

ADAPTIVE PHYSICS-INFORMED NEURAL NETWORKS

ASSAF ZIMAND AND SHAI DEKEL

ABSTRACT. Physics-Informed Neural Networks (PINNs) provide a flexible framework for solving partial differential equations (PDEs). However, standard PINNs typically employ a fixed network architecture chosen a priori and optimize only the corresponding network weights. Such architectures may be inefficient when the complexity of the target solution varies substantially across the spatiotemporal domain. In this work, we introduce a method that adapts the network architecture during PINN training. Inspired by adaptive domain-decomposition methods and grounded in nonlinear approximation theory, our approach allocates network capacity according to the local complexity of the solution. Experiments on the Burgers, nonlinear Schrödinger, Allen–Cahn, Korteweg–de Vries and Kuramoto–Sivashinsky equations demonstrate that the method produces more parameter-efficient models while achieving accuracy comparable to or better than that of standard fixed-architecture PINNs.

1. INTRODUCTION

1.1. Motivation. Physics-informed neural networks (PINNs) have become a widely used framework for approximating solutions to forward and inverse problems governed by partial differential equations (PDEs). A major appeal of PINNs lies in their conceptual elegance and ease of implementation: governing equations and associated initial and boundary conditions are incorporated directly into the training objective, without requiring a problem-specific numerical discretization. However, standard PINNs can be inefficient when the complexity of the target solution varies substantially across the spatial or spatiotemporal domain, as occurs in the presence of sharp transition layers, rapidly oscillatory regions, or localized structures. A fixed global network architecture cannot explicitly adapt its representational capacity to these local variations. Consequently, accurately resolving the most complex regions may require a substantially larger standard PINN model, thereby increasing both training and inference costs.

This limitation motivates adaptive network architecture strategies that, similarly to domain decomposition methods, concentrate representational capacity in regions where the solution is more difficult to approximate. Nonlinear approximation theory provides a principled mathematical framework for constructing such adaptive partitions using hierarchical domain decompositions and tree-based representations. Guided by this framework, we propose the Adaptive Tree of Experts (AToE) PINN method that follows a hierarchical domain decomposition to adapt the network architecture to the local spatiotemporal complexity of the solution.

2020 *Mathematics Subject Classification.* 41A25, 41A46, 65M70, 68T07.

Key words and phrases. Physics-informed neural networks, domain decomposition, nonlinear approximation, Besov spaces, decision trees, mixture of experts.

1.2. Related Work. Physics-Informed Neural Networks (PINNs) were introduced by Raissi et al. (2019) [15]. Their training process, although in principle extremely elegant, presents several challenges, including competing loss terms and the need to compute higher-order derivatives, which can lead to instability and slow convergence [25, 23]. These difficulties have motivated a wide range of improvements to the standard PINN framework.

1.2.1. Domain decomposition methods for PINNs. A prominent line of work focuses on decomposing the computational domain into subregions. Early approaches include conservative PINNs (cPINNs) by Jagtap et al. (2020) [9], extended PINNs (XPINNs) by Jagtap et al. (2020) [8], hp-VPINNs by Kharazmi et al. (2021) [10], which decompose the test space rather than the network, and FBPINNs by Moseley et al. (2023) [14], which combine subdomain networks through a partition of unity, later extended to multilevel hierarchies of overlapping subdomains by Dolean et al. (2024) [5].

In all of these the partition is prescribed manually and held fixed throughout training, so it cannot reflect the structure of the solution and its resolution is limited by what the user specifies in advance. More recent methods introduce adaptive decomposition mechanisms, including APINNs by Hu et al. (2023) [7], PINN Balls by Bonfanti et al. (2025) [2], Pro-PINN by Wang et al. (2025) [27], and Progressive Domain Decomposition (PDD) by Luo et al. (2025) [12]. These methods dynamically allocate subnetworks to regions of high error or complexity, relying either on residual-based heuristics or on learned partitioning mechanisms.

1.2.2. Architectural improvements. Several works aim to enhance the expressive capacity of PINNs through architectural modifications. The use of random Fourier features by Tancik et al. (2020) [16] mitigates the spectral bias of neural networks and improves their ability to capture high-frequency components. More advanced architectures, such as PirateNet by Wang et al. (2024) [22], modify the standard network structure to improve training stability, especially for deeper models. Additional approaches, such as the random weight factorization of Wang et al. (2022) [26], aim to improve the network’s ability to represent complex solution patterns.

1.2.3. Adaptive sampling strategies. Adaptive sampling methods refine the distribution of the PINNs loss function’s collocation points during training. Residual-based adaptive refinement (RAR) and its variants, including residual-based adaptive sampling (RAD) by Wu et al. (2023) [29], focus computational effort on regions with large residuals. Variational residual-based adaptive approaches, such as vRBA by Toscano et al. (2026) [18], provide a variational interpretation of residual-based adaptivity and a principled basis for designing such schemes.

1.2.4. Loss balancing techniques. Balancing the different components of the PINN loss function is critical for stable training. Learning Rate Annealing (LRA), introduced by Wang et al. (2021) [25], and related adaptive weighting strategies dynamically adjust the contribution of residual, boundary, and initial condition terms to mitigate gradient pathologies and improve convergence.

1.2.5. Training and optimization strategies. Several works propose modifications to the training procedure itself. Causal training schemes introduced by Wang et al. (2024) [23] enforce temporal structure in time-dependent problems, while sequential time-marching approaches, such as those of Wight and Zhao (2021) [28]

and Matthey and Ghosh (2022) [13], decompose long time horizons into smaller subproblems solved in succession. More recently, optimization strategies have received increasing attention, with studies exploring advanced quasi-Newton methods and optimizer-switching schemes [11, 19]. For example, Kiyani et al. (2025) [11] demonstrated that optimization choices can substantially influence PINN convergence and accuracy, and Urbán et al. (2025) [19] showed that carefully optimized small networks can reach accuracies previously attributed to much larger models. Comprehensive benchmark studies, such as PINNacle by Hao et al. (2024) [6], provide systematic evaluations of these techniques.

While these approaches demonstrate solid empirical performance, many adaptive decomposition methods rely on trainable mechanisms, such as gating networks or learned weighting functions [7, 2, 27], to define the interaction between local experts. Although effective, such mechanisms can be difficult to interpret and may behave as black-box components in the decomposition process. In contrast, the present work proposes a decomposition strategy grounded in nonlinear approximation theory, providing a principled and interpretable framework for allocating computational resources.

1.3. Contributions. Our main contribution is the introduction of the Adaptive Tree of Experts (AToE) PINN method, which adapts the neural network architecture to the local regularity of the PDE solution. The AToE method is based on the well-established theory of nonlinear approximation and adaptive domain decomposition. It allows improved interpretability and control compared to the classic non-adaptive PINNs as well as compared to existing adaptive methods. Our model allocates network capacity according to local solution complexity, enabling efficient representations that use simpler sub-network models in smooth regions and higher capacity only where needed.

