $\hat{u}_\theta$ satisfies (1) at a given point $x \in \Omega$. When the loss $\mathcal{L}(\theta)$ defined by (3) is minimized to zero it holds that $\mathcal{R}_\theta(x) = 0$ for all $x \in \Omega$, thus ensuring that $\hat{u}_\theta$ solves (1)-(2). In practice, the integral in (3) cannot be analytically computed and one relies on the Monte–Carlo approximation

$$\hat{\mathcal{L}}(\theta) = \frac{1}{N} \sum_{k=1}^{N} \mathcal{R}_\theta(x_k), \tag{4}$$

where $\{x_k\}_{k=1}^N$ are *collocation points* sampled i.i.d. from $\mathrm{Unif}(\Omega)$, the uniform distribution over Ω. Using fixed collocation points throughout training may lead to overfitting the PDE residuals, and thus it is best practice to randomly resample the collocation points throughout training [31].

While the PINN framework described above is conceptually straightforward and simple to implement, in practice the approach can struggle to converge when the underlying differential equation exhibits complex, multiscale behavior. In recent years, researchers have identified several key modifications to the PINN training pipeline which significantly improve the solution accuracy; see [31] for a review of these developments.

2.2 Adaptive Sampling and Reweighted Residuals

To improve the convergence of PINNs, several works utilize reweighted formulations of the loss (3). The key idea behind these strategies is to focus training on regions of the spatio-temporal domain where the PINN may be struggling to represent the underlying solution. More specifically, given a strictly positive density $\pi : \Omega \to \mathbb{R}$ one can instead consider the loss

$$\mathcal{L}_\pi(\theta) = \int_\Omega \mathcal{R}_\theta(x)\pi(x)\,\mathrm{d}x. \tag{5}$$

Clearly, if $\mathcal{L}_\pi(\theta) = 0$, then $\hat{u}_\theta$ still solves (1)-(2). Writing down the Monte–Carlo approximation of (5) leads to a new finite-sample loss:

$$\hat{\mathcal{L}}_\pi(\theta) = \frac{1}{N}\sum_{k=1}^{N} \mathcal{R}_\theta(x_k), \qquad x_k \sim \pi. \tag{6}$$

The strategy (6), which involves drawing the collocation points from the distribution π, has motivated numerous works to study optimal strategies for selecting π in practice [32, 33, 34, 35]. Most of these focus on adaptive strategies in which π is determined using the current solution ansatz $\hat{u}_\theta$ and PDE residual $\mathcal{R}_\theta$. A simple approach studied in [32] involves selecting $\pi(x) \propto \mathcal{R}_\theta(x) + c$, the motivation being that more collocation points should be sampled in regions of high PDE residual, thereby focusing the PINN to train where the underlying solution is challenging to represent.

Another class of approaches avoids the need to directly sample from the distribution π and instead assigns a weight $w_k > 0$ to each collocation point, i.e.,

$$\hat{\mathcal{L}}_w(\theta) = \frac{1}{N}\sum_{k=1}^{N} w_k \mathcal{R}_\theta(x_k), \qquad x_k \sim \text{Unif}(\Omega). \tag{7}$$

Note that if $w_k = \pi(x_k) \cdot \text{vol}(\Omega)$, then (7) also approximates (5) via importance sampling. Multiple approaches have been developed to assign pointwise weights w_k, either through direct gradient ascent or by defining or learning a residual-to-weight mapping, in order to guide PINN training toward regions exhibiting large PDE residuals [36, 37, 38, 39]. A different line of work [40] focuses on choosing adaptive residual weights which enforce temporal causality, such that the solution is learned sequentially over time starting from the initial condition.

We remark that our proposed method for adaptive domain decomposition primarily involves changing the base neural network architecture and is compatible with any adaptive sampling or residual reweighting scheme, which are expected to improve performance.

2.3 Finite-Basis Physics Informed Neural Networks

We now describe the finite-basis physics informed neural network (FBPINN) [18] in detail. FBPINNs are a "soft" domain decomposition strategy for training PINNs, which does not involve learning interface conditions between subdomains during optimization. In particular, the FBPINN ansatz is given by a summation of local solutions, each consisting of a product between a compactly supported window function and a subnetwork composed with a normalizing transformation mapping the support of the window function into reasonable inputs for effective subnetwork training. The support of each window function thereby defines a subdomain and each subnetwork is primarily responsible for learning the solution within its corresponding subdomain. In regions where subdomains overlap, multiple subnetworks contribute to the solution.

In particular, the FBPINN solution ansatz is given by

$$u_\theta(x) = \sum_{i=1}^{K} \phi_i(x) \texttt{NN}_i(r_i(x); \theta_i), \qquad x \in \Omega, \tag{8}$$

where $\Omega \subseteq \mathbb{R}^d$ is the problem domain, K is the number of subdomains, ϕ_i is a smooth (effectively) compactly supported *window function* defining the support of each subdomain, the *subnetwork* $\texttt{NN}_i$ is a universal function approximator, $\theta_i \in \Theta \subseteq \mathbb{R}^p$ comprise its tunable parameters, and r_i is a coordinate transformation normalizing the support of ϕ_i to a standard input range for effective neural network training. In (8), the window functions ϕ_i should be chosen to have overlapping supports and cover the full domain Ω. The subnetwork $\texttt{NN}_i$ is primarily responsible for learning the solution within the support of ϕ_i, and in any region where subdomains overlap the solution is given by the summation of multiple subnetwork contributions. For a visualization of the overlapping FBPINN subdomains, see [18, Figure 2].

In Section 3, we make the parameters of the window functions ϕ_i and coordinate transformations r_i tunable, enabling each subdomain to adapt to the geometry of the underlying PDE during training. Second, we modify the decomposition (8) on-the-fly by introducing new subdomains during training in regions of high PDE residual. We show in Section 5 that both of these modifications significantly improve the convergence of the FBPINN framework and can substantially reduce the number of parameters needed to train an accurate FBPINN model for multiscale problems.

2.4 Modified Network Architecture

Several works have investigated how modifying the base neural network architecture can improve the performance of PINNs. While it is standard in many cases to use a standard multi-layer perceptron (MLP) with Glorot initialization

[41], modified MLPs [42], Kolmogorov Arnold Networks (KANs) [43], and SIREN [44], are variations which have exhibited improved performance in certain situations. Similar to the FBPINN setup, our approach can be viewed as a new neural network architecture used to train PINNs. While in this work, we parameterize each subnetwork $\texttt{NN}_i$ as an MLP, any of the previously mentioned architectures are also compatible with our framework.

2.5 Related Work

Three notable works [45, 46, 47] study adaptive "soft" domain decompositions for training PINNs using the mixture-of-experts framework [48], in which a coordinate-dependent gating network parametrizes a weighted average of the subnetworks' outputs. Although this approach seeks to induce an implicit domain decomposition, in practice, it is difficult to tune the gating network so that each subnetwork contributes meaningfully to the solution only over a finite subdomain. Achieving such locality typically requires either pretraining the gating network, adding sparsity-inducing regularization, or only activating a predefined number of subnetworks at any point ("top-k" routing), and remains an active area of research within the mixture-of-experts community [49, 50]. In contrast to these studies, our method employs effectively compactly-supported, tunable radial basis functions to define the domain decomposition. This design naturally enables (1) the introduction of new subdomains in regions of high PDE residual, and (2) the use of simple coordinate transformations for each subdomain which are crucial for efficient training of the associated subnetworks. Neither of these key capabilities is straightforward when the domain decomposition is implicitly parameterized by a gating network. A different direction uses a PINN-based residual to perform adaptive mesh refinement for a classical numerical PDE solver [51]. Finally, we note that several works have also studied neural networks with adaptive architectures to enhance PINN performance outside of the context of domain decomposition [52, 53, 54, 55].

3 Adaptive-Basis Physics-Informed Neural Networks (AB-PINNs)

In this Section we describe our proposed AB-PINN model in full detail. Section 3.1 first describes how we parameterize each trainable subdomain. In Section 3.2, we discuss our method for introducing new subdomains in regions of high PDE residual. Section 3.3 then discusses how AB-PINNs are made compatible with Fourier embeddings to enforce periodicity of the predicted PDE solution, which is required specifically for the PDEs considered in Sections 5.2.2, 5.3.3, and 5.3.4. Section 3.4 goes over some practical implementation details for the AB-PINN model.

3.1 Learnable Subdomains

We now describe the base architecture of AB-PINNs. First, we modify (8) such that the window function ϕ_i is comprised of tunable parameters, which in turn define the coordinate transformation r_i. While there are many ways to parameterize both the window functions ϕ_i and resulting coordinate transformations r_i, we choose a simple representation by constructing the functions with a tunable affine transformation. In particular, we define some reference window function $\psi : \Omega \to \mathbb{R}$ and define each tunable window function and coordinate transformation by

$$\phi_i(x) := (\psi \circ r_i)(x), \qquad r_i(x) := L_i^\top (x - \mu_i), \qquad L_i \in \mathbb{R}^{d\times d}, \qquad \mu_i \in \mathbb{R}^d. \tag{9}$$

We choose L_i as a lower-triangular matrix with strictly positive diagonal, where the parameters of each $L_i \in \mathbb{R}^{d\times d}$ and $\mu_i \in \mathbb{R}^d$ are learnable. For notational simplicity, we write β_i to denote this vector of tunable parameters corresponding to r_i. Notably, when ψ is chosen as a *radial basis function* (RBF), i.e., its output depends only on the norm of the input vector through a function $\hat{\psi} : [0, \infty) \to \mathbb{R}$, we have that

$$\phi_i(x) = \hat{\psi}(|r_i(x)|) = \hat{\psi}(\|x - \mu_i\|_{M_i}), \qquad M_i = L_i L_i^\top \in \mathbb{R}^{d\times d}, \qquad \|x\|_M := \sqrt{x^\top M x}. \tag{10}$$

The window functions ϕ_i thus become RBFs centered at μ_i and stretched along the eigenvectors of M_i. Importantly, our parameterization (9) can represent any shift μ_i, as well as stretches induced by any norm of the form $\|\cdot\|_{M_i}$. Indeed, the Cholesky decomposition guarantees that any symmetric positive-definite matrix M_i admits a unique decomposition of the form $L_i L_i^\top$ where L_i is a lower-triangular matrix with strictly positive diagonal entries.

Remark 3.1. *Consider the case where $\psi(x) = \exp(-|x|^2/2)$. Then, by* (10) *we have*

$$\phi_i(x) = \exp\left(-\frac{1}{2}(x-\mu_i)^\top L_i L_i^\top (x-\mu_i)\right) = \exp\left(-\frac{1}{2}(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\right), \quad \Sigma_i := \left(L_i L_i^\top\right)^{-1} \in \mathbb{R}^{d\times d},$$

which is proportional to a multivariate Gaussian distribution with mean μ_i and covariance Σ_i, both of which are fully tunable during training. In this setup, the coordinate transformation $r_i : \Omega \to \Omega$ is a so-called whitening transformation that maps samples from $\mathcal{N}(\mu_i, \Sigma_i)$ to samples from $\mathcal{N}(0, I)$.

In practice, while one can choose any radial basis function ψ, it is helpful to select a master window function ψ such that the support of ψ contains inputs in the range that is best-suited for neural network training, e.g., samples with mean zero and unit variance. By selecting ψ in this way, the coordinate transformation r_i maps the support of the window function ϕ_i into this range, thus providing $\texttt{NN}_i(r_i(x))$ the best capability to represent the local solution on the support of ϕ_i. For example, for a one-dimensional domain, choosing ψ to be compactly supported on $[-1, 1]$ automatically ensures that r_i normalizes the support of the window function ϕ_i to $[-1, 1]$, which corresponds to the strategy used in [18]. In Figure 4, we compare the performance of several AB-PINN and FBPINN models with different choices of ψ.

Given that the window functions ϕ_i are now fully adaptive, one cannot guarantee that their support will entirely cover the domain Ω during training. To prevent situations in which parts of the domain are left outside the effective support of the window functions, and to promote the propagation of information between sub-domains, we also include a fixed global network as part of our parameterization. This network helps learn the low-frequency behavior of the unknown PDE solution and can also reduce the number of subdomains necessary for successful training.

Everything together, we can write our base architecture for the AB-PINN as

$$u(x;\theta,\beta) = \texttt{NN}_0(x;\theta_0) + \sum_{i=1}^{K} \phi_i(x;\beta_i)\texttt{NN}_i(r_i(x;\beta_i);\theta_i). \tag{11}$$

Note that the only differences between (11) and (8) are the parameterization of the window functions ϕ_i and coordinate transformations r_i, as well as the addition of the global network $\texttt{NN}_0$.

3.2 Residual-Based Subdomain Addition

While (11) constitutes the base AB-PINN architecture, we also modify (11) on-the-fly during training to introduce new subdomains in regions of high PDE residual. This involves constructing a new window function ϕ_{K+1} with parameters β_{K+1} and a new subnetwork $\texttt{NN}_{K+1}$ with parameters θ_{K+1}. While the subnetwork parameters θ_{K+1} are initialized according to a standard Glorot scheme [41], the choice of β_{K+1}, i.e., the center and spread of the new subdomain, plays a crucial role in the PINN training. First, the location of the new subdomain is adaptively determined as $\mu_{K+1} \sim \pi$, where π is the distribution proportional to the current PDE residual $\mathcal{R} = \mathcal{R}_\theta$; see (3). In practice, we approximate the measure π defined by

$$\pi(B) = \frac{\int_B \mathcal{R}(x)\,\mathrm{d}x}{\int_\Omega \mathcal{R}(x)\,\mathrm{d}x}, \qquad \forall B \subseteq \Omega \quad \text{measurable} \tag{12}$$

via the empirical refinement measure

$$\pi_N := \sum_{j=1}^{N} w_j\,\delta_{x_j}, \qquad w_j := \frac{\mathcal{R}(x_j)}{\sum_{\ell=1}^{N}\mathcal{R}(x_\ell)}, \qquad x_1,\dots,x_N \sim \text{Unif}(\Omega)\ \text{ i.i.d.}, \tag{13}$$

where $x_1, \dots, x_N$ are the same set of samples utilized to evaluate the loss function for updating θ and β. Thus, the center of the new subdomain is likely to be placed in a region of high PDE residual, where the current solution ansatz is struggling to represent the true solution. Note the similarity between this strategy and the adaptive sampling strategies discussed in Section 2.2. The spread of the new subdomain, determined by the lower-triangular matrix L_{K+1}, is simply a user-specified hyper-parameter which indicates the fraction of the domain the new subnetwork is responsible for. After the new subdomain is placed, both its center and spread continue to dynamically evolve during training, as described in Section 3.1.

Throughout the remainder of the paper, we will write AB-PINN+ as shorthand for an AB-PINN model which uses residual-based subdomain addition. A pseudocode for the AB-PINN+ training procedure and residual-based subdomain addition is provided in Algorithm 1.