1.4. Organization of the Paper. The remainder of the paper is organized as follows. Section 2 introduces the theoretical and mathematical foundations. Section 3 presents the proposed AToE PINN method. In Section 4 we discuss additional techniques and variants that can further improve the performance of our approach. Numerical experiments are provided in Section 5, followed by discussion and conclusion.

2. BACKGROUND

2.1. PDE Benchmarks. In this work we evaluate the proposed AToE method on five standard benchmark problems commonly used in the PINN literature, each chosen to test a distinct source of difficulty. The solution to the Burgers equation develops a sharp front at small viscosity, testing the resolution of localized gradients. The solution to the Nonlinear Schrödinger (NLS) equation is complex valued, requiring a network architecture with two coupled outputs of the real and imaginary parts of the solution. Allen-Cahn is a stiff reaction diffusion problem that develops thin, steep transition regions. KdV is third-order and dispersive, while Kuramoto–Sivashinsky (KS) is fourth-order and exhibits chaotic dynamics. For each equation we select a demanding parameter regime, where the solution develops over time localized or sharp features. These are precisely the settings our adaptive PINN method targets, since it is designed to adapt its neural network architecture to the local structure of the solution.

Tables 1 and 2 summarize the governing benchmark equations, their domain and their initial and boundary conditions. The first group (Burgers, NLS, Allen–Cahn) involves derivatives up to second order, while the second (KdV, KS) involves third and fourth order. The higher order equations require more care during training, so we report them separately.

PDE	Equation	Domain	Initial Condition	Boundary Conditions
Burgers	$h_t + hh_x - \frac{0.01}{\pi} h_{xx} = 0$	$x \in [-1, 1]$ $t \in [0, 1]$	$h(x, 0) = -\sin(\pi x)$	$h(-1, t) = h(1, t) = 0$
NLS	$ih_t + \frac{1}{2} h_{xx} + h ^2 h = 0$	$x \in [-5, 5]$ $t \in [0, \pi/2]$	$h(x, 0) = 2 \operatorname{sech}(x)$	$h(-5, t) = h(5, t)$ $h_x(-5, t) = h_x(5, t)$
Allen–Cahn	$h_t - 10^{-4} h_{xx} - 5(h - h^3) = 0$	$x \in [-1, 1]$ $t \in [0, 1]$	$h(x, 0) = x^2 \cos(\pi x)$	$h(-1, t) = h(1, t)$ $h_x(-1, t) = h_x(1, t)$

TABLE 1. Second order benchmark PDEs.

PDE	Equation	Domain	Parameters	IC	BC
KdV	$h_t + hh_x + \mu^2 h_{xxx} = 0$	$x \in [-1, 1]$ $t \in [0, 1]$	$\mu = 0.022$	$h(x, 0) = \cos(\pi x)$	$h(-1, t) = h(1, t)$ $h_x(-1, t) = h_x(1, t)$ $h_{xx}(-1, t) = h_{xx}(1, t)$
KS	$h_t + \alpha h h_x + \beta h_{xx} + \gamma h_{xxxx} = 0$	$x \in [0, 2\pi]$ $t \in [0, 0.5]$	$\alpha = \frac{100}{76}$ $\beta = \frac{100}{16^2}$ $\gamma = \frac{100}{16^4}$	$h(x, 0) = \cos(x)(1 + \sin(x))$	$h(0, t) = h(2\pi, t)$ $h_x(0, t) = h_x(2\pi, t)$ $h_{xx}(0, t) = h_{xx}(2\pi, t)$ $h_{xxx}(0, t) = h_{xxx}(2\pi, t)$

TABLE 2. Higher order benchmark PDEs.

2.2. PDE Setting and PINN Formulation. Before we introduce our adaptive approach, we first provide a short overview of the classical PINN method for the numerical solution of PDEs. We consider a general partial differential equation of the form

$$\mathcal{N}[u](x, t) = 0, \quad (x, t) \in \Omega \times [0, T], \quad \Omega \subset \mathbb{R}^d,$$

subject to appropriate initial and boundary conditions.

In the PINN framework, the solution

$$u : \Omega \times [0, T] \rightarrow \mathbb{C}^m$$

is approximated by a neural network u_θ , trained by minimizing a loss function of the form

$$\mathcal{L}(\theta) = \lambda_r \mathcal{L}_r + \lambda_b \mathcal{L}_b + \lambda_i \mathcal{L}_i.$$

The three loss components are as follows. First, with randomly sampled collocation points $\{(x_j, t_j)\}_{j=1}^{N_r}$ in $\Omega \times (0, T)$, the residual PDE loss is

$$\mathcal{L}_r = \frac{1}{N_r} \sum_{j=1}^{N_r} |\mathcal{N}[u_\theta](x_j, t_j)|^2.$$

The boundary condition loss in the case of our benchmark PDEs, with a domain $\Omega = [a, b]$, is

$$\mathcal{L}_b = \frac{1}{NN_b} \sum_{m=0}^{N-1} \sum_{j=1}^{N_b} \left| \frac{\partial^m u_\theta}{\partial^m x}(a, t_j) - \frac{\partial^m u_\theta}{\partial^m x}(b, t_j) \right|^2,$$

where $\{(a, t_j), (b, t_j)\}_{j=1}^{N_b}$ are collocation points sampled at the boundary and N is the differential order of the equation (see Table 1 and Table 2). For Burgers, whose boundary data are homogeneous Dirichlet, the periodic differences are simply replaced by the sum $|u_\theta(a, t_j)|^2 + |u_\theta(b, t_j)|^2$.

Given an initial function $u(\cdot, 0) = f_0 : \Omega \rightarrow \mathbb{C}^m$, the initial condition loss is given by

$$\mathcal{L}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} |u_\theta(x_j, 0) - f_0(x_j)|^2,$$

where $\{x_j\}_{j=1}^{N_i}$ are collocation points sampled in Ω . In practice, the sampling sets associated with the boundary and initial condition losses are often relatively small compared to the residual points used for the loss $\mathcal{L}_r$.