In our numerical experiments, we observe that the addition of new subdomains in regions of high PDE residual can be useful for avoiding local minima and accelerating convergence; see Section 5.3.3. Moreover, in certain situations, the domain decomposition learned via residual-based subdomain addition can provide significant benefits over a decomposition arrived at by initializing all subdomains at the start of training; see Section 5.3.2. Finally, notable computational savings are gained by introducing the subdomains throughout training, rather than starting with all subdomains at the beginning of training.

3.3 Enforcing Periodicity in AB-PINNs

As with any PINN architecture, a suitable constraining operator can be applied to enforce a Dirichlet or Neumann boundary or initial condition as a hard constraint in AB-PINNs; see Section 2.1. In several of our numerical experiments

Algorithm 1: AB-PINN+ Training Procedure

Initialize: Model $u(x;\theta,\beta)$ following (11) with K adaptive basis functions.

Repeat until convergence:

1. Sample N collocation points $\{x_k\}_{k=1}^N$.
2. Evaluate the residual loss:
$$\mathcal{L}(\theta,\beta) = \frac{1}{N}\sum_{k=1}^{N} \left|f(x_k) - \mathcal{D}[\hat{u}(x_k;\theta,\beta)]\right|^2.$$
3. Compute gradients $\nabla_\theta \mathcal{L}(\theta,\beta)$ and $\nabla_\beta \mathcal{L}(\theta,\beta)$, and update θ and β.
4. **If subdomain addition criterion is met:**
   (a) Sample $\mu_{K+1} \sim \pi_N$, where π_N approximates the residual distribution; see (13).
   (b) Define L_{K+1} and initialize θ_{K+1} using a Glorot scheme.
   (c) Update the model:
$$u(x;\theta,\beta) \leftarrow u(x;\theta,\beta) + \phi_{K+1}(x;\beta_{K+1})\,\mathtt{NN}_{K+1}(r_{K+1}(x;\beta_{K+1});\theta_{K+1})$$
   (d) Increment $K \leftarrow K+1$.

the underlying PDE is also subject to periodic spatial boundary conditions. While these boundary conditions can be imposed as a soft constraint in the loss function [3], it is also possible to use an input-coordinate Fourier embedding to enforce periodicity as a hard constraint [31]. We illustrate how this strategy is made compatible with AB-PINNs in a simple one-dimensional case that generalizes to higher dimensions.

For a one-dimensional spatial input $x \in [-1,1]$, the Fourier embedding is given by the map $x \mapsto (\cos(\pi x), \sin(\pi x)) \in \mathbb{S}^1 \subseteq \mathbb{R}^2$, where $\mathbb{S}^1$ is the unit circle. By defining a neural network that acts on the embedded space $u_\theta : \mathbb{R}^2 \to \mathbb{R}$, one can ensure that the function $v_\theta : [-1,1] \to \mathbb{R}$ defined by $v_\theta(x) = u_\theta(\cos(\pi x), \sin(\pi x))$ satisfies the periodic condition $v_\theta(-1) = v_\theta(1)$. In our numerical experiments, we model u_θ as an AB-PINN defined on $\mathbb{R}^2$ with the centers $\mu_i \in \mathbb{R}^2$ constrained to remain inside $\mathbb{S}^1$ via the Fourier embedding, while the lower-triangular matrices $L_i \in \mathbb{R}^{2\times 2}$ are parameterized as before. Thus, the AB-PINN subdomain decomposition occurs within the embedded coordinates rather than in the original spatial domain, automatically ensuring periodicity of the window functions ϕ_i in the original domain.

3.4 Additional Implementation Details

During training of the AB-PINN architecture (11), all window functions $\{\phi_i\}_{i=1}^K$, coordinate transformations $\{r_i\}_{i=1}^K$, and subnetworks $\{\mathtt{NN}_i\}_{i=1}^K$ are simultaneously evaluated in a single vectorized pass, which makes the Ansatz (11) practical for complex problems consisting of many subdomains. One of the primary reasons FBPINNs are so efficient is that each subnetwork only needs to be evaluated on collocation points falling within its respective subdomain. While for simplicity we evaluate each AB-PINN subnetwork on all collocation points, existing work from the mixture-of-experts community can provide methods for selective collocation point evaluation which is expected to improve efficiency [56, 57].

Motivated by the observation that sometimes small perturbations of a window function can lead to significant changes in the predicted PDE solution, we also find it helpful to manually freeze the learning rates of the tunable window function parameters μ_i and L_i after a fixed number of iterations have elapsed. This effectively separates the AB-PINN training into two distinct phases:

1. **Learning the solution features:** The window functions adapt, alongside all subnetwork and global network parameters, to learn the distribution of the underlying solution features over the domain.
2. **Fine-tuning:** The window function parameters are frozen, and the subnetworks and global networks continue training until convergence.

We have empirically observed that the addition of the fine-tuning stage improves the AB-PINN training stability and prediction accuracy.

4 Theoretical Properties of AB-PINNs

We present several basic theoretical properties of our AB-PINN architecture and residual-driven refinement strategy. In particular, we show that the AB-PINN ansatz (11) enjoys universal approximation properties in appropriate function spaces and that the insertion of a subdomain produces a bounded perturbation of the training loss. We also discuss convergence of the empirical refinement measure π_N introduced in Section 3.2 as $N \to \infty$.

Throughout this subsection, we let $m, d \in \mathbb{N}$ be fixed. For an open subset $\Omega \subseteq \mathbb{R}^d$, we write $C^m(\Omega)$ to denote the space of real-valued functions on Ω with continuous derivatives up to order m. For a compact subset $F \subseteq \Omega$ and $u \in C^m(\Omega)$ we define the seminorm

$$\|u\|_{m,F} = \max_{|\alpha| \leq m} \sup_{x \in F} |D^\alpha u(x)|. \tag{14}$$

Following [58], we say that a set of functions $\mathcal{F} \subseteq C^m(\Omega)$ is *dense on compact sets in* $C^m(\Omega)$ if for any $v \in C^m(\Omega)$, any compact subset $F \subseteq \Omega$, and any $\varepsilon > 0$, there exists some $u \in \mathcal{F}$ satisfying $\|u - v\|_{m,F} < \varepsilon$. We will assume that $\psi \in C^m(\mathbb{R}^d)$, that $r_i : \mathbb{R}^d \to \mathbb{R}^d$ is an affine map parameterized by β_i, and that the number of basis terms $K \in \mathbb{N}$ is fixed. We now define the *adaptive basis class* $\mathcal{V}_K \subseteq C^m(\mathbb{R}^d)$ as

$$\mathcal{V}_K = \bigcup_{n=1}^{\infty} \mathcal{V}_K^{(n)}, \qquad \mathcal{V}_K^{(n)} = \left\{ \sum_{i=1}^{K} \phi_i(\cdot;\beta_i)\, \texttt{NN}_i(r_i(\cdot;\beta_i);\theta_i) : \sum_{i=1}^{K} |\theta_i| = n \right\}, \tag{15}$$

where $\phi_i = \psi \circ r_i$, and thus $\phi_i \in C^m(\mathbb{R}^d)$. We now aim to prove that the adaptive basis class $\mathcal{V}_K$ is dense on compact sets in $C^m(\Omega)$ for an open domain $\Omega \subseteq \mathbb{R}^d$. Towards this, we first establish the following lemma which provides bounds on the C^m norm under pointwise multiplication and composition by affine transformations.

Lemma 4.1. *Let $u, v \in C^m(\mathbb{R}^d)$, let $F \subseteq \mathbb{R}^d$ be compact, and let $r : \mathbb{R}^d \to \mathbb{R}^d$ be an affine map. Then, there exist constants $c_1 > 0$ depending only on m, and $c_2 = c_2(r) > 0$ depending only on m, d, and r, such that the following hold:*

1. $\|uv\|_{m,F} \leq c_1 \|u\|_{m,F} \|v\|_{m,F}$.
2. $\|u \circ r\|_{m,F} \leq c_2 \|u\|_{m,r(F)}$

Proof. The first bound is a direct consequence of Leibniz's product rule. The second follows from the chain rule by expressing

$$D^\alpha (u \circ r) = \sum_{|\gamma| = |\alpha|} c_{\alpha,\gamma} D^\gamma u \circ r,$$

where $c_{\alpha,\gamma}$ are constants depending only on the affine transformation r. Thus,

$$\begin{aligned} \|u \circ r\|_{m,F} \leq \max_{|\alpha| \leq m} \sup_{x \in F} \sum_{|\gamma| = |\alpha|} |c_{\alpha,\gamma}| |D^\gamma u(r(x))| &= \max_{|\alpha| \leq m} \sup_{x \in r(F)} \sum_{|\gamma| = |\alpha|} |c_{\alpha,\gamma}| |D^\gamma u(x)| \\ &\leq c_2 \max_{|\alpha| \leq m} \sup_{x \in r(F)} |D^\alpha u(x)| \\ &= c_2 \|u\|_{m,r(F)}, \end{aligned}$$

where $c_2 > 0$ depends on m, d, and r. □

Theorem 4.2. *Assume that $\Omega \subseteq \mathbb{R}^d$ is open, that $\psi \in C^m(\mathbb{R}^d)$, and that each subnetwork $\texttt{NN}_i$ comprises one hidden layer and uses a nonconstant, bounded, and C^m activation. Further assume*

$$S(x) = \sum_{i=1}^{K} \phi_i(x;\beta_i) > 0, \qquad \forall x \in \Omega \tag{16}$$

Then, $\mathcal{V}_K$ is dense on compact sets in $C^m(\Omega)$.

Proof. Let $u \in C^m(\Omega)$, let $F \subseteq \Omega$ be compact, and fix $\varepsilon > 0$. For each $1 \leq i \leq K$ we now define $v_i \in C^m(r_i(\Omega))$ by setting

$$v_i(x) = \frac{(u \circ r_i^{-1})(x)}{(S \circ r_i^{-1})(x)}, \qquad x \in r_i(\Omega).$$

Now, note by [58, Theorem 3] that for each $1 \le i \le K$ there exists a subnetwork $\mathtt{NN}_i = \mathtt{NN}_i(\cdot;\theta_i)$ such that

$$\|\mathtt{NN}_i - v_i\|_{m,r_i(F)} < \frac{\varepsilon}{Kc_1c_2(r_i)\max\{1, \|\phi_i\|_{m,F}\}},$$

where $c_1, c_2(r_i) > 0$ are the constants appearing in Lemma 4.1. Now, observe that

$$\begin{aligned}
\left\| \sum_{i=1}^K \phi_i \mathtt{NN}_i \circ r_i - u \right\|_{m,F} &= \left\| \sum_{i=1}^K \phi_i \mathtt{NN}_i \circ r_i - \sum_{i=1}^K \phi_i \frac{u}{S} \right\|_{m,F} \\
&= \left\| \sum_{i=1}^K \phi_i \left(\mathtt{NN}_i \circ r_i - \frac{u}{S} \right) \right\|_{m,F} \\
&\le \sum_{i=1}^K c_1 \|\phi_i\|_{m,F} \left\| \mathtt{NN}_i \circ r_i - \frac{u}{S} \right\|_{m,F} \qquad (17) \\
&\le \sum_{i=1}^K c_1 c_2(r_i) \|\phi_i\|_{m,F} \|\mathtt{NN}_i - v_i\|_{m,r_i(F)} \qquad (18) \\
&< \varepsilon,
\end{aligned}$$

where Lemma 4.1 was used to obtain (17) and (18). Thus, $\mathcal{V}_K$ is dense on compact sets in $C^m(\Omega)$. □

We also note that if $\Psi : C^m(\mathbb{R}^d) \to C^m(\mathbb{R}^d)$ is a continuous constraining operator, then the assumptions of Theorem 4.2 imply that the constrained adaptive basis class $\Psi(\mathcal{V}_K)$ is dense on compact sets in the constrained solution space $\Psi(C^m(\Omega))$. To extend universal approximation results to the Sobolev space $H^m(\Omega)$, one typically requires additonal assumptions on Ω, such as the *segment property* [58, 59].

Theorem 4.2 shows that adaptivity does not come at the expense of approximation power. For the Gaussian window functions used in most of our experiments, the positivity condition on S is automatically satisfied on bounded domains. In this case, the adaptive-basis portion of the AB-PINN model is already dense on compact sets $C^m(\Omega)$ and the global network in (11) is not needed to recover expressivity; rather, its main role is to improve optimization, propagate information between subdomains, and capture large-scale solution structure. On the other hand, if one were to select window functions with compact support, then the global network would be necessary for the AB-PINN model to be dense on compact sets in $C^m(\Omega)$.

We next show that adding a new adaptive subdomain is a controlled perturbation of the residual loss. For a sufficiently regular function v and a bounded, open domain $\Omega \subseteq \mathbb{R}^d$ define

$$\mathcal{G}[v] := \mathcal{D}[v] - f, \qquad \mathcal{L}(v) := \frac{1}{\mathrm{vol}(\Omega)} \int_\Omega |\mathcal{G}[v]|^2 \, \mathrm{d}x.$$

In the following proposition, we assume the operator $\mathcal{G} : H^m(\Omega) \to L^2(\Omega)$ is (c, α)-Hölder for constants $c, \alpha > 0$, i.e.,

$$\|\mathcal{G}[u_1] - \mathcal{G}[u_2]\|_{L^2(\Omega)} \le c\|u_1 - u_2\|^\alpha_{H^m(\Omega)}$$

for all $u_1, u_2 \in H^m(\Omega)$. This provides us with a notion of stability for the operator $\mathcal{G}$, which is needed for controlling perturbations in the training loss when new subdomains are added.

Proposition 4.3 (Stability of adaptive basis insertion)**.** *Assume that Ω is open and bounded, and that the map $\mathcal{G} : H^m(\Omega) \to L^2(\Omega)$ is (c, α)-Hölder. Suppose further that $v \in H^m(\Omega)$ and set $\tilde{v} = v + \delta$ for some $\delta \in H^m(\Omega)$. It then holds that*

$$\mathcal{L}(\tilde{v}) \le \left(\sqrt{\mathcal{L}(v)} + \frac{c}{\sqrt{\mathrm{vol}(\Omega)}} \|\delta\|^\alpha_{H^m(\Omega)} \right)^2.$$

Proof. Notice that

$$\begin{aligned}
\sqrt{\mathcal{L}(\tilde{v})} - \sqrt{\mathcal{L}(v)} &= \frac{1}{\sqrt{\mathrm{vol}(\Omega)}} \left(\|\mathcal{G}[\tilde{v}]\|_{L^2(\Omega)} - \|\mathcal{G}[v]\|_{L^2(\Omega)} \right) \\
&\le \frac{1}{\sqrt{\mathrm{vol}(\Omega)}} \|\mathcal{G}[v + \delta] - \mathcal{G}[v]\|_{L^2(\Omega)} \\
&\le \frac{c}{\sqrt{\mathrm{vol}(\Omega)}} \|\delta\|^\alpha_{H^m(\Omega)}.
\end{aligned}$$

Rearranging the inequality above yields the desired result. □

We remark that a similar conclusion also holds when one assumes that the residual map $\mathcal{G}[\cdot]$ is only locally Hölder, where in that case the constants (c, α) depend on the neighborhood of $H^m(\Omega)$.

To relate Proposition 4.3 directly to the proposed AB-PINN architecture, we will assume that a constraining operator of the form $\Psi[u] = h_1 u + h_2$ is applied where $h_1, h_2 \in H^m(\Omega)$. Thus, if we denote by $\hat{u}_K = h_1 u_K + h_2$ the constrained AB-PINN solution after K subdomains have been added, the constrained solution after inserting the $(K+1)$-st subdomain is

$$\underbrace{\hat{u}_{K+1}}_{\tilde{v}} = \underbrace{\hat{u}_K}_{v} + \underbrace{h_1 \mathtt{NN}_{K+1} \phi_{K+1}}_{\delta}. \tag{19}$$

Equation (19) relates the notation from Proposition 4.3 to the constrained AB-PINN ansatz, showing that the residual loss increase is bounded by the Sobolev norm of a perturbation depending solely on the newly added subdomain and the constraining ansatz.

Finally, we show convergence of the empirical refinement measure π_N as $N \to \infty$, demonstrating that our selection criteria for the location of new subdomains is consistent with the underlying residual distribution π.

Proposition 4.4. *Assume that $\mathcal{R} \geq 0$, $\mathcal{R} \in L^1(\Omega)$, $\int_\Omega \mathcal{R}\,\mathrm{d}x > 0$, and fix a bounded $g \in C(\Omega)$. Then, the convergence*

$$\lim_{N\to\infty} \int_\Omega g\,\mathrm{d}\pi_N = \int_\Omega g\,\mathrm{d}\pi$$

holds almost surely, where π is the residual measure (12) *and π_N is the empirical refinement measure* (13)*.*

Proof. Fix any bounded $g \in C(\Omega)$ and let $x_1, \ldots, x_N \sim \mathrm{Unif}(\Omega)$ be i.i.d. samples. Then,

$$\int_\Omega g\,\mathrm{d}\pi_N = \frac{\frac{1}{N}\sum_{j=1}^N \mathcal{R}(x_j) g(x_j)}{\frac{1}{N}\sum_{j=1}^N \mathcal{R}(x_j)} \xrightarrow{N\to\infty} \frac{\int_\Omega \mathcal{R}(x) g(x)\,\mathrm{d}x}{\int_\Omega \mathcal{R}(x)\,\mathrm{d}x} = \int_\Omega g\,\mathrm{d}\pi \qquad \text{a.s.} \tag{20}$$

The first equality in (20) follows from the definition of π_N (see (13)), the convergence as $N \to \infty$ is an almost sure property guaranteed by the law of large numbers, and the final equality follows from the definition of π (see (12)). □

Proposition 4.4 presents a well-known result concerning the convergence of importance sampling. While the law of large numbers is used to obtain the almost sure convergence (20), the central limit theorem can be used to analyze the root mean squared error of the approximation $\int_\Omega g\,\mathrm{d}\pi_N$, which achieves the standard Monte–Carlo rate of $\mathcal{O}(1/\sqrt{N})$; see [60].

5 Results

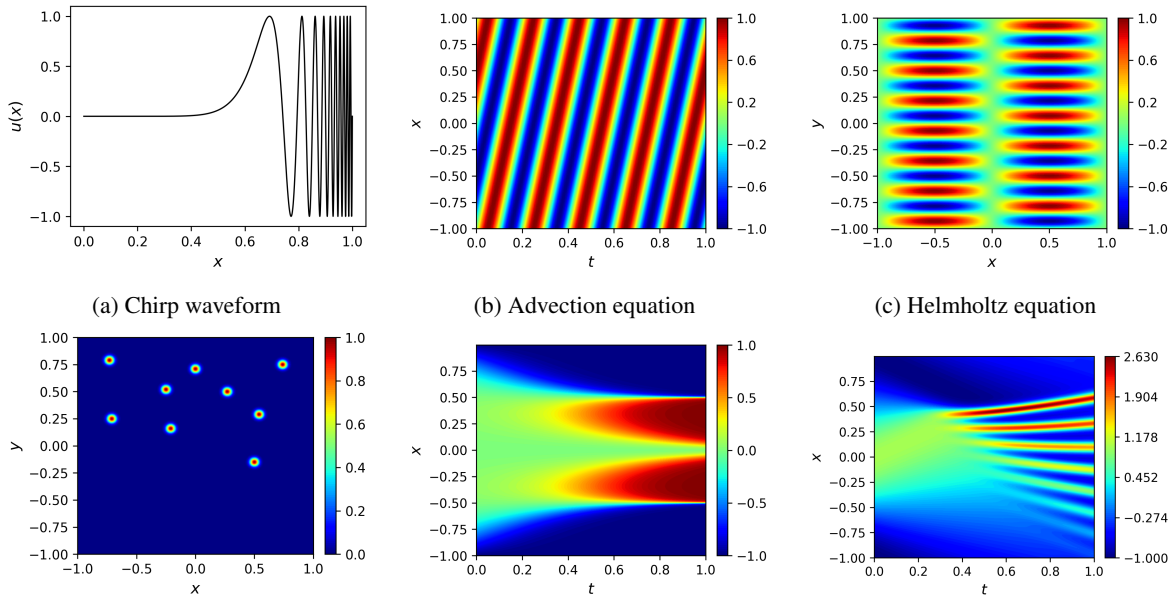
5.1 Overview of Numerical Experiments

In this section, we present comprehensive numerical results which showcase the effectiveness of AB-PINNs at solving complex multiscale differential equations compared to both standard PINNs and FBPINNs. The code implementation is publicly available at `https://github.com/merlresearch/ab-pinns`.

Figure 1 displays the ground truth reference solutions for the test cases we consider, including a multiscale chirp waveform, the advection equation, a Helmholtz equation, a locally forced Poisson equation, the Allen–Cahn equation, and the Korteweg–De Vries (KdV) equation. The reference solutions for the Allen–Cahn and KdV equations are approximated using a high-accuracy spectral method, while the solutions for the remaining differential equations are analytically available. Throughout, the error of all PINN models are quantified over 5 randomized trials using the L^1 norm

$$\frac{1}{|\mathrm{vol}(\Omega)|} \int_\Omega |u(x) - u_\theta(x)| dx.$$

We begin our numerical experiments in Section 5.2 by considering a fixed AB-PINN architecture without the subdomain addition described in Section 3.2. The only difference between this architecture and the FBPINN is the adaptivity of the window functions and the inclusion of a global network. Throughout our numerical experiments, the baseline FBPINNs are implemented as AB-PINNs with frozen subdomains and without the global network. Under this setup, we show in Table 1a that AB-PINNs can improve upon the accuracy of FBPINNs by several orders of magnitude when solving multiscale problems whose solution features are not uniformly distributed across the spatio-temporal domain. The only example we consider where the FBPINN has higher accuracy than the AB-PINN is in approximating a sine wave



(a) Chirp waveform (b) Advection equation (c) Helmholtz equation

(d) Forced Poisson equation (e) Allen–Cahn equation (f) Korteweg–De Vries equation

Figure 1: Reference solutions for the test cases studied in this paper.

solution. In this case, the uniform configuration of the FBPINN window functions is likely already optimal and any perturbation from this configuration learned by the AB-PINN may cause a disadvantage.

We then go on to show in Section 5.3 how the residual-based subdomain addition can be used to avoid local minima during training and deliver needed expressive power in regions of the spatio-temporal domain where the PINN is struggling to converge. In these experiments, we begin with a standard PINN architecture and add AB-PINN+ subdomains throughout training in regions of high PDE residual. The results for these experiments are summarized in Table 1b, revealing that adding subdomains during training steers the AB-PINN+ away from bad local minima and toward the correct solution. Moreover, we showcase the differences in the learned domain decomposition either when using the residual-based subdomain addition (AB-PINN+) or when relying solely on the local adaptivity of existing subdomains (AB-PINN). We note that some of the error distributions in Table 1b are heavy-tailed with high variance. In these cases, the mean and standard deviation alone are insufficient to characterize model performance across five runs. Thus, we also report median errors.

Across each test, we use the Adam optimizer and decay all learning rates on an exponential schedule to 1% of their initial value by the end of training [61]. The initial learning rate for all MLP weights and biases is always set to 10^{-3}. The collocation points are always randomly uniformly resampled each iteration, and all global networks and subnetworks considered are multi-layer perceptrons (MLPs) with the hyperbolic tangent activation function. Unless otherwise specified, the reference window function considered is always the Gaussian $\psi(x) = e^{-x^2/2}$; see Remark 3.1. The number of training steps for the two training phases (see Section 3.4), number of collocation points, number of subdomains, initial learning rates for the window function parameters μ_i and L_i, and size of each MLP are problem-dependent.

During optimization, the subnetwork parameters θ_i, subdomain centers μ_i, and tunable parameters of L_i corresponding to each subdomain, as well as the global network parameters θ_0, are all declared as separate parameter groups with their own optimizer and learning rates. When a new subdomain is added during training the associated tunable parameters are declared under a new optimizer, with learning rates that similarly decay on an exponential schedule, until all learnable window parameters are frozen at once for fine-tuning of the solution; see Section 3.4. Throughout training, the adaptive subdomain centers μ_i are restricted to remain inside the computational domain, and the diagonal of L_i is constrained to be positive through the absolute value function.

In many of our numerical experiments, we visualize the learned window functions ϕ_i in two-dimensions; see Figure 5 for example. Namely, we plot their centers μ_i with an x marker, we show a single thick contour line to indicate where the window function has reduced to half of its peak value, and we show a filled contour plot of the envelope $E(x) = \max_{1 \leq i \leq K} \phi_i(x)$ of all window functions. As a convention for the visualizations, we assign $E(x) = 0$ for

Method	Chirp waveform	Sine wave	Helmholtz equation	Advection equation
PINN	0.49 ± 0.16 (0.59)	0.50 ± 0.15 (0.48)	0.57 ± 0.35 (0.37)	0.47 ± 0.00 (0.47)
FBPINN	0.15 ± 0.20 (0.08)	$\mathbf{1.4{\times}10^{-4}} \pm 5.0{\times}10^{-5}$ $(\mathbf{1.2{\times}10^{-4}})$	0.31 ± 0.12 (0.31)	0.28 ± 0.04 (0.30)
AB-PINN	$\mathbf{0.02} \pm 0.03$ $(\mathbf{4.0{\times}10^{-4}})$	$4.8{\times}10^{-4} \pm 2.5{\times}10^{-4}$ $(4.2{\times}10^{-4})$	$\mathbf{0.003} \pm 0.001$ $(\mathbf{0.003})$	$\mathbf{0.06} \pm 0.05$ $(\mathbf{0.04})$

(a) Comparison where the AB-PINN only exhibits local adaptivity of a fixed number of subdomains.

Method	Allen–Cahn	Forced Poisson	KdV
PINN	0.008 ± 0.004 (0.007)	0.01 ± 0.00 (0.01)	0.06 ± 0.01 (0.05)
FBPINN	0.005 ± 0.001 (0.004)	0.08 ± 0.02 (0.07)	0.10 ± 0.01 (0.10)
AB-PINN+	$\mathbf{0.002} \pm 0.001$ $(\mathbf{0.002})$	$\mathbf{0.006} \pm 0.002$ $(\mathbf{0.006})$	$\mathbf{0.04} \pm 0.007$ $(\mathbf{0.04})$

(b) Comparison where new AB-PINN subdomains are added during training according to regions of high residual loss.

Table 1: Benchmarking AB-PINNs against the standard PINN architecture and FBPINNs in terms of the L^1 solution error. Table 1a shows results for AB-PINNs with local adaptivity of the window functions, whereas the results reported in Table 1b also consider the addition of new subdomains in regions of high PDE residual. Throughout, all models are trained 5 times with different random initializations and the mean, standard deviation, and median errors (in parenthesis) are reported.

the standard PINN. For problems with periodic boundary conditions, we instead visualize the composition of the window functions with the Fourier coordinate embedding. Even if the window functions are Gaussian in the embedded coordinates, when visualized in the original coordinate system they appear distorted, and sometimes disconnected. Nevertheless, this visualization strategy reveals which regions of the spatio-temporal domain each subnetwork is responsible for. Note that while Table 1 shows error statistics across 5 randomized trials, throughout this section we visualize the realization achieving the median PDE residual error by the end of training.

The rest of this section is organized as follows. The experiments on local adaptivity of window functions, including the chirp waveform, the advection equation, and the Helmholtz equation are presented in Sections 5.2.1, 5.2.2, and 5.2.3 respectively. Next, the tests on residual-based subdomain addition, including the Allen–Cahn equation, the forced Poisson equation, and the KdV equation are shown in Sections 5.3.3, 5.3.2, and 5.3.4. For each example comprehensive hyperparameter and optimization details are provided, along with plots of the learned solutions, PDE residuals, absolute errors, loss curves, and learned domain decompositions. Finally, we present an ablation study of the AB-PINN architecture in Section 5.4.

5.2 Local Adaptivity of Subdomains

In the following experiments, we consider a basic AB-PINN architecture in which the number of subdomains is fixed throughout training. For every comparison with FBPINNs, we initialize the AB-PINN window functions in exactly the same configuration as the FBPINN model tested. Thus, the only difference between the FBPINN and AB-PINN models in this section is the adaptivity of the window functions and the inclusion of a global network.

5.2.1 Chirp Waveform

We begin our numerical experiments with a motivating toy example which highlights the benefit of including adaptive basis functions within a PINN-based domain decomposition framework. In particular, we attempt to solve the first-order differential equation

$$u'(x) = 2\pi\omega p \cos(2\pi\omega x^p) x^{p-1}, \qquad x \in (0,1), \qquad u(1) = 0, \tag{21}$$

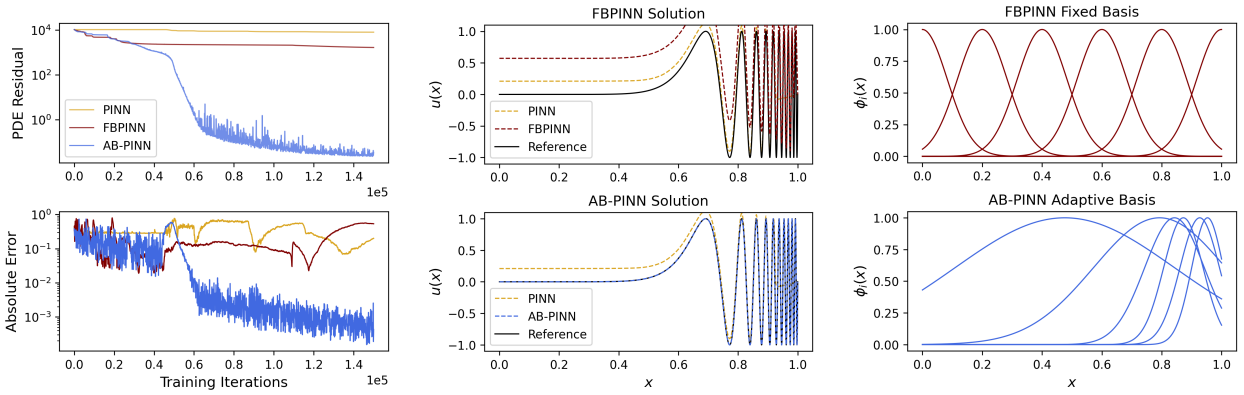
where $\omega, p \in \mathbb{N}$ are fixed. While the solution to (21) is analytically known to be

$$u(x) = \sin(2\pi\omega x^p), \qquad x \in (0,1). \tag{22}$$

The problem (21) provides a helpful example which can illustrate the difficulties of standard PINN frameworks and fixed domain decomposition strategies for solving multiscale oscillatory differential equations. For high values of ω the solution (22) becomes more oscillatory, while increasing the power p localizes the oscillations. To produce a challenging test case, we select $\omega = 10$ and $p = 10$.