A standard vanilla PINN u_θ approximates u using a fully connected multilayer perceptron (MLP). A real-valued network

$$u_\theta(x, t) : \Omega \times [0, T] \rightarrow \mathbb{R}^m, \quad \Omega \subset \mathbb{R}^d$$

consists of L hidden layers of width n with trainable parameters θ , where m denotes the dimension of the PDE solution. Setting $z^{(0)} = (x, t)$, each hidden layer applies an affine transformation followed by a nonlinear activation,

$$z^{(l)} = \sigma\left(W^{(l)}z^{(l-1)} + b^{(l)}\right), \quad l = 1, \dots, L,$$

where $W^{(1)} \in \mathbb{R}^{n \times (d+1)}$, $W^{(l)} \in \mathbb{R}^{n \times n}$ for $l = 2, \dots, L$, $b^{(l)} \in \mathbb{R}^n$. Since the network u_θ needs to be differentiable in its inputs, up to the corresponding orders of the equation, the nonlinear activation function is most commonly chosen as $\sigma(u) = \tanh(u)$, which is infinitely differentiable. The output is obtained by a final affine map without activation,

$$u_\theta(x, t) = W^{(L+1)}z^{(L)} + b^{(L+1)}, \quad W^{(L+1)} \in \mathbb{R}^{m \times n}.$$

A common enhancement of this architecture replaces the raw input with a random Fourier feature embedding [16]: the input $z^{(0)} = (x, t)$ is mapped to $[\cos(Bz^{(0)}), \sin(Bz^{(0)})]$, where B is a fixed, non-trainable random matrix whose entries are drawn from $\mathcal{N}(0, s^2)$ with a configurable scale s . The embedding mitigates the spectral bias of MLPs and improves their ability to capture high-frequency components of the solution; we employ it in our higher order pipeline (§3.2).

The depth L and width n vary across problems and training schemes, but a typical vanilla PINN uses on the order of 3–9 layers with a few dozen to a few hundred neurons per layer. The two ends of this range are illustrated by architectures discussed in §1: PirateNet [22] uses 3 residual blocks of 3 layers each with 256 neurons in every layer for harder PDEs, while Kiyani et al. [11] reach high accuracy with an MLP of 4 layers of width 20, in double precision, on the Burgers benchmark. The objective is to optimize the parameter vector

$$\theta^* \in \arg \min_{\theta} \mathcal{L}(\theta),$$

thereby obtaining a neural network that simultaneously satisfies approximately the PDE residual, initial conditions, and boundary conditions. The resulting network u_{θ^*} serves as the approximate solution to the underlying PDE.

2.3. Tree-Based Approximation. In contrast to linear approximation methods based on fixed bases or uniform partitions, nonlinear approximation techniques adapt the representation to the local structure of the function. Such methods are known to achieve improved approximation efficiency for functions with spatially varying regularity, by allocating resources non-uniformly across the domain (see, e.g., DeVore (1998) [4]).

We consider hierarchical partitions of domain $\Omega \times [0, T]$ generated by a decision tree and adapted to a discrete set of samples of $f \in L_p(\Omega \times [0, T])$. Starting from the root domain $\Omega \times [0, T]$, each subdomain is recursively divided into two child subdomains, using a local optimization method adapted to f .

The process continues until a stopping criterion is satisfied, such as a threshold on the approximation error or a minimal number of samples in each region. Each node in the tree $\eta \subset \Omega \times [0, T]$ is associated with a ‘local’ mean approximation to f

$$Q_\eta := \frac{1}{|\eta|} \int_\eta f(u) du,$$

where for a measurable $E \subset \mathbb{R}^d$, $|E|$ denotes the measure (volume) of E .

2.3.1. Geometric wavelets. Given a parent region η_p and a child region η_c from a tree $\mathcal{T}$, the associated geometric wavelet [3] is defined as

$$\psi_{\eta_c}(u) := (Q_{\eta_c} - Q_{\eta_p}) \mathbf{1}_{\eta_c}(u), \quad u \in \Omega \times [0, T],$$

where $\mathbf{1}_E$ is the indicator function over a domain $E \subset \mathbb{R}^d \times [0, T]$. The ‘base’ wavelet associated with $\Omega \times [0, T]$, is the constant function

$$\psi_{\Omega \times [0, T]} := Q_{\Omega \times [0, T]}.$$

The wavelet norm for $1 \leq p < \infty$ is

$$(1) \quad \|\psi_{\eta_c}\|_p = |Q_{\eta_c} - Q_{\eta_p}| |\eta_c|^{1/p},$$

which quantifies the contribution of the wavelet to the representation of the function f . As we shall see below, the norms are commonly used to quantify the function’s (weak-type) smoothness, guide adaptive refinement, compression, or pruning of the tree representation.

2.4. Sparse Wavelet Representations and Besov Spaces. Besov spaces provide a general framework for measuring smoothness that extends classical notions such as Sobolev and Lipschitz spaces, enabling the description of functions that are highly irregular and even discontinuous. Sparse Besov spaces provide a more flexible and applicable notion of smoothness that is better suited for higher dimensions. A sparse Besov space is typically associated with a decision tree or a forest of trees [1]. Let $\mathcal{T}$ be a decision tree associated with $f \in L_p(\Omega \times [0, T])$, $1 \leq p < \infty$ as in §2.3. The sparse Besov semi-norm of f associated with $\mathcal{T}$ and smoothness $\alpha > 0$ is equivalent to the sparsity of its wavelet representation [3]

$$|f|_{B_\tau^\alpha(\mathcal{T})} \sim N_\tau(f, \mathcal{T}).$$

Here

$$(2) \quad \frac{1}{\tau} = \alpha + \frac{1}{p},$$

and

$$N_\tau(f, \mathcal{T}) := \left(\sum_{\eta \in \mathcal{T}, \eta \neq \Omega \times [0, T]} \|\psi_\eta\|_p^\tau \right)^{1/\tau},$$

where the wavelet norms are computed as in (1) and the constants of equivalence do not depend on the function f or the tree $\mathcal{T}$.

Also, since $\ell_{\tau_2} \subset \ell_{\tau_1}$, for $\tau_2 < \tau_1$, using the equivalence with wavelet sparsity, one can see that $B_{\tau_2}^{\alpha_2}(\mathcal{T}) \subset B_{\tau_1}^{\alpha_1}(\mathcal{T})$ for $\alpha_1 < \alpha_2$, where a larger index α implies the function is ‘smoother’ in a weak sense.

2.5. Approximation Rates. The relationship between smoothness and wavelet sparsity leads to Jackson-type approximation estimates [3]. Let

$$(3) \quad S_M(f) := \sum_{k=0}^M \psi_{\eta_k}(f),$$

denote the M -term geometric wavelet approximation obtained by retaining the term $\psi_{\eta_0} := \psi_{\Omega \times [0, T]}$ and in addition the M largest geometric wavelet terms ordered by

$$\|\psi_{\eta_1}\|_p \geq \|\psi_{\eta_2}\|_p \cdots.$$

Then, under suitable assumptions,

$$\|f - S_M(f)\|_p \leq CM^{-\alpha} N_\tau(f, \mathcal{T}),$$

where α , τ and p satisfy (2) and C is independent of M and f .

This result implies that functions with higher weak-type smoothness α indices admit more accurate approximations using fewer significant tree nodes. Consequently, geometric wavelet norms and tree-based smoothness measures provide a natural criterion for guiding adaptive refinement and resource allocation. The estimate itself concerns the piecewise constant sum $S_M(f)$ over all selected nodes, whereas our method keeps only the leaves of the selected tree and represents each by a neural network expert rather than a constant, so the rate above motivates the decomposition but is not an error bound for the trained model. In Figures 1 and 2 below we provide examples of sparse wavelet approximations to several benchmark PDE solutions, grouped by the order of the underlying PDE. In both cases we see that regions of the solution with higher variation undergo additional refinement.

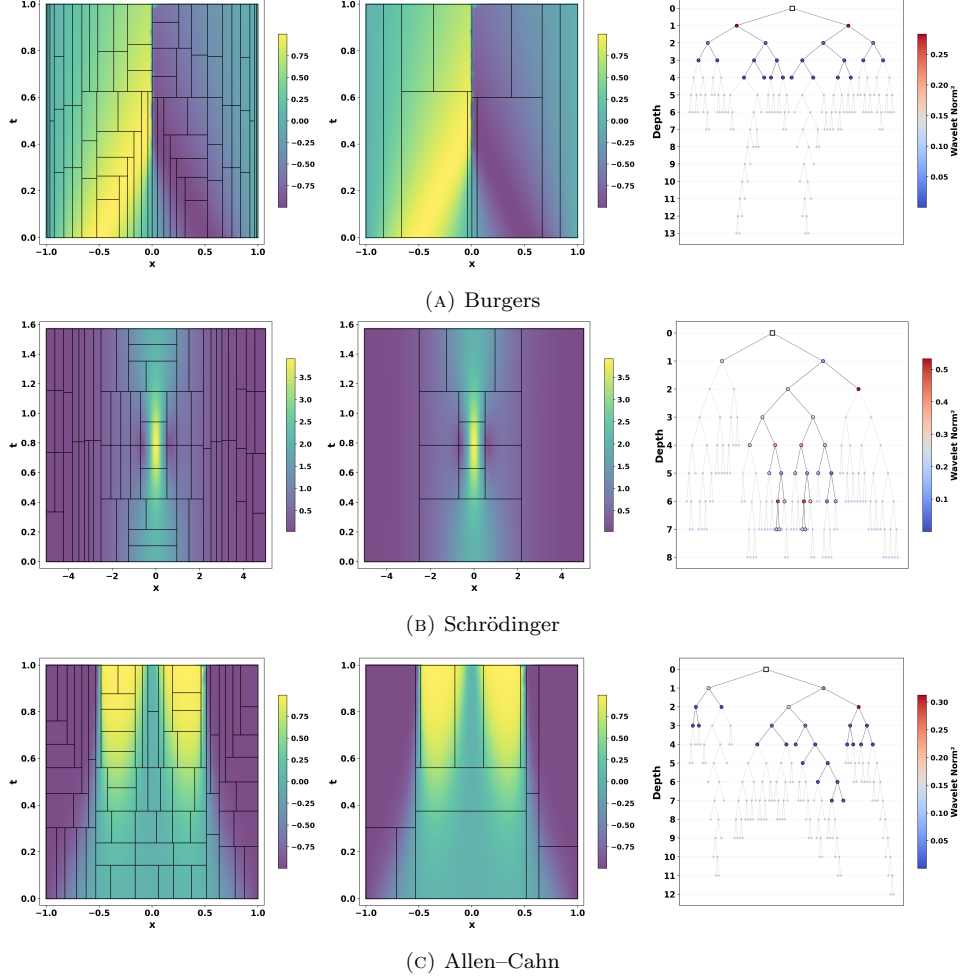


FIGURE 1. Tree-based approximation of ground truth solutions of the second order benchmark PDEs. For each PDE, the left panel shows the full decision tree built using 40k samples, with maximal depth 15. The middle panel shows a sparse approximation associated with $S_{20}(f)$, after retaining the $M = 20$ largest geometric wavelet terms along with their siblings and ancestors to preserve a full binary tree. The right panel shows the corresponding tree topological hierarchy.

The solutions of the higher order benchmarks evolve rapidly in time, with KS exhibiting chaotic dynamics. For such equations, a common practice in the PINN literature is sequential time marching [28, 13], in which the time horizon is divided into windows that are solved one after the other (see §3.2).

In nonlinear approximation, a known related construction is to partition the domain into subdomains and associate a separate tree with each one [1]. We find this construction useful for the higher order benchmarks, with a motivation that also builds on the time marching idea: the time axis is divided into equal-width tiles and an independent tree is grown inside each tile, as further explained in §3.3.

The M -term selection (3) is then performed over the nodes of all tile trees, while the root of every tile tree is always retained, so that the leaves of all tile trees together cover the entire space–time domain. Figure 2 depicts the resulting tiled trees.