Figure 2 shows the results for solving (21) using a standard PINN, an FBPINN and an AB-PINN. Across each test, the operator

$$\Psi[u](x) = u(x)\tanh(10(x-1))$$

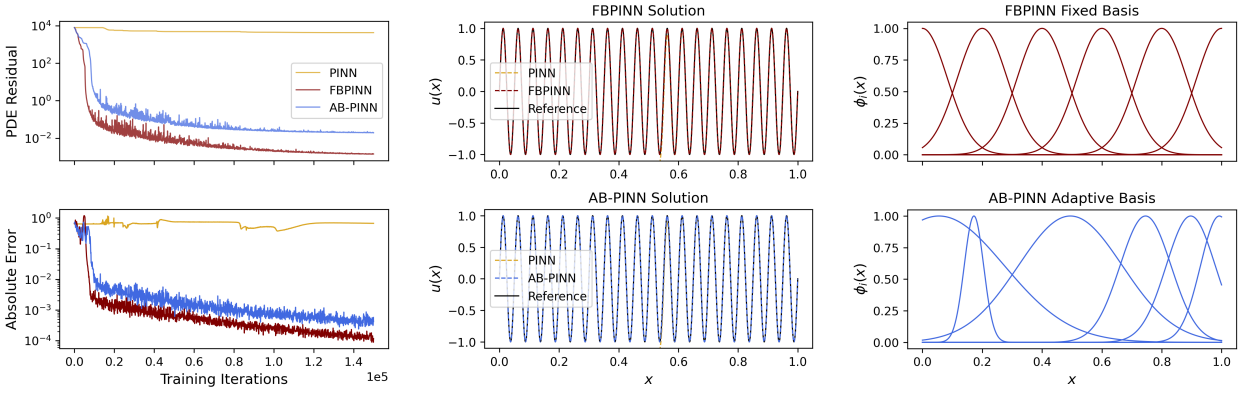


(a) Training and testing loss history (b) Learned solutions (c) Window functions

Figure 2: Solving a multiscale chirp waveform using a standard PINN, an FBPINN and an AB-PINN. In the second column the standard PINN solution is shown in yellow, while the FBPINN solution is plotted in red, the AB-PINN in blue, and the ground truth in black.

is used to enforce $u(1) = 0$ as a hard constraint, and at each training iteration, $2 \cdot 10^3$ collocation points are uniformly randomly sampled. The standard PINN is given by an MLP with four hidden layers, each consisting of 34 nodes. Both the AB-PINN and FBPINN models use $K = 6$ subdomains initialized according to (9) with $L_i = 12$ and μ_i linearly spaced over $[0, 1]$; for a visualization see the top row of Figure 2c. Each subnetwork in the AB-PINN model is an MLP with four hidden layers and 12 nodes in each layer, while the global network also has 12 nodes in each layer. In the FBPINN model, each subnetwork has four hidden layers, each with 13 nodes per layer. We use a slightly different number of nodes per layer for the AB-PINN and FBPINN subnetworks to ensure a fair comparison, such that all models have roughly the same total number of tunable parameters. In this case, all models considered have roughly $3.5 \cdot 10^3$ tunable parameters and are trained for $1.5 \cdot 10^5$ iterations. We use learning rates of 10^{-3} and 10^{-2} for the learnable window function parameters μ_i and L_i, respectively; see (9). After $5 \cdot 10^4$ training iterations, the window function parameters μ_i and L_i are frozen and no longer adapt.

As shown in Figure 2, the standard PINN and FBPINN approaches struggle to represent the solution (22), while the AB-PINN converges faster in terms of both the residual loss and the absolute solution error. The success of the AB-PINN is attributed to the flexibility of the adaptive window functions to concentrate on the region of the domain where the solution is most complex and the residual loss is highest. As shown in Figure 2c, the window functions cluster nearby $x = 1$, where the solution is highly oscillatory. The corresponding subnetworks, whose inputs are normalized within the support of these basis functions, are then able to minimize the residual loss in this region with greater success than the FBPINN.



(a) Training and testing loss history (b) Visualization of basis functions (c) Visualization of learned solutions

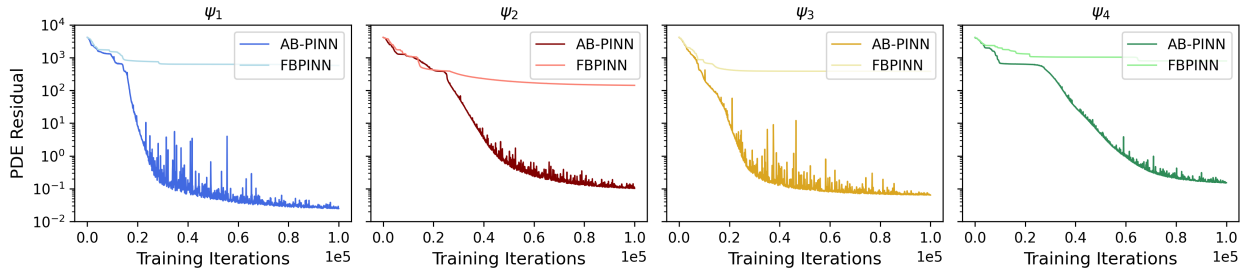
Figure 3: Solving a uniform sine wave using a standard PINN, an FBPINN, and an AB-PINN.

In Figure 3, we repeat the same experiment from Figure 2 with a differential equation that does not exhibit any multiscale behavior. We again consider (21), but now with $\omega = 20$ and $p = 1$. The reference solution for this equation is simply a sinusoidal oscillation with uniform frequency across the domain; see (22). As shown in Figure 3, the AB-PINN architecture no longer has a significant advantage compared to the FBPINN, likely due to the fact that the uniform spacing of the FBPINN window functions is already optimal. The results of Figures 2 and 3 collectively suggest that AB-PINNs are best suited for solving multiscale problems where the solution features are not uniformly distributed across the domain.

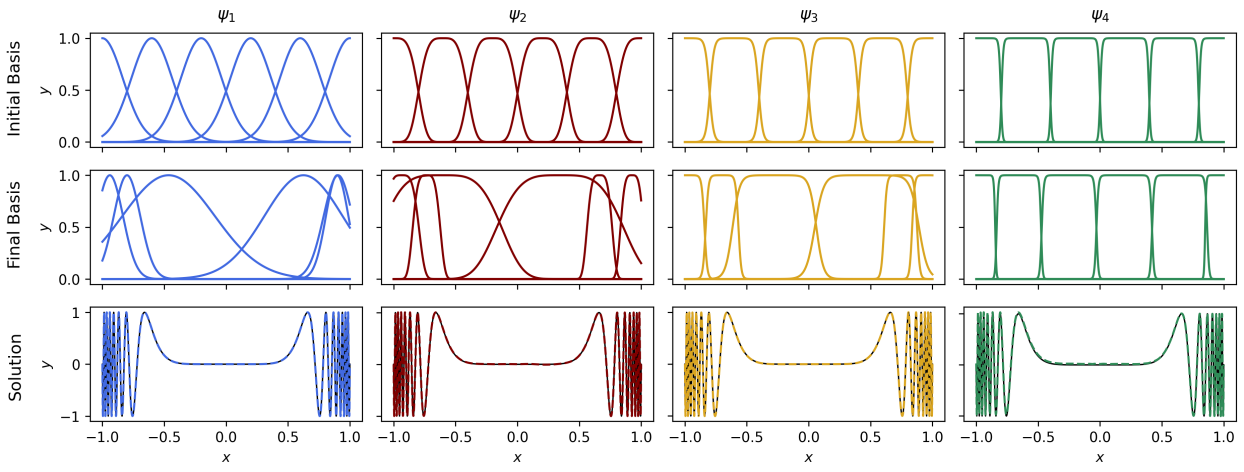
In Figure 4, we study the effect of the choice of the reference window function ψ (see (9)) for solving (21). For this experiment, we consider (21) with $x \in (-1, 1)$, $\omega = 7$, and $p = 8$. We consider the following reference window functions for defining both our AB-PINN and FBPINN models:

$$\begin{cases} \psi_1(x) = e^{-x^2/2} \\ \psi_2(x) = e^{-x^4/(4\ln 2)} \\ \psi_3(x) = \dfrac{1}{1+e^{-10(\sqrt{2\ln 2}+x)}} \cdot \dfrac{1}{1+e^{-10(\sqrt{2\ln 2}-x)}}, \\ \psi_4(x) = \dfrac{1}{1+e^{-10(2\ln 2 - x^2)}}. \end{cases} \tag{23}$$

The functions in (23) are designed such that each reaches a maximum of approximately 1 at $x = 0$ and symmetrically reduce to $1/2$ at $x = \pm\sqrt{2\ln 2}$. We remark that ψ_3 belongs to the collection of window functions considered in the original FBPINN paper [18].



(a) Loss curve comparison between AB-PINNs and FBPINNs for different choices of the reference window function ψ.



(b) Visualization of the FBPINN basis (top), the learned AB-PINN basis (middle), and the learned AB-PINN solution (bottom)

Figure 4: Studying the effect of the choice of reference window function on the performance of AB-PINNs for a multiscale chirp problem.

For the study in Figure 4, the individual window functions ϕ_i are initialized according to (9) with $L_i = 6$ and μ_i linearly spaced over $[-1, 1]$; for a visualization see the top row of Figure 4. The loss curves depicted in Figure 4a indicate that for all choices of the reference window function, an AB-PINN framework in which the window function parameters adapt outperforms the static FBPINN model. The middle row of Figure 4 shows the learned window functions after

their parameters have been frozen following $2 \cdot 10^4$ training iterations, and the predicted AB-PINN solution is shown in Figure 4. We observe that the window functions with flat peaks and quickly decaying tails, e.g. ψ_4, do not display as much adaptivity as those without flat peaks and with slowly decaying tails, e.g. ψ_1. In the remainder of our numerical experiments we select ψ_1 as our reference window function.

5.2.2 Advection Equation

We next consider the advection equation

$$\partial_t u(t,x) + c \cdot \partial_x u(t,x) = 0, \qquad t \in (0,1),\ x \in (-1,1) \tag{24}$$

$$u(0,x) = \sin(\pi x), \qquad x \in (-1,1) \tag{25}$$

$$u(t,1) = u(t,-1), \qquad t \in [0,1], \tag{26}$$

with $c = 10$. The ground truth solution to (24)-(26) is analytically given by $u(t,x) = \sin(\pi(x - ct))$, which is visualized in Figure 1b. In Figure 5, we approximate the solution of the advection equation using standard PINNs, FBPINNs, and AB-PINNs. All models considered use the constraining operator

$$\Psi[u](t,x) = u(t,x)\tanh(t) + \sin(\pi x)$$

to enforce the initial condition (25) as a hard constraint, while input-coordinate Fourier embeddings are used to enforce the periodic boundary condition (26); see Section 3.3.

The FBPINN and AB-PINN models both use $K = 16$ subdomains, with the AB-PINN subdomains initialized according to the FBPINN layout shown in Figure 5. Each MLP used in this test has four hidden layers, with the number of nodes in each layer carefully selected such that the PINN, FBPINN, and AB-PINN models all have roughly $7 \cdot 10^3$ tunable parameters. The initial learning rates for the tunable window function parameters μ_i and L_i are $5 \cdot 10^{-4}$ and 10^{-3}, respectively. Each model is trained for $6 \cdot 10^4$ iterations, with 1000 collocation points uniformly sampled at each step. After 10^4 training steps, the tunable window function parameters in the AB-PINN model are frozen.

As shown in Figure 5, the AB-PINN converges faster and with higher accuracy than both the standard PINN and the FBPINN. Several notable features of the learned domain decomposition shown in the bottom row of Figure 5 enable this faster convergence. First, the window functions adapt to the geometry of the underlying solution, stretching along its characteristic lines. Moreover, we observe that the spatial distribution of window functions is skewed towards $t = 0$, indicating that the adaptive subdomains place a higher emphasis on learning the solution at earlier times. Such behavior is similar to curriculum training and causality-inducing reweighting or resampling strategies for PINNs [62, 63, 64, 40], all of which encourage the solution to be learned sequentially along time for more accurate results. We note that the AB-PINN architecture should be viewed as an enhancement rather than a replacement of these approaches.

5.2.3 Helmholtz Equation

We now consider the two-dimensional Helmholtz equation

$$\Delta u(x,y) + k^2 u(x,y) = (k^2 - (a\pi)^2 - (b\pi)^2)\sin(a\pi x)\sin(b\pi y), \qquad (x,y) \in (-1,1)^2, \tag{27}$$

$$u(x,\pm 1) = 0, \qquad u(\pm 1, y) = 0, \qquad x \in [-1,1],\ y \in [-1,1] \tag{28}$$

where we set $k = 2$, $a = 1$, and $b = 7$. The solution of (27)-(28) is analytically given by

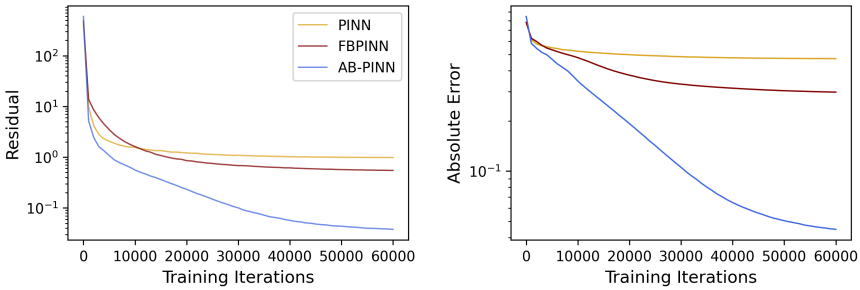
$$u(x,y) = \sin(a\pi x)\sin(b\pi y), \tag{29}$$

which is visualized in Figure 1c. By selecting $a = 1$ and $b = 7$ we produce a solution with rapid oscillations in one direction and slow oscillations in the other. Without this prior knowledge, it is reasonable to initialize the window functions to uniformly cover the domain on a fixed isotropic grid. In Figure 6, we show that allowing the window functions to evolve during training under our AB-PINN framework lets them adapt to the geometry of the underlying PDE solution, thereby leading to a model which produces more accurate predictions.