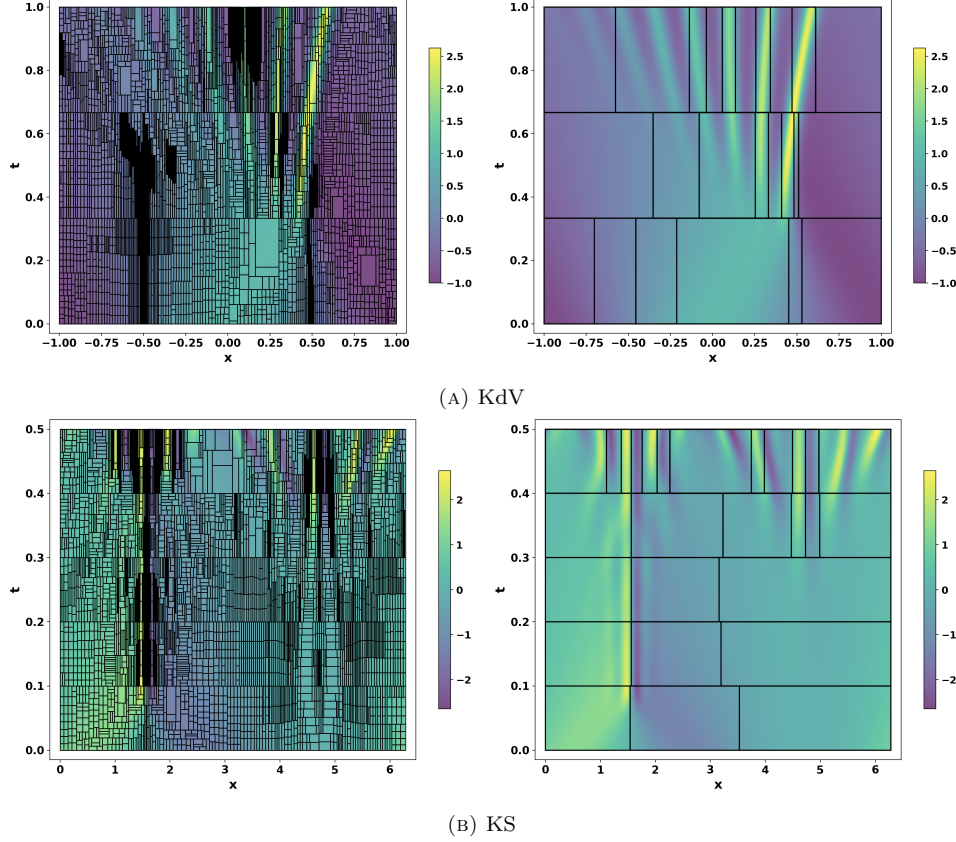


FIGURE 2. Tiled tree-based approximation of ground truth solutions of the higher order benchmark PDEs, as used in our pipeline: the time axis is divided into equal-width tiles (3 for KdV, 5 for KS, see §3.3) and an independent tree is built within each tile using 40k samples, with maximal depth 15. The left panel shows the full tile trees, and the right panel shows the sparse approximation obtained by retaining the $M = 25$ largest geometric wavelet terms pooled across the tile trees, along with their siblings and ancestors within each tile tree.

3. THE ADAPTIVE TREE OF EXPERTS (AToE) PINN METHOD

3.1. Overview of the Method. The tree based structure described in §2 forms a natural starting point for a wide range of domain decomposition designs that combine local models into a Mixture of Experts (MoE). In this section we present such a method which we refer to as AToE (Adaptive Tree of Experts). Section 4 describes additional variants of the main AToE algorithm that can further improve its performance. The full procedure is summarized in Algorithm 1, where each of its stages is described in detail in the subsections that follow.

Algorithm 1 AToE: Adaptive Tree of Experts

Require: PDE problem, template NN architecture $\mathcal{A}_{\text{base}}$, template expert sub-net architecture $\mathcal{A}_{\text{exp}}$, budget of sub-network local experts M

- 1: Train a vanilla PINN u_0 based on the $\mathcal{A}_{\text{base}}$ architecture
** higher order benchmarks: trained by sequential time marching*
- 2: Build an approximating tree to u_0 ; trim to an M -term full binary sub-tree
** higher order benchmarks: one tree per time tile, global M -term selection*
- 3: Record u_0 on the interfaces of the tree's leaves $\{\Omega_j\}$, $\Omega \times [0, T] = \cup_j \Omega_j$
- 4: **for all** leaf regions Ω_j **in parallel do**
- 5: Train an expert u_j with architecture $\mathcal{A}_{\text{exp}}$ on Ω_j , using the values of u_0 as sub-region boundary conditions
- 6: **end for**
- 7: Blend the experts via smoothstep POU (partition of unity)
- 8: Perform fine-tune training of the blended composition under the global PINN loss

Ensure: Global approximate solution u_θ

3.2. Learning the Initial Network. The goal of this initial step is to obtain a coarse approximation of the PDE solution that is sufficient to detect local spatiotemporal structures of the true solution.

The architecture and training process of the initial model u_0 follows the classic PINN formulation described in §2. For the second order benchmarks we use an MLP with 4 hidden layers of 60 neurons. Training consists of 1k Adam iterations with learning rate 10^{-3} , followed by 40k SSBroyden iterations, and all computations are performed in double precision, as suggested in [11].

For the higher order benchmarks we use larger networks: an MLP with 6 hidden layers of 128 neurons, combined with the random Fourier feature embedding of §2.2, with scale $s = 2$ and embedding dimension equal to the layer width. To keep a network of this size manageable, we train it in single precision. As suggested by recent literature, we compared a longer Adam phase followed by L-BFGS against a single run of the SOAP optimizer [20, 21], and found SOAP to be both more effective and more stable in this regime. We therefore train with 100k SOAP iterations, with learning rate 10^{-3} under a schedule of linear warm-up over the first 2k iterations, then exponential decay by a factor of 0.9 every 2k iterations, following the training recipe of [24].

For the higher order benchmarks, whose solutions accumulate error rapidly over a long horizon (dramatically so for the chaotic KS dynamics), the root itself is trained by the sequential time marching approach of [28]; advancing over successive time windows, rather than fitting the full horizon at once, is the common treatment of KS in the PINN literature. The time interval is divided into equal windows, 3 for KdV and 5 for KS, and an independent network with the higher order architecture above is trained on each window in order, with the converged prediction of the preceding network at the shared time slice serving as the initial condition of the next.

Each window is trained on a slightly extended interval, reaching 5% of a window length past its end (the last window is capped at the final time), while predictions and the handoff to the next window use only the original window; this places the error growth that a PINN accumulates near the end of its training interval beyond

the portion of the window that is actually used, a practice we adopt from the reference implementation accompanying [24].

For KS the Fourier feature embedding of the root is constructed to be exactly periodic in space, which we found necessary for obtaining a reasonable root for this equation. The resulting piecewise model over the windows serves as the root u_0 , and the remainder of the pipeline, the tree construction with time tiling, the expert training, and the global fine-tuning, proceeds unchanged.

In both regimes, all stages use full batch training with 20k residual points and 256 points each for the initial and boundary conditions; the residual set is redrawn at every first order iteration and held fixed during the quasi-Newton stages of the second order benchmarks; the higher order benchmarks redraw it at every iteration (excluding the brief L-BFGS fine-tuning stage of §3.6, during which it is held fixed). The loss weights are $\lambda_r = \lambda_{ic} = \lambda_{bc} = 1$ for the second order benchmarks, while the higher order ones use $\lambda_{ic} = \lambda_{bc} = 1000$ with $\lambda_r = 1$; these weights are kept unchanged across all training phases.

Typically, the initial approximation u_0 has large errors in spatiotemporal regions where the solution develops transients and shocks (see for example Figure 9(c) where a vanilla PINN has large errors around $x = 0$ at times $0.3 \leq t \leq 1$ for the Burgers equation).

3.3. Construction of the Approximating Tree and Domain Decomposition.

The goal of the decomposition is to partition the domain into regions that reflect the local complexity of the solution. A geometric wavelet tree $\mathcal{T}$ is constructed from the initial approximation u_0 and compressed using an M -term approximation (3). To preserve a clean hierarchical structure, both the ancestors and the siblings of the selected nodes are also retained, to form a connected binary sub-tree, in which every retained node has either zero or two children (the sub-tree is not necessarily balanced)

As introduced in §2, for higher order PDEs we tile the time axis into equal-width tiles and build independent trees over each tile. This tiling imposes a structure similar to sequential time marching [28]: each expert’s local initial condition originates at most one tile width in the past, keeping every local problem anchored to temporally nearby data.