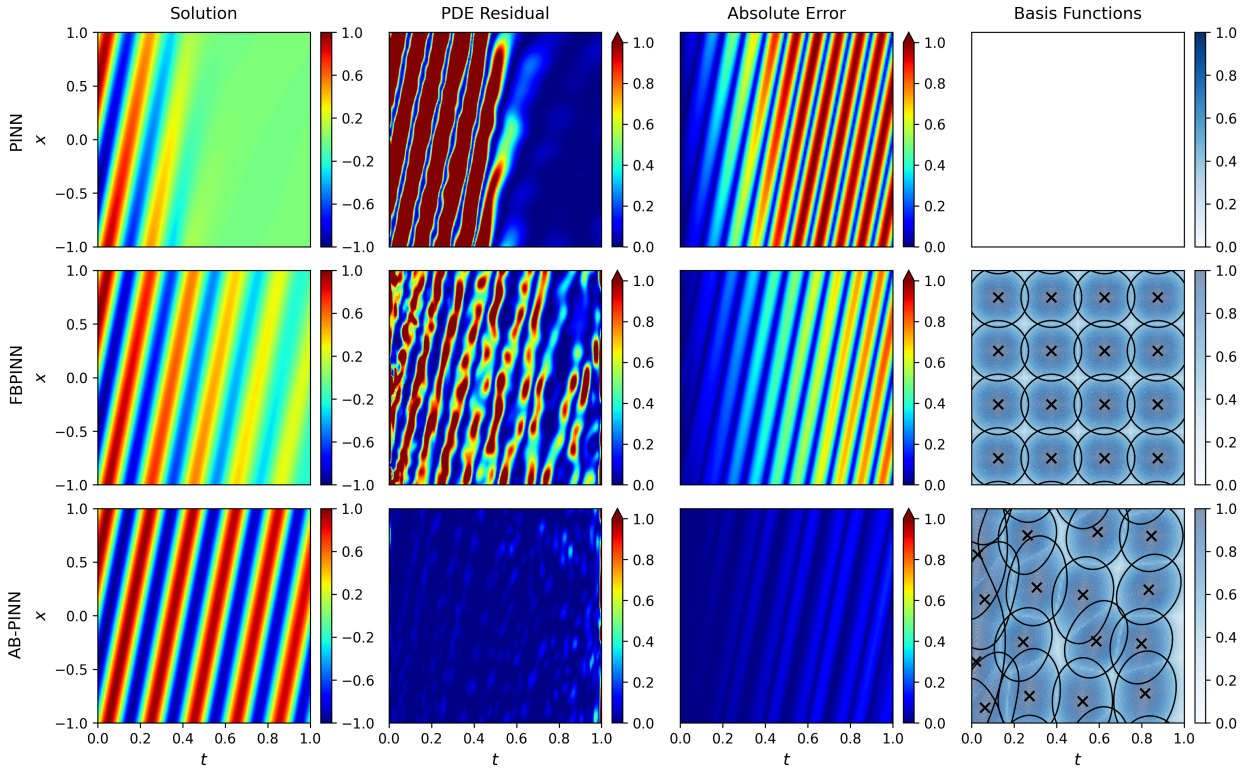
In particular, we show the results of training a standard PINN, an FBPINN, and an AB-PINN to approximate the solution of (27)-(28). Throughout, we use the constraining operator

$$\Psi[u](x,y) = u(x,y)\tanh(x-1)\tanh(x+1)\tanh(y-1)\tanh(y+1) \tag{30}$$

to enforce the boundary condition (28). Both the FBPINN and AB-PINN models use $N = 16$ subdomains. We test two different subdomain layouts for the FBPINN model (see Figure 6), neither of which is able to match the accuracy of the AB-PINN. The AB-PINN model is initialized such that the window functions are in the same configuration as the first FBPINN model tested; see Figure 6. All models considered use global networks and subnetworks with two hidden



(a) Convergence of the PDE residual loss and the absolute error of the predicted solution



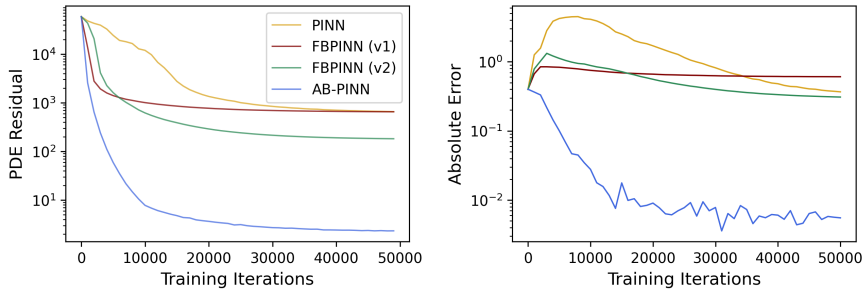
(b) Visualization of the learned basis functions, PINN solution, PDE residual, and absolute solution error.

Figure 5: Solving the advection equation using PINNs, FBPINNs, and AB-PINNs.

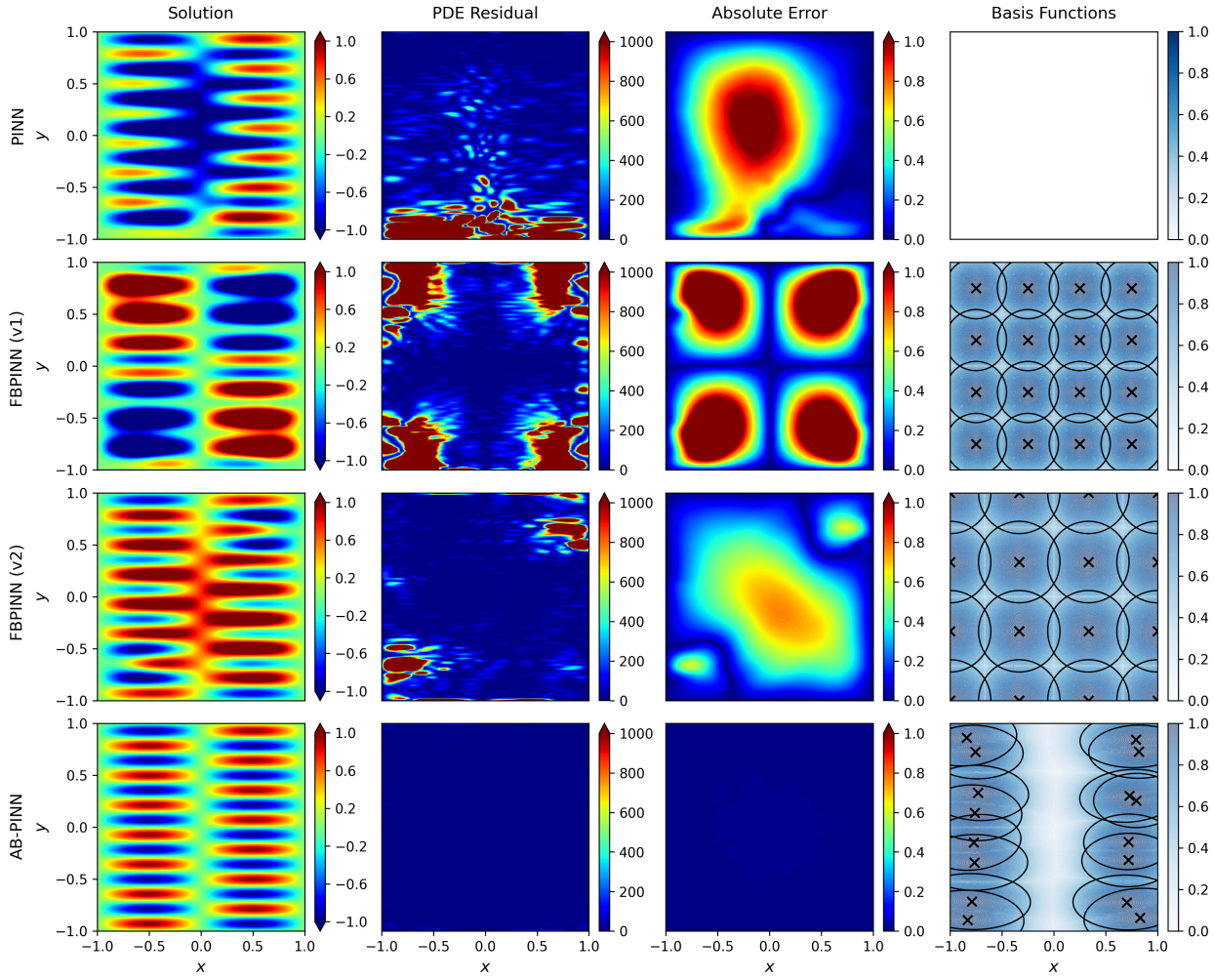
layers with a variable number of nodes per model, such that each model has roughly 10^4 free parameters. In particular, the standard PINN uses four hidden layers with 57 nodes each, the FBPINNs use two hidden layer subnetworks with 23 nodes in each layer, and the AB-PINN uses two hidden layer subnetworks with 22 nodes per layer along with a two hidden layer global network with 20 nodes in each layer. We train each model for $5 \cdot 10^4$ training iterations, uniformly sampling 10^3 collocation points at each step. All initial learning rates are set to 10^{-3}, aside from those for the window parameters L_i which we set as $5 \cdot 10^{-4}$. After the first 10^4 iterations, the tunable window function parameters in the AB-PINN model are frozen. While in Figure 6, we use solely the local adaptivity of the window functions to learn an accurate AB-PINN model for the Helmholtz equation, in the following section, we show how the residual-based subdomain addition strategy introduced in Section 3.2 can be used to solve the Helmholtz problem as well; see Figure 7.

5.3 Residual-Based Subdomain Addition

In Section 5.2 we performed several comparisons between AB-PINNs, FBPINNs, and standard PINNs in the case where the number of AB-PINN window functions is fixed throughout training. In this section, we study the residual-based subdomain addition technique introduced in Section 3.2. Our experiments show that adding new AB-PINN subdomains



(a) Training and testing loss history



(b) Visualization of the PINN solution, PDE residual, and absolute solution error after training both the fixed basis and adaptive basis approaches.

Figure 6: Solving the Helmholtz equation (27)-(28) using a vanilla PINN, a fixed basis approach, and an adaptive basis approach. The adaptive basis fits the multiscale geometry of the PDE and leads to faster convergence of the PDE residual loss and a more accurate solution.

during training can be useful for avoiding unwanted local minima and delivering needed expressive power in regions of the domain where the PINN is struggling to converge. As a reminder, we write AB-PINN+ as shorthand to denote an AB-PINN model whose subdomains were introduced throughout training using residual-based subdomain addition.

5.3.1 Helmholtz Equation

We begin by illustrating how the residual-based subdomain addition can be also used to effectively solve the Helmholtz problem introduced in Section 5.2.3. In Figure 7, we solve the Helmholtz equation (27)-(28) using the AB-PINN+ framework, progressively adding in more subdomains throughout training. Figure 7a compares the AB-PINN+ performance to the fixed AB-PINN initialization from Section 5.2.3. Each column of Figure 7b then shows the predicted solution, PDE residual, and AB-PINN+ subdomains after a fixed number of iterations have elapsed.

We begin the experiment with a single two-hidden layer global network with 20 nodes per layer. Every 10^3 steps of training we add a new window function and corresponding subnetwork with two hidden layers each with 22 nodes per layer. As described in Section 3.2, the location of the new subdomain is adaptively determined by sampling from a density which is proportional to the current PDE residual. After the subdomains are introduced, using the same initialization as in Section 5.2.3, i.e., $L_i = 4I$ where I is the identity matrix, their tunable parameters μ_i and L_i are free to evolve dynamically until $2.5 \cdot 10^4$ iterations have elapsed, at which point their learning rates are set to zero. As in Section 5.2.3, the initial learning rates of the subdomain parameters μ_i and L_i are 10^{-3} and $5 \cdot 10^{-4}$, respectively. In total, we introduce $K = 16$ subdomains and arrive at a final AB-PINN+ architecture with approximately 10^4 free parameters, same as the fixed AB-PINN from Section 5.2.3.

As shown in Figure 7b, the PDE residual reduces significantly in the regions where new subdomains have been added. This suggests that the addition of new AB-PINN+ subdomains can be used to reduce localized regions of high PDE residual. Notably, while the FBPINN and PINN architectures studied in Section 5.2.3 were not able to provide an accurate solution to the Helmholtz equation, the AB-PINN+ is successful despite having roughly the same number of tunable parameters. Finally, we note that the learned domain decomposition from residual-based subdomain addition (AB-PINN+) differs significantly from the learned domain decomposition when all basis functions are initialized at the start of training (AB-PINN); see Figures 6 and 7. In the next section, we provide a detailed comparison of the learned domain decomposition under these two setups for a different PDE.

5.3.2 Locally Forced Poisson equation

In this example, we demonstrate that the AB-PINN+ domain decompositions arrived at via residual-based subdomain addition can, in certain situations, yield a more accurate solution than the AB-PINN decomposition learned using solely the local adaptivity of the window functions. In particular, we consider the locally forced Poisson equation

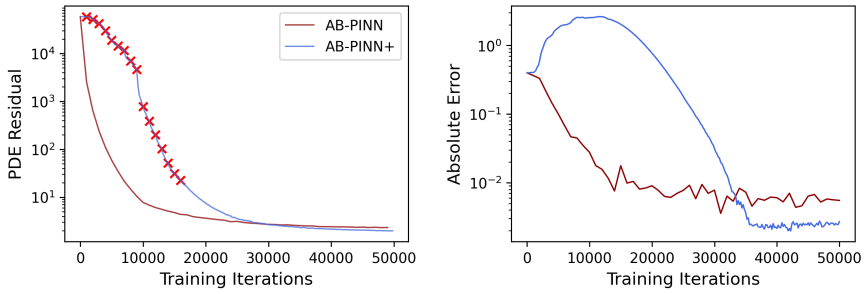
$$\Delta u(x,y) = \sum_{i=1}^{9} \frac{(x-c_{i,1})^2 + (y-c_{i,2})^2 - 2\sigma^2}{\sigma^4} \exp\left(- \frac{(x-c_{i,1})^2 + (y-c_{i,2})^2}{2\sigma^2} \right), \quad (x,y) \in \Omega \tag{31}$$

$$u(x,y) = \sum_{i=1}^{9} \exp\left(- \frac{(x-c_{i,1})^2 + (y-c_{i,2})^2}{2\sigma^2} \right), \qquad (x,y) \in \partial\Omega, \tag{32}$$

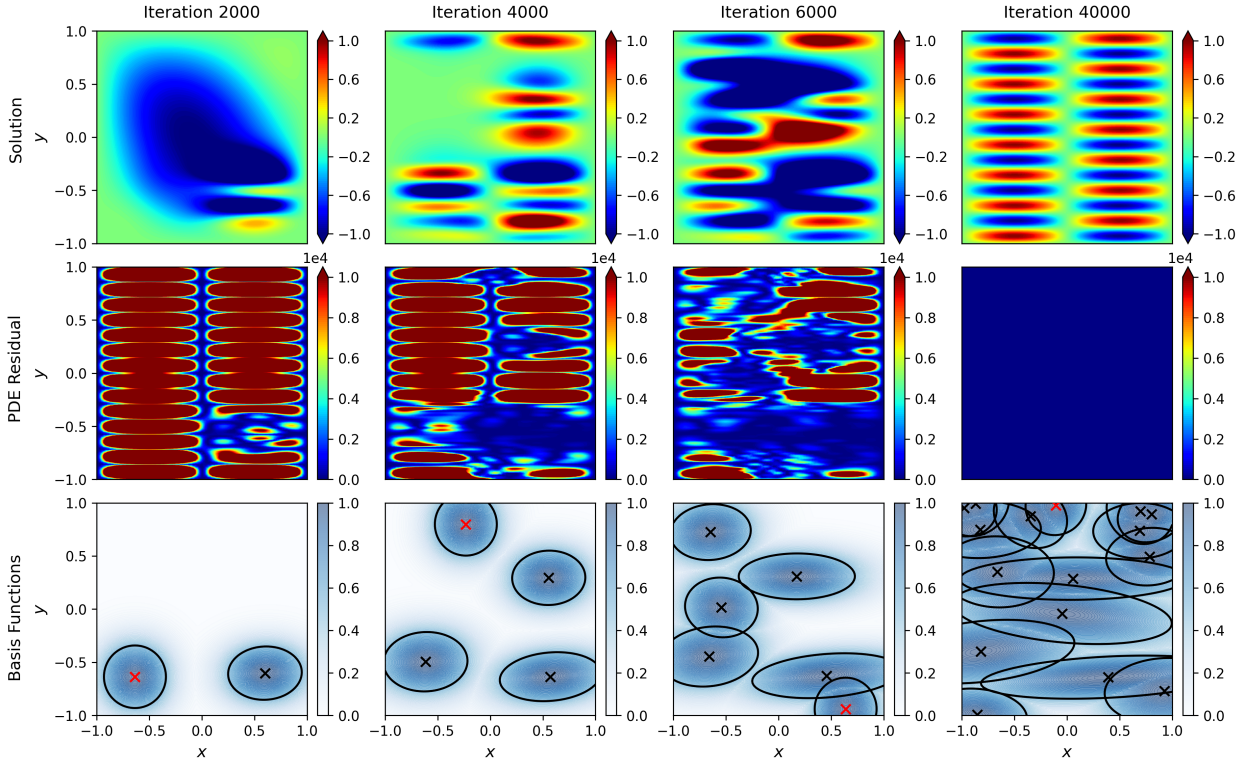
where $\Omega = [-1,1]^2$, $\sigma = 0.025$, and $c_{i,j} \in (-1,1)$. The solution to (31)-(32) over Ω is given by the sum of Gaussians used to specify the boundary condition in (32); see Figure 1d. For small values of σ, the boundary condition (32) can be approximated as $u(x,y) \approx 0$ for $(x,y) \in \partial\Omega$, and thus we use the same constraining operator (30) from Section 5.2.3 to enforce this approximate boundary condition.

In Figure 8, we solve the locally forced Poisson equation using a standard PINN, two AB-PINN models where all subdomains are initialized at the beginning of training, as well as an AB-PINN+ which makes use of residual-based subdomain addition. Throughout, all models tested have roughly $5 \cdot 10^3$ tunable parameters, are trained for 10^5 iterations, and use 1000 uniform random collocation points per iteration. In particular, the PINN uses two hidden layers with 70 nodes each, while the AB-PINN and AB-PINN+ models use a two hidden layer networks with 20 nodes per layer for both the global network and subnetworks. The FBPINN model uses two hidden layer subnetworks with 22 nodes per layer. The learning rates of the tunable window parameters μ_i and L_i are initialized at 10^{-3} and $5 \cdot 10^{-3}$, respectively. The AB-PINN and FBPINN models use exactly $K = 9$ subdomains and freeze the tunable parameters of the window functions after $7 \cdot 10^4$ steps. For the AB-PINN+, we begin with a single global network and add in a new subdomain every $5 \cdot 10^3$ training iterations, until $K = 9$ subdomains have been introduced. The AB-PINN and FBPINN models are initialized with the centers μ_i uniformly spaced over the unit square, while the parameters L_i are initialized as $5I$ where I denotes the identity matrix. For the AB-PINN+, we also set $L_i = 5I$.

If the AB-PINN models configures their subdomains optimally, then one window function will be concentrated on each peak of the solution. In Figure 8, we observe that the AB-PINN+ with residual-based subdomain addition indeed achieves this goal and leads to an accurate solution, while the two AB-PINNs with local adaptivity starting from a square grid layout of window functions struggle. Indeed, each time a new AB-PINN+ subdomain is introduced it targets a region of high PDE residual, which for this problem corresponds to one of the Gaussian peaks. After the subdomain



(a) Training and testing loss history with and without residual-based subdomain addition. We write AB-PINN+ to denote the model which uses residual-based subdomain addition. The red x indicates when new subdomains are added.



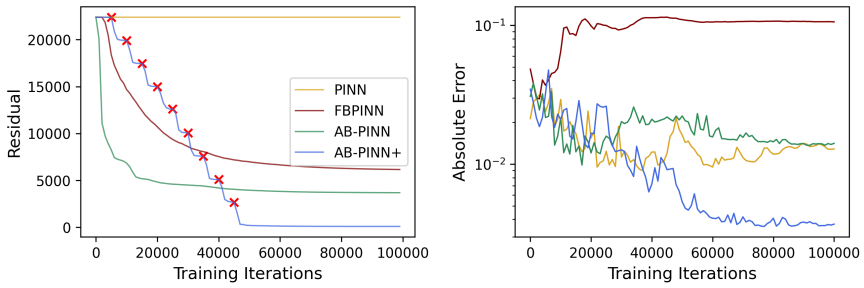
(b) Visualizing the AB-PINN+ as training progresses. Each column shows the solution, residual, and learned domain decomposition after a fixed number of iterations have passed. The red x marker indicates the newly introduced subdomain.

Figure 7: Solving the Helmholtz equation using the AB-PINN+ framework. Figure 7a compares its convergence with the AB-PINN from section 5.2.3, and Figure 7b visualizes the training progress.

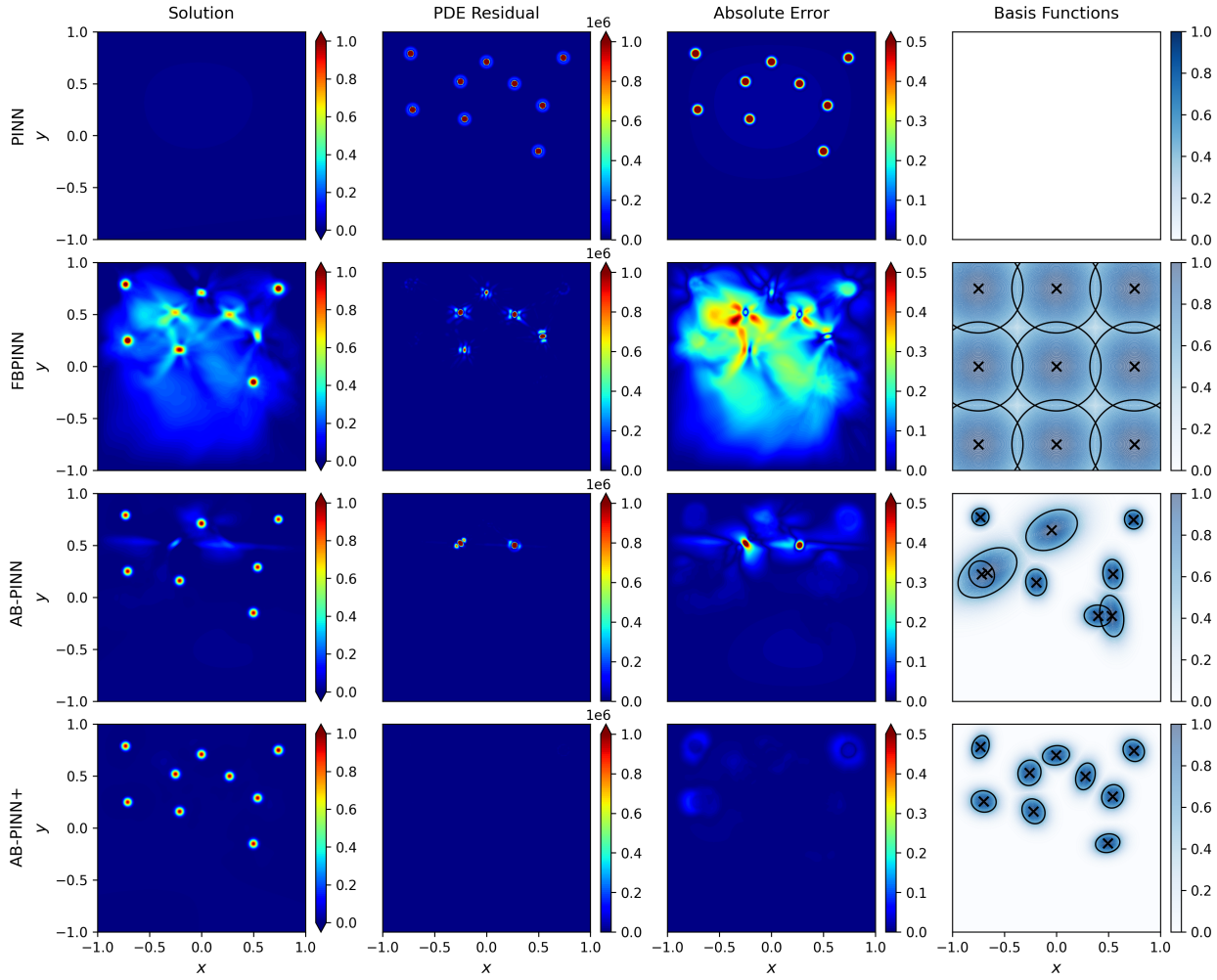
is introduced, the residual in the targeted region decreases, and thus the next subdomain is placed on a different peak. This leads to a near optimal configuration of subdomains where each is placed on its own Gaussian peak, as shown in the bottom row of Figure 8b. On the other hand, local adaptivity alone of the subdomains results in some Gaussian peaks being covered by multiple subdomains while other peaks don't have a dedicated subdomain, as shown in the middle two rows of Figure 8b. This example showcases the benefits of residual-based subdomain addition for problems with localized features.

5.3.3 Allen–Cahn Equation

In this example, we demonstrate how the residual-based subdomain addition strategy can be used to improve the convergence of a PINN which is either exhibiting slow convergence or has gotten stuck in a local minima. Typically, one stops training when a PINN has reached a local minimum, tunes several hyperparameters, and then attempts to train



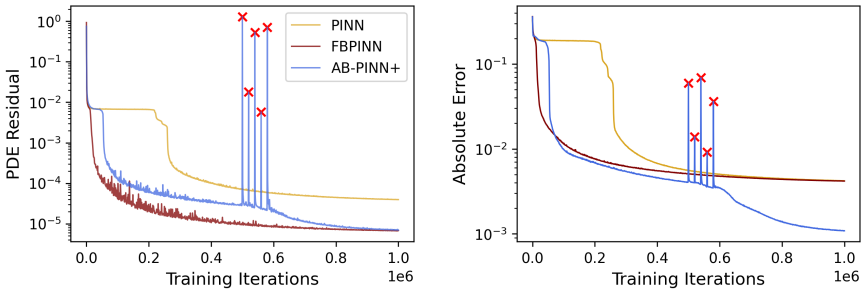
(a) PDE residual loss and absolute error of the predicted solutions. The red x in the loss curve indicates when a new subdomain is added.



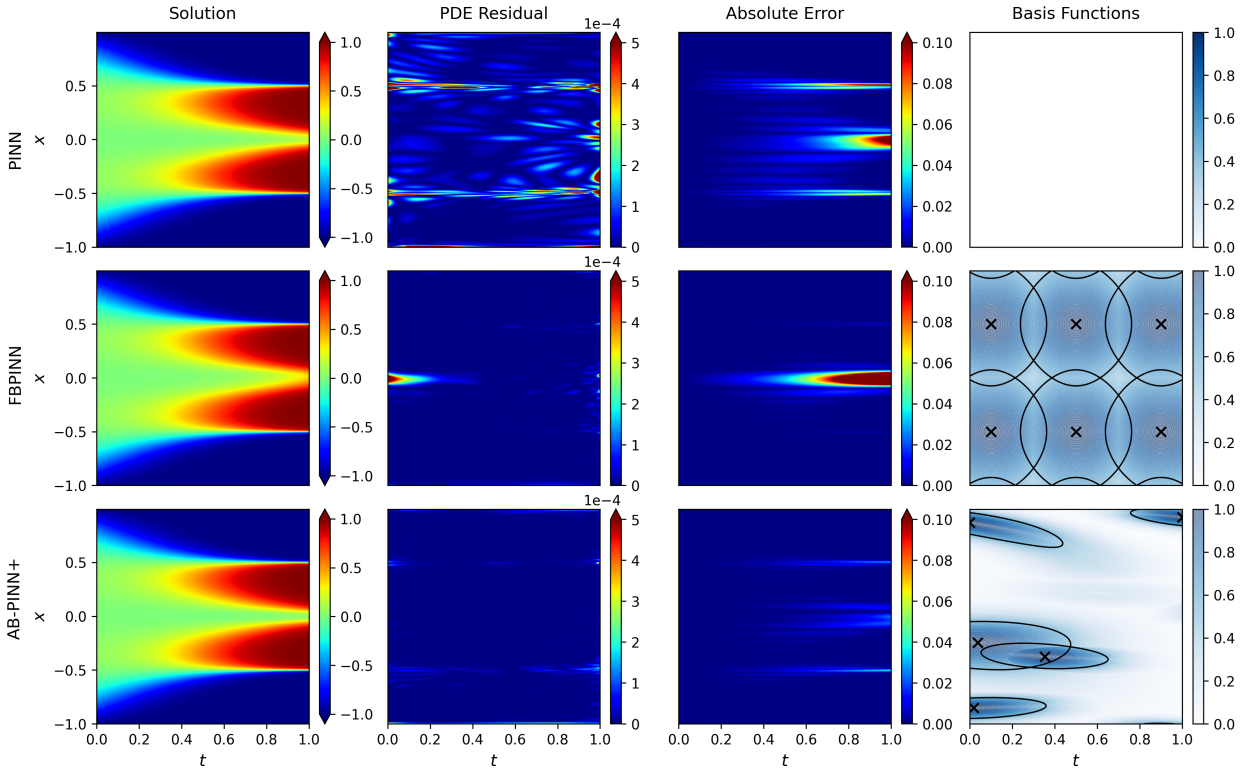
(b) Visualization of the PINN solution, PDE residual, and absolute solution error, and learned domain decomposition.

Figure 8: Solving the locally forced Poisson equation using the AB-PINN+ (bottom row of Figure 8b) leads to better accuracy than relying solely on local adaptivity of the window functions initialized from a uniform configuration (middle two rows of Figure 8b).

the model again from scratch. With our residual-based subdomain addition, we instead retain the progress of the PINN which has struggled to converge and we improve the solution with the help of new AB-PINN+ subdomains. In practice, this simply corresponds to training an AB-PINN+ model in which the initial PINN is regarded as the global network.