We emphasize that even with tiling, all of the nodes of all of the trees over the separate tiles participate in a single ‘global’ M -term optimization, where the wavelets with largest norms over all tiles are retained. Each tile’s selection is then completed to a connected binary sub-tree, as before. In addition, the root of every tile is always retained, so that the representation is guaranteed to tile the entire space–time domain. The allocation of terms across the time segments is therefore not prescribed in advance but emerges from the solution itself: time segments over which the solution has more complexity will automatically be assigned more resources.

Each leaf region $\Omega_j \in \Lambda$ of the resulting tree is assigned a neural network expert u_j with the expert architecture $\mathcal{A}_{\text{exp}}$, which is smaller than the root architecture. The reduced size is possible without sacrificing accuracy because each expert approximates the solution only on its own subdomain, where by construction of the tree the solution has more uniform local regularity, rather than the full problem with all its variation.

Each expert is an MLP with 3 hidden layers of 24 neurons for the second order benchmarks, and, with the random Fourier feature embedding of §2.2, 3 hidden layers of 45 neurons for the higher order ones. In the experiments shown in Section 5 we set $M = 8$ for the second order PDEs and $M = 25$ for the higher order ones, with 3 time tiles for KdV and 5 for KS. Since the completion to a full binary tree adds or removes leaves relative to the selected terms, the resulting number of experts varies per problem (see Tables 3 and 4).

3.4. Interpretation of the Adaptive Decomposition. As discussed in §2.4, the equivalence between the Tree-Besov semi-norm and the geometric wavelet norm implies that the decay of geometric wavelet norms is directly related to the smoothness of the underlying function.

A large geometric wavelet norm at a single node does not necessarily imply that the corresponding region is non-smooth, since nodes at lower levels of the tree may simply capture large scale variations of the solution. Instead, smoothness is reflected in how rapidly the geometric wavelet norms decay along a branch of the tree. Regions associated with branches whose coefficients remain significant at increasing depths exhibit slower decay and therefore correspond to lower regularity and higher local complexity.

Viewed through this lens, the tree construction procedure described in §3.3 naturally favors the retention of branches that exhibit slow wavelet norm decay across multiple levels of the hierarchy. Consequently, the resulting decomposition concentrates refinement in the less smooth and more complex portions of the domain, while maintaining a coarser representation in regions where the wavelet norm decay is more rapid.

Extended methods that take further advantage of our theoretical setup are detailed in §4. However, even the simple choice of a fixed expert architecture already applies a form of adaptive capacity allocation. Regions associated with branches that persist to greater depths receive a larger number of experts and therefore a larger effective approximation capacity, whereas smoother regions are represented by shallower portions of the tree. As illustrated in Figure 3, this mechanism produces a nonlinear distribution of capacity across the domain that reflects the underlying complexity of the solution.

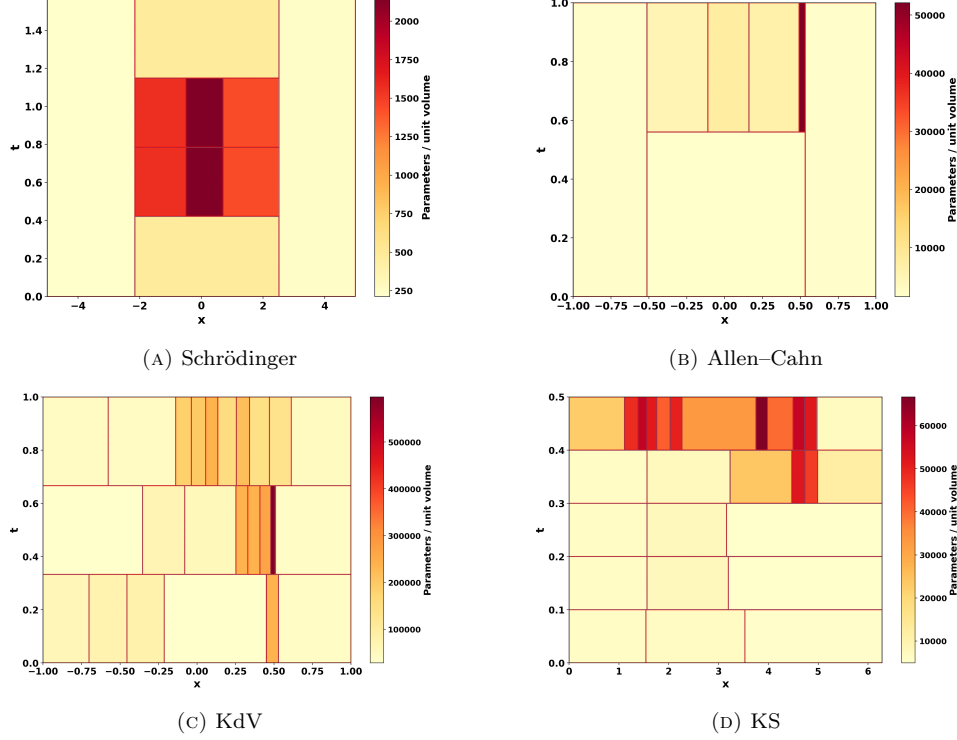


FIGURE 3. Effective approximation capacity induced by the adaptive decomposition relative to the subdomain volume for benchmark PDEs. Trees are constructed using the M -term procedure of Section 3.3 with $M = 8$ for the second order examples and, with time tiling, $M = 25$ for KdV and KS.

3.5. Expert Training. Before we assemble and fine-tune the experts together through the global loss terms, we exploit a core MoE idea and let each expert solve a localized, and therefore easier problem on its own subdomain. Inspired by the progressive scheme of [12], each local problem is made independent by using evaluations of u_0 and its partial derivatives as boundary conditions on the shared interfaces. Concretely, on each leaf subdomain Ω_j we minimize a local objective

$$(4) \quad \mathcal{L}_j = \lambda_r \mathcal{L}_j^{\text{res}} + \lambda_{ic} \mathcal{L}_j^{\text{ic}} + \lambda_{bc} \mathcal{L}_j^{\text{bc}} + \lambda_\Gamma \mathcal{L}_j^\Gamma,$$

where $\mathcal{L}_j^{\text{res}}$ enforces the PDE residual on collocation points inside Ω_j , and $\mathcal{L}_j^{\text{ic}}$, $\mathcal{L}_j^{\text{bc}}$ enforce the initial condition f_0 and given boundary conditions on the parts of $\partial\Omega_j$ that coincide with the boundary of Ω .

We now explain the loss term $\mathcal{L}_j^\Gamma$. Recall that $\Omega_j \subset \Omega \times [0, T] \subset \mathbb{R}^{d+1}$, so each leaf has $2d$ spatial faces and two temporal faces. The region Ω_j generally borders a different neighbor along each spatial dimension, so let $\mathcal{I}_j \subseteq \{1, \dots, d\}$ denote the spatial dimensions along which Ω_j has an interior neighbor, as opposed to coinciding with the physical boundary of Ω , and for $i \in \mathcal{I}_j$ let $X_i^{\Gamma_j}$ denote the sample points on the corresponding interior face of Ω_j . Let $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ denote the spatial coordinates and N the differential order of the PDE. On the face normal to x_i we match the value together with the derivatives of orders $m = 1, \dots, N - 1$.

Matching the value alone would pin down each expert only up to a mismatch in its derivatives at the interface, which the PDE operator differentiates further. Matching derivatives up to order $N - 1$ encourages approximate C^{N-1} agreement between neighboring experts, the level at which local solutions of an order- N equation compose into a global one. The agreement is only approximate, and it is the smooth blending of §3.6 that yields a globally smooth model. Since all interface data are generated by the single approximate solution u_0 , these conditions are mutually consistent by construction.

Writing $X_j^\Gamma = \bigcup_{i \in \mathcal{I}_j} X_i^{\Gamma_j}$ for the union over all such faces, the loss ties u_j and its derivatives, on every interior face, to the frozen initial prediction u_0 :

$$(5) \quad \mathcal{L}_j^\Gamma = \frac{1}{N|X_j^\Gamma|} \sum_{i \in \mathcal{I}_j} \sum_{(x,t) \in X_i^{\Gamma_j}} \sum_{m=0}^{N-1} \left| \frac{\partial^m u_j}{\partial x_i^m}(x, t) - \frac{\partial^m u_0}{\partial x_i^m}(x, t) \right|^2 + \frac{1}{|X_j^t|} \sum_{(x,t) \in X_j^t} |u_j(x, t) - u_0(x, t)|^2.$$

The second term treats the time dimension in the same way, with X_j^t sampling the lower temporal face of the leaf: the equations are first order in time, so only the value is matched and only on the lower face, while the upper face is left unconstrained, as in time marching. In this way every expert solves a localized problem whose interface data on the interior interfaces is supplied by the root, so the leaves can be trained independently, and in parallel, while remaining mutually consistent.

For the higher order benchmarks, two practical adjustments are made to the derivative matching in (5). In single precision, and with solutions whose derivative magnitudes grow steeply with order, the mismatch terms of orders $m \geq 2$ attain values that dominate the gradient budget and destabilize training. We therefore truncate the matching to the value and the first derivative, and divide the first derivative term ($m = 1$) of each expert j by the scale

$$(6) \quad s_{1,j} = 1 + \frac{1}{|X_j^\Gamma|} \sum_{i \in \mathcal{I}_j} \sum_{(x,t) \in X_i^{\Gamma_j}} \left| \frac{\partial u_0}{\partial x_i}(x, t) \right|^2,$$

computed once from the frozen root over the expert's interior spatial faces and held fixed during training, so that the derivative mismatch becomes a relative quantity of magnitude comparable to the value term. The $+1$ floor guarantees that normalization can only tame a large term, never amplify a small one.

A third adjustment is specific to KS. Enforcing the physical boundary conditions directly at the expert level, through the term $\lambda_{bc} \mathcal{L}_j^{\text{bc}}$ of (4), proved unstable for this chaotic equation, for the same reason that led us to build the periodicity into the root's architecture in §3.2. For KS we therefore remove this term and treat the faces of Ω_j that lie on the physical boundary exactly like the interior interfaces. Since u_0 is exactly periodic, the experts at both ends of the domain are tied to identical targets, and periodicity is inherited from the root up to the interface error.

The experts are trained with the same optimizer schedule as the root of the corresponding benchmark (§3.2). Each leaf draws a uniformly sampled point set, on the same resampling schedule as the root, whose size is proportional to its volume, subject to a minimum count, so that even thin leaves around a fine feature train on enough points to solve their local problem. The loss weights λ_r , λ_{ic} , and λ_{bc}

are kept from the root training, and the interface terms inherit them as well: the spatial-face term of (5) carries $\lambda_\Gamma = \lambda_{bc}$, while the time-face term (local initial condition) carries λ_{ic} . Examples of interface training samples for inner domain experts are shown in Figure 4.