(a) PDE residual loss and absolute error of the predicted solution. The curves spike every time a new subdomain is added, which we highlight with red x markers.



(b) Visualization of the PINN solutions, PDE residuals, and absolute solution errors, and the learned window functions.

Figure 9: Using the AB-PINN+ framework to achieve a high-accuracy solution of the Allen–Cahn equation.

We will consider the Allen–Cahn equation

$$\partial_t u(t,x) - 0.0001 \cdot \partial_x^2 u(t,x) + 5u^3(t,x) - 5u(t,x) = 0, \qquad (t,x) \in (0,1) \times (-1,1), \tag{33}$$

$$u(0,x) = x^2 \cos(\pi x), \qquad x \in (-1,1), \tag{34}$$

$$u(t,1) = u(t,-1), \quad u_x(t,1) = u_x(t,-1), \qquad t \in (0,1), \tag{35}$$

which is a standard benchmark problem in the PINN literature. The reference solution to (33)-(35) is shown in Figure 1e. We begin by solving the system (33)-(34) using a standard PINN with four hidden layers and 10 nodes per layer which imposes the periodic boundary conditions (35) using Fourier embedding (see Section 3.3), and the initial condition (34) with the constraining operator

$$\Psi[u](t,x) = \tanh(t)u(t,x) + x^2 \cos(\pi x).$$

The PINN we consider has approximately 400 tunable parameters, and thus its convergence slows significantly before a high-accuracy prediction of the PDE is achieved. After $5 \cdot 10^5$ iterations have elapsed, we begin adding new AB-PINN+ subdomains every $2 \cdot 10^4$ iterations until $K = 5$ subdomains have been introduced, while the original PINN

continues training as the global network. Each subdomain is initialized with a four hidden layer subnetwork also comprising of 10 nodes per layer, and we set $L_i = \text{diag}(4.5, 0.75, 0.75)$ which is defined in the three-dimensional Fourier embedded space. After all AB-PINN+ subdomains have been added, the resulting model has approximately $2.3 \cdot 10^3$ tunable parameters. The learning rates for the window function parameters μ_i and L_i are initialized at 10^{-3} and 10^{-2}, respectively, and they are frozen after $6.5 \cdot 10^5$ iterations. We also compare with an FBPINN using $K = 6$ subdomains using the same L_i and each FBPINN subnetwork comprising two 4 hidden layers with 10 nodes per layer.

In Figure 9a, we see that both the PDE residual loss and absolute error of the predicted solution spike every time a new subdomain is added. This makes intuitive sense, as we significantly perturb the solution every time a new subdomain is introduced. In total, we see five spikes in Figure 9a, each corresponding to the addition of a new subdomain. After the new window functions and corresponding subnetworks have been added, the residual loss and absolute error reduce rapidly. The final predicted AB-PINN+ solution then shows improvement compared to the solution without subdomain addition; see Figure 9b.

5.3.4 Korteweg–De Vries Equation

We conclude our numerical experiments with a challenging PINN benchmark example, the Korteweg–De Vries (KdV) equation:

$$\partial_t u(t,x) + \eta \cdot u(t,x)\partial_x u(t,x) + \mu^2 \cdot \partial_x^3 u(t,x) = 0, \qquad (t,x) \in (0,1) \times (-1,1) \tag{36}$$

$$u(0,x) = \cos(\pi x), \qquad x \in [0,1] \tag{37}$$

$$u(t,1) = u(t,-1), \qquad t \in (0,1). \tag{38}$$

The KdV equation is used to model nonlinear dispersive waves and can feature a range of multiscale behaviors. Here, we consider parameter values $\eta = 1$ and $\mu = 0.022$ leading the initial cosine to evolve into a train of solitary-type waves [65, 54]; see Figure 1f for a visualization of the reference solution.

We attempt to solve (36)-(37) using both a standard PINN and an AB-PINN+. The initial condition (37) is enforced using the constraining operator

$$\Psi[u](t,x) = \tanh(t)u(t,x) + \cos(\pi x),$$

and the periodic boundary condition (38) is enforced using Fourier embeddings; see Section 3.3.

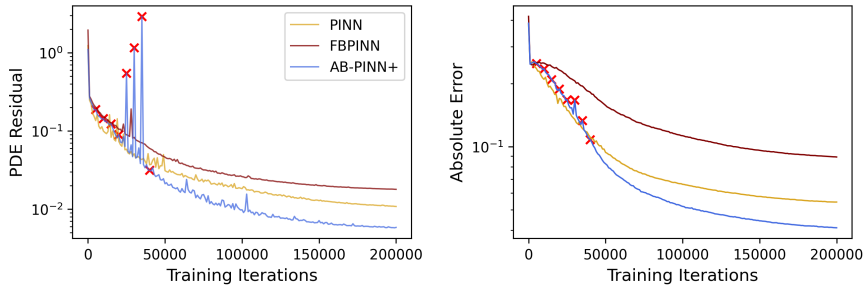
In Section 5.3.3, we showed how residual-based subdomain addition can prevent PINNs from getting stuck in local minima by adding additional expressive power in regions of high residual loss. Thus, the final AB-PINN+ architecture in Section 5.3.3 was larger than the standard PINN we used for comparison. Here, we instead compare the final AB-PINN+ model with a standard PINN of equal size. That is, the primary difference in experimental setup with Section 5.3.3 is simply the size of the standard PINN we compare with.

In particular, all models considered have roughly $5 \cdot 10^3$ tunable parameters. The PINN uses four hidden layers with 41 nodes per layer, while the AB-PINN+ uses a four hidden layer global network with 25 nodes per layer and four hidden layer subnetworks with 10 nodes per layer. Moreover, the FBPINN uses $K = 9$ subnetworks, each consisting of 4 hidden layers with 13 nodes per layer. We train the models for $2 \cdot 10^5$ iterations, randomly sampling 1000 collocation points each step. The tunable window function parameters μ_i and L_i have learning rates initialized as 10^{-3} and $5 \cdot 10^{-3}$, respectively, and they are frozen at 10^5 iterations. The parameters L_i are initialized in the same way as in Section 5.3.3. We use residual-based subdomain addition to introduce a new subdomain every $5 \cdot 10^3$ iterations, until $K = 8$ subdomains are active.

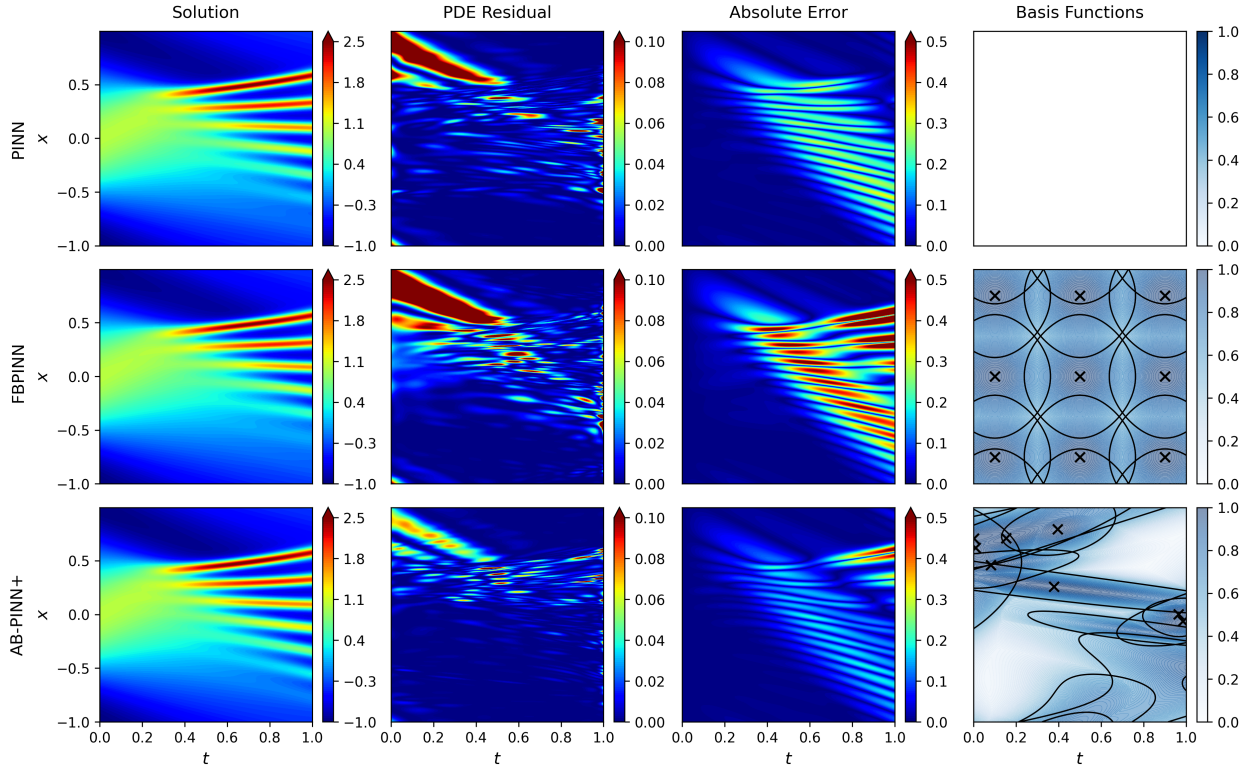
As shown in Figure 10, the AB-PINN+ model using residual-based subdomain addition approximates the reference solution of the KdV equation with higher accuracy than the standard PINN. Similar to Figure 9a, we see that the loss spikes each time a new subdomain is added. After the subdomains have been added, the AB-PINN+ converges with lower PDE residual and absolute solution error than the standard PINN. In Figure 10b, we observe that a large cluster of subdomains is placed in the upper-left corner of the spatio-temporal domain, which corresponds to a region where the standard PINN solution has particularly high PDE residual. Thus, the placement of these adaptive subdomains is likely responsible for the AB-PINN+'s reduction of the PDE residual in this region, contributing to a more accurate solution.

5.4 Ablation Study

Our proposed AB-PINN architecture involves three central modifications to the standard FBPINN architecture: (1) locally adaptive subdomains, (2) residual-based subdomain addition during training, and (3) a low-frequency global network. In this section, we conduct a comprehensive ablation study of these features.



(a) PDE residual loss and absolute error of the predicted solution. The red x indicates when new subdomains are introduced.



(b) Visualization of the PINN solution, PDE residual, absolute solution error, and the learned window functions.

Figure 10: Solving the KdV equation using a standard PINN, an FBPINN, and an AB-PINN+.

As a representative multiscale test case, we consider a first-order differential equation over the domain $[0, 1]$ with solution given by

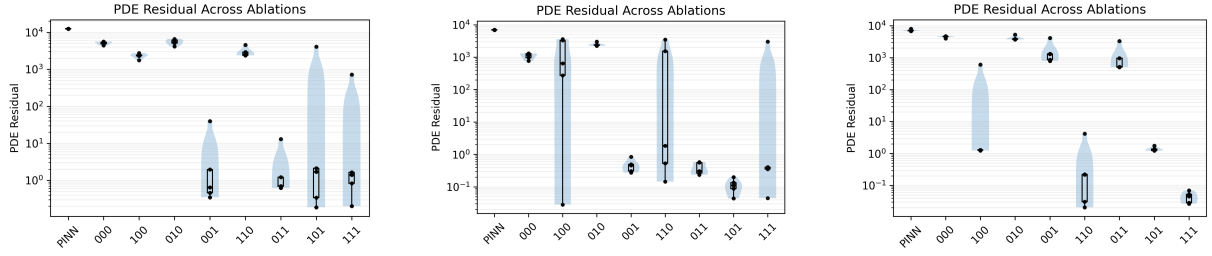
$$u(x) = \sin(2\pi\omega x^p) + \sin(4\pi x) + \exp(-(x-c)^2/\sigma^2). \tag{39}$$

The first term in (39) is the chirp waveform (21) introduced in Section 5.2.1 with $\omega = 7$ and $p \in \{15, 25\}$, and we set $c = 0.15$, as well as $\sigma = 0.0025$. We also use same constraining operator as in Section 5.2.1 to enforce the boundary condition $u(1) = 0$. This example incorporates high frequency oscillations from the chirp component, global low frequency behavior through the function $\sin(4\pi x)$, and a localized bump, each of which is chosen to highlight the benefits of local adaptivity, global networks, and residual-based subdomain addition, respectively.

Throughout, we use subnetworks and global networks with two hidden layers, each consisting of 32 nodes per layer. We initialize each experiment with $K = 6$ subdomains. When subdomain addition is enabled, two additional subdomains are introduced via the residual-based strategy following $5 \cdot 10^4$ and $6 \cdot 10^4$ iterations. Architectures without a global network or subdomain addition (FBPINN) comprise approximately $7 \cdot 10^3$ tunable parameters, architectures using the global network without subdomain addition comprise roughly $8 \cdot 10^3$ tunable parameters (AB-PINN), while architectures using both the global network and subdomain addition (AB-PINN+) have approximately 10^4 parameters by the end of

training. Across all tests, we train for 10^5 iterations and freeze the tunable subdomain parameters after $8 \cdot 10^4$ iterations have elapsed. At the onset of training, subdomains are initialized with means μ_i uniformly spaced over $(0, 1)$. All tunable parameters have a learning rate of 10^{-3}, except for the shape parameters L_i which use a learning rate of 10^{-2}. All learning rates are decayed on an exponential schedule to 1% of their initial value by the end of training.

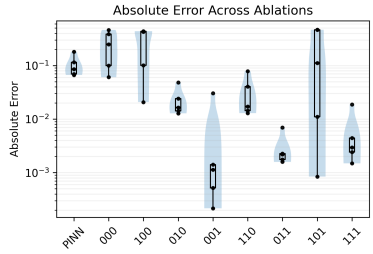
Figure 11 summarizes the ablation results across all 8 architecture variations and across three scenarios in which the frequency parameter p in (39) and the subdomain initialization L_i are varied. Each architecture is denoted by a binary triplet xyz where $x, y, z \in \{0, 1\}$, specifying whether local adaptivity, a global network, and subdomain addition are used. In this notation, 000 is the standard FBPINN architecture, 110 is the AB-PINN and 111 is the AB-PINN+. The results in Figure 11 indicate that the AB-PINN+ architecture is generally the most robust across the range of experimental conditions considered. The trials with $L_i = 12$ illustrate the importance of residual-based subdomain addition, while the trials with $L_i = 100$ highlight the benefits of the global network and local adaptivity of subdomains. The improvements from residual-based subdomain addition and a global network come with increased computational cost due to the growing model size, as shown in the wall-clock times presented in Figure 12. However, this tradeoff may be favorable in practice, as subdomain addition can avoid the need to restart training with larger architectures and retune hyperparameters.



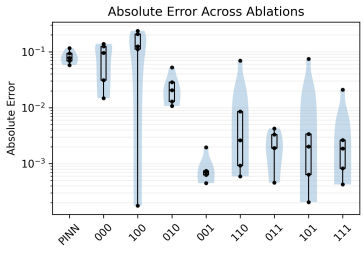
(a) PDE residual ablation with $p = 25$ and $L_i = 12$.

(b) PDE residual ablation with $p = 15$ and $L_i = 12$.

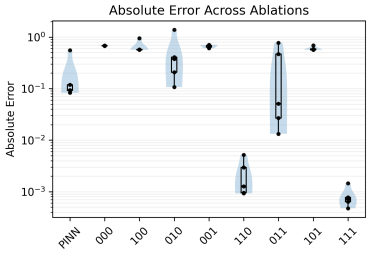
(c) PDE residual ablation with $p = 15$ and $L_i = 100$.



(d) L^1 error ablation with $p = 25$ and $L_i = 12$.



(e) L^1 error ablation with $p = 15$ and $L_i = 12$.



(f) L^1 error ablation with $p = 25$ and $L_i = 100$.

Figure 11: Ablation studies of the AB-PINN architecture for the modified chirp equation (39). Each architecture is represented by a binary triplet with each digit indicating the presence of subdomain adaptivity, a global network, and residual-based subdomain addition. For example, 000 is an FBPINN, while 110 is the AB-PINN, and 111 is an AB-PINN+. Each architecture is used to solve the problem five times and the resulting distributions of PDE residuals and L^1 errors are presented via box- and violin plots.

6 Discussion

Across the numerical results in Section 5, we have shown that AB-PINNs present several advantages when solving multiscale PDEs. Notably, the adaptive soft domain decomposition adapts to the intrinsic features of the underlying PDE solution and in many cases leads to faster convergence compared to a fixed FBPINN domain decomposition. The introduction of new subdomains on-the-fly during training also helps prevent unwanted local minima when the current architecture is not sufficiently expressive. It also provides a non-local mechanism for adjusting the underlying domain decomposition which we found to be advantageous when solving the locally forced Poisson equation (see Section 5.3.2) and the modified chirp equation (see Section 5.4).

Despite these benefits, the approach also faces its own limitations which should be addressed in future works. First, the current implementation is computationally inefficient since all subnetworks are evaluated on all collocation points,

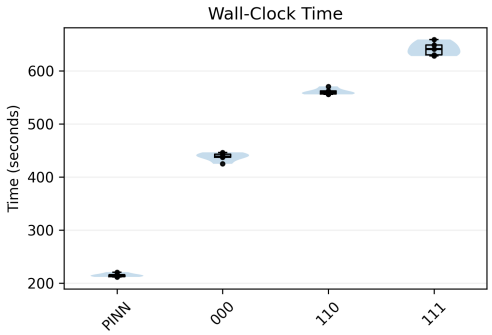


Figure 12: Wall-clock comparison between a PINN, FBPINN (000), AB-PINN (110), and AB-PINN+ (111) for solving the modified chirp problem (39). Training is conducted on the CPU of a MacBook Air using the Apple M3 chip.

including those outside the support of their corresponding subdomain. Because the total number of trainable parameters in an AB-PINN scales linearly with the number of subdomains, the leading-order operation counts of the forward and backward passes, as well as the memory footprint of the optimizer state, also scale linearly with the number of subdomains. While selective evaluation of collocation points has been developed to reduce overhead in FBPINNs [18], implementing this in AB-PINNs is challenging because the subdomains corresponding to each subnetwork change dynamically during training. Incorporating ideas from the mixture-of-experts literature [56] to enable selective evaluation despite these dynamic subdomains is an important direction for future work. Moreover, our method introduces additional hyperparameters associated with the creation of subdomains, when to freeze the subdomain parameters, and the covariance of newly initialized window functions. Further investigation on how to adaptively choose these parameters is important to reduce hyperparameter tuning and improving the method's robustness.

In our numerical results, we have seen that AB-PINNs deliver substantial advantages over FBPINNs for strongly multiscale problems, such as the Helmholtz equation, forced Poisson equation, and chirp waveform. In contrast, for non-multiscale problems such as the uniform sine wave, a fixed, uniform domain decomposition is preferable. For intermediate cases, such as the KdV equation and Allen–Cahn equation, the gains made by the AB-PINN were marginal. It is therefore desirable to gain a deeper understanding of what classes of problems AB-PINNs are expected to excel at relative to fixed domain decomposition strategies, and how this performance depends on the number of subdomains as well as the approximation capacity of their corresponding subnetworks.

Finally, physics-informed losses are known to be effective for solving inverse problems, parametric families of PDEs, and operator learning tasks. Generalizing the AB-PINN architecture to these contexts and investigating its performance remains another important direction for future work.

7 Conclusion

We have introduced AB-PINNs, a simple framework for dynamically learning PINN-based domain decompositions for multiscale PDEs. Building off of the FBPINN framework for static PINN-based domain decompositions [18], our approach provides three main new contributions. First, the parameters of each FBPINN subdomain are made learnable during training and are updated via gradient-based methods. Thus, the domain decomposition evolves throughout training and adapts to the structure of the underlying PDE solution. Second, we introduce new adaptive subdomains in regions of high PDE residual. This effectively delivers localized expressive power in regions of the spatio-temporal domain where the solution of the PDE is challenging to represent and can help prevent PINNs from getting stuck in unwanted local minima. Finally, we include a global network which is primarily responsible for propagating information between subdomains and learning the low frequency behavior of the underlying solution. We have presented numerical experiments on a wide range of PDEs which collectively show that AB-PINNs can offer significant advantages compared to both standard PINNs and FBPINNs.

Acknowledgments

This work was partially completed while J. Botvinick-Greenhouse interned at Mitsubishi Electric Research Laboratories (MERL). J. Botvinick-Greenhouse was also supported in part by a fellowship award under contract FA9550-21-F-0003 through the National Defense Science and Engineering Graduate (NDSEG) Fellowship Program, sponsored by the Air

Force Research Laboratory (AFRL), the Office of Naval Research (ONR) and the Army Research Office (ARO). S. Mowlavi and W. H. Ali were supported solely by MERL. This work was done prior to M. Benosman joining Amazon Robotics.

References

[1] Zhilin Li, Zhonghua Qiao, and Tao Tang. *Numerical solution of differential equations: introduction to finite difference and finite element methods*. Cambridge University Press, 2017.

[2] Rene-Edouard Plessix. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, 167(2):495–503, 2006.

[3] Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019.

[4] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

[5] Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Solving the wave equation with physics-informed deep learning. *arXiv preprint arXiv:2006.11894*, 2020.

[6] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[7] Yeonjong Shin, Jérôme Darbon, and George Em Karniadakis. On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs. *Communications in Computational Physics*, 28(5), 2020.

[8] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating PDEs. *IMA Journal of Numerical Analysis*, 43(1):1–43, 2023.

[9] Ivan G Graham, Patrick O Lechner, and Robert Scheichl. Domain decomposition for multiscale pdes. *Numerische Mathematik*, 106(4):589–626, 2007.

[10] Ke Li, Kejun Tang, Tianfan Wu, and Qifeng Liao. D3M: A deep domain decomposition method for partial differential equations. *IEEE Access*, 8:5283–5294, 2019.

[11] Wuyang Li, Xueshuang Xiang, and Yingxiang Xu. Deep domain decomposition method: Elliptic problems. In *Mathematical and Scientific Machine Learning*, pages 269–286. PMLR, 2020.

[12] Ameya D Jagtap and George Em 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.

[13] Chuang Liu and HengAn Wu. cv-PINN: Efficient learning of variational physics-informed neural network with domain decomposition. *Extreme Mechanics Letters*, 63:102051, 2023.

[14] Júlia Vicens Figueres, Juliette Vanderhaeghen, Federica Bragone, Kateryna Morozovska, and Khemraj Shukla. $PINN-a Domain Decomposition Method for Bayesian Physics-Informed Neural Networks. *arXiv preprint arXiv:2504.19013*, 2025.

[15] Khemraj Shukla, Ameya D Jagtap, and George Em Karniadakis. Parallel physics-informed neural networks via domain decomposition. *Journal of Computational Physics*, 447:110683, 2021.

[16] Alena Kopaničáková, Hardik Kothari, George E Karniadakis, and Rolf Krause. Enhancing training of physics-informed neural networks using domain decomposition–based preconditioning strategies. *SIAM Journal on Scientific Computing*, 46(5):S46–S67, 2024.

[17] Axel Klawonn, Martin Lanser, and Janine Weber. Machine learning and domain decomposition methods-a survey. *Computational Science and Engineering*, 1(1):2, 2024.

[18] Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations. *Advances in Computational Mathematics*, 49(4):62, 2023.

[19] Victorita Dolean, Alexander Heinlein, Siddhartha Mishra, and Ben Moseley. Multilevel domain decomposition-based architectures for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 429:117116, 2024.

[20] Marsha J Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of computational Physics*, 53(3):484–512, 1984.

[21] Denise Burgarelli, Mauricio Kischinhevsky, and Rodney Josué Biezuner. A new adaptive mesh refinement strategy for numerically solving evolutionary PDE's. *Journal of Computational and Applied Mathematics*, 196(1):115–131, 2006.

[22] HS Tang, RD Haynes, and Guillaume Houzeaux. A review of domain decomposition methods for simulation of fluid flows: Concepts, algorithms, and applications. *Archives of Computational Methods in Engineering*, 28(3):841–873, 2021.

[23] Zihan Shao, Konstantin Pieper, and Xiaochuan Tian. Solving nonlinear pdes with sparse radial basis function networks. *arXiv preprint arXiv:2505.07765*, 2025.

[24] Amuthan A Ramabathiran and Prabhu Ramachandran. SPINN: Sparse, physics-based, and partially interpretable neural networks for PDEs. *Journal of Computational Physics*, 445:110600, 2021.

[25] Zhiwen Wang, Minxin Chen, and Jingrun Chen. Solving multiscale elliptic problems by sparse radial basis function neural networks. *Journal of Computational Physics*, 492:112452, 2023.

[26] Vikas Dwivedi, Balaji Srinivasan, Monica Sigovan, and Bruno Sixou. Kernel-Adaptive PI-ELMs for Forward and Inverse Problems in PDEs with Sharp Gradients. *arXiv preprint arXiv:2507.10241*, 2025.

[27] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

[28] Suchuan Dong and Naxian Ni. A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. *Journal of Computational Physics*, 435:110242, 2021.

[29] Natarajan Sukumar and Ankit Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022.

[30] Stefano Berrone, Claudio Canuto, Moreno Pintore, and Natarajan Sukumar. Enforcing dirichlet boundary conditions in physics-informed neural networks and variational physics-informed neural networks. *Heliyon*, 9(8), 2023.

[31] Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. An expert's guide to training physics-informed neural networks. *arXiv preprint arXiv:2308.08468*, 2023.

[32] Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023.

[33] Zhiwei Gao, Liang Yan, and Tao Zhou. Failure-informed adaptive sampling for PINNs. *SIAM Journal on Scientific Computing*, 45(4):A1971–A1994, 2023.

[34] Zhiping Mao and Xuhui Meng. Physics-informed neural networks with residual/gradient-based adaptive sampling methods for solving partial differential equations with sharp solutions. *Applied Mathematics and Mechanics*, 44(7):1069–1084, 2023.

[35] Yuling Jiao, Di Li, Xiliang Lu, Jerry Zhijian Yang, and Cheng Yuan. A gaussian mixture distribution-based adaptive sampling method for physics-informed neural networks. *Engineering Applications of Artificial Intelligence*, 135:108770, 2024.

[36] Levi D McClenny and Ulisses M Braga-Neto. Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474:111722, 2023.

[37] Guangtao Zhang, Huiyu Yang, Fang Zhu, Yang Chen, et al. Dasa-pinns: Differentiable adversarial self-adaptive pointwise weighting scheme for physics-informed neural networks. *Available at SSRN*, 2023.

[38] Sokratis J Anagnostopoulos, Juan Diego Toscano, Nikolaos Stergiopulos, and George Em Karniadakis. Residual-based attention in physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116805, 2024.

[39] Yanjie Song, He Wang, He Yang, Maria Luisa Taccari, and Xiaohui Chen. Loss-attentional physics-informed neural networks. *Journal of Computational Physics*, 501:112781, 2024.

[40] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116813, 2024.

[41] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[42] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[43] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–Arnold Networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

[44] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

[45] Zheyuan Hu, Ameya D Jagtap, George Em Karniadakis, and Kenji Kawaguchi. Augmented Physics-Informed Neural Networks (APINNs): A gating network-based soft domain decomposition methodology. *Engineering Applications of Artificial Intelligence*, 126:107183, 2023.

[46] Rafael Bischof and Michael A Kraus. Mixture-of-experts-ensemble meta-learning for physics-informed neural networks. In *Proceedings of 33. Forum Bauinformatik*, 2022.

[47] Patrick Stiller, Friedrich Bethke, Maximilian Böhme, Richard Pausch, Sunna Torge, Alexander Debus, Jan Vorberger, Michael Bussmann, and Nico Hoffmann. Large-scale neural solvers for partial differential equations. In *Smoky Mountains Computational Sciences and Engineering Conference*, pages 20–34. Springer, 2020.

[48] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[49] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

[50] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[51] Yongzheng Zhu, Shiji Zhao, Yuanye Zhou, Hong Liang, and Xin Bian. An unstructured adaptive mesh refinement for steady flows based on physics-informed neural networks. *arXiv preprint arXiv:2411.19200*, 2024.

[52] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proceedings of the Royal Society A*, 476(2239):20200334, 2020.

[53] Honghui Wang, Lu Lu, Shiji Song, and Gao Huang. Learning specialized activation functions for physics-informed neural networks. *arXiv preprint arXiv:2308.04073*, 2023.

[54] Sifan Wang, Bowen Li, Yuhan Chen, and Paris Perdikaris. Piratenets: Physics-informed deep learning with residual adaptive networks. *Journal of Machine Learning Research*, 25(402):1–51, 2024.

[55] Spyros Rigas, Michalis Papachristou, Theofilos Papadopoulos, Fotios Anagnostopoulos, and Georgios Alexandridis. Adaptive training of grid-dependent physics-informed kolmogorov-arnold networks. *IEEE Access*, 2024.

[56] Siyuan Mu and Sen Lin. A comprehensive survey of mixture-of-experts: Algorithms, theory, and applications. *arXiv preprint arXiv:2503.07137*, 2025.

[57] Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5:288–304, 2023.

[58] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[59] Robert A Adams and John JF Fournier. *Sobolev spaces*, volume 140. Elsevier, 2003.

[60] Art B Owen. Monte carlo theory, methods and examples, 2013.

[61] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[62] Colby L Wight and Jia Zhao. Solving Allen-Cahn and Cahn-Hilliard Equations Using the Adaptive Physics Informed Neural Networks. *Communications in Computational Physics*, 29(3):930–954, 2021.

[63] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

[64] Arka Daw, Jie Bu, Sifan Wang, Paris Perdikaris, and Anuj Karpatne. Mitigating Propagation Failures in Physics-informed Neural Networks using Retain-Resample-Release (R3) Sampling. In *International Conference on Machine Learning*, pages 7264–7302. PMLR, 2023.

[65] Norman J Zabusky and Martin D Kruskal. Interaction of "Solitons" in a Collisionless Plasma and the Recurrence of Initial States. *Physical review letters*, 15(6):240, 1965.