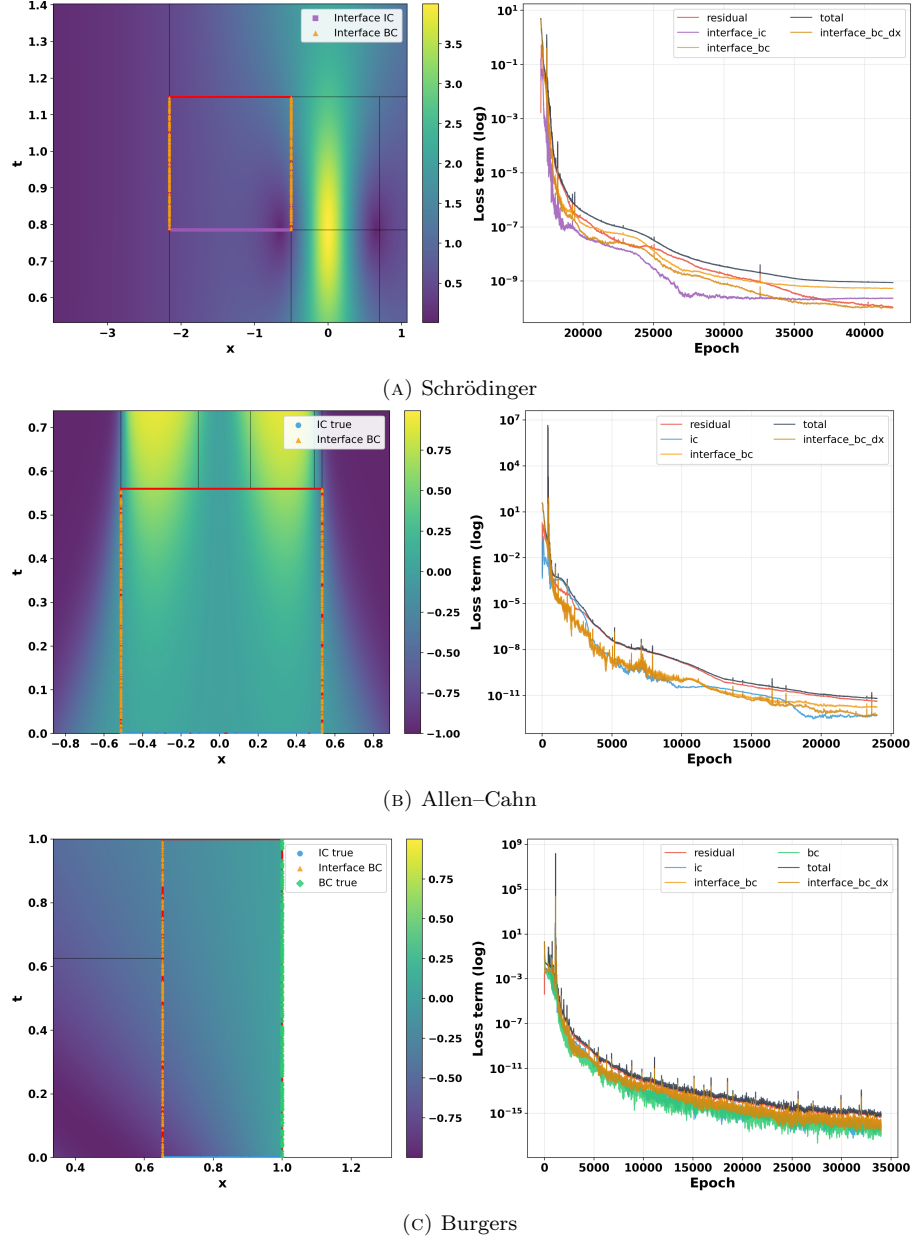


FIGURE 4. Examples of the localized training problem for individual experts, showing interface samples carrying data from the root prediction u_0 . Panels show an interior expert, an expert adjacent to the initial time, and an expert adjacent to the physical boundary.

3.6. Smooth Blending and Global Fine-Tuning. The trained leaf experts are combined into a single global approximation through smooth, compactly supported window functions similarly to FBPINNs [14]. Two differences are essential here: the decomposition is not a fixed uniform grid but based on the adaptive tree of §3.3, and the experts are first trained independently on local sub-problems before they are joined through a partition of unity and trained further under the global loss.

Let $\{\Omega_j\}_{j=1}^k$ be the set of leaf sub-domains obtained from the hierarchical decision tree decomposition, where

$$\Omega \times [0, T] = \cup_{j=1}^k \Omega_j$$

and each region is an axis-aligned hyperrectangle

$$\Omega_j = \left(\prod_{i=1}^d [a_{j,i}, b_{j,i}] \right) \times [a_{j,t}, b_{j,t}].$$

To smoothly blend the contributions of different experts while ensuring compatibility with the PDE's differential order, we define compactly supported window functions with configurable smoothness. Let N_w denote the window smoothness order, chosen such that $N_w \geq N$, the differential order of the PDE (e.g., $N_w \geq 2$ for Burgers or Allen–Cahn, $N_w \geq 3$ for KdV, $N_w \geq 4$ for KS). Since the domain decomposition operates over all dimensions including time, no dimension receives special treatment and the window must be sufficiently smooth for all derivatives.

We first define the smoothstep polynomial $S_{N_w} : [0, 1] \rightarrow [0, 1]$, which satisfies $S_{N_w}(0) = 0$, $S_{N_w}(1) = 1$, and $S_{N_w}^{(k)}(0) = S_{N_w}^{(k)}(1) = 0$ for $k = 1, \dots, N_w$:

$$(7) \quad S_1(t) = 3t^2 - 2t^3 \quad (C^1)$$

$$(8) \quad S_2(t) = 10t^3 - 15t^4 + 6t^5 \quad (C^2)$$

$$(9) \quad S_3(t) = 35t^4 - 84t^5 + 70t^6 - 20t^7 \quad (C^3)$$

$$(10) \quad S_4(t) = 126t^5 - 420t^6 + 540t^7 - 315t^8 + 70t^9 \quad (C^4)$$

The one-sided compact ramp function is defined as:

$$(11) \quad \rho_{N_w}(s) = \begin{cases} 0 & \text{if } s \leq 0 \\ S_{N_w}(s) & \text{if } 0 < s < 1 \\ 1 & \text{if } s \geq 1 \end{cases}$$

For region Ω_j with bounds $[a_{j,i}, b_{j,i}]$ along the i -th spatial coordinate, $1 \leq i \leq d$, or the time coordinate t , let $\delta_{j,i} = \kappa(b_{j,i} - a_{j,i})$ denote the overlap (transition) half-width, where $\kappa > 0$ is a configurable fraction. The one dimensional window along the i -th coordinate is:

$$(12) \quad \omega_{j,i}(z) = \rho_{N_w}\left(\frac{z - (a_{j,i} - \delta_{j,i})}{\delta_{j,i}}\right) \cdot \rho_{N_w}\left(\frac{(b_{j,i} + \delta_{j,i}) - z}{\delta_{j,i}}\right)$$

This produces a flat-top window: $\omega_{j,i}(z) = 1$ for $z \in [a_{j,i}, b_{j,i}]$, smooth C^{N_w} ramps in the overlaps $[a_{j,i} - \delta_{j,i}, a_{j,i}]$ and $[b_{j,i}, b_{j,i} + \delta_{j,i}]$, and exactly zero outside $[a_{j,i} - \delta_{j,i}, b_{j,i} + \delta_{j,i}]$. The window thus extends slightly beyond the leaf on which the expert was trained; in these overlap zones the expert extrapolates, and it is the global fine-tuning below that trains it there.

The region window is therefore the tensor product of the one dimensional windows over all coordinates:

$$(13) \quad \Psi_{\Omega_j}(x, t) = \omega_{j,t}(t) \prod_{i=1}^d \omega_{j,i}(x_i), \quad x = (x_1, \dots, x_d).$$

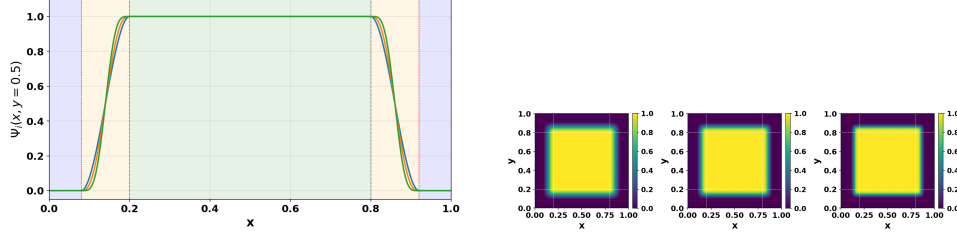


FIGURE 5. Compact smoothstep window functions (C^1 , C^2 , C^4 and overlap fraction $\kappa = 0.2$) for a representative domain decomposition. Left: one dimensional slices for several regions, showing the flat-top response and smooth overlap transitions. Right: two dimensional window heatmaps $\Psi_{\Omega_j}(x, t)$, illustrating the compact support and smooth blending overlaps.

Given the windows, the global solution of ATtoE over the set of leaves $\Lambda := \{\Omega_j\}$ of the trimmed tree is

$$(14) \quad u_\theta(x, t) = \sum_{\Omega_j \in \Lambda} \tilde{\Psi}_{\Omega_j}(x, t) u_j(x, t),$$

with the partition of unity

$$(15) \quad \tilde{\Psi}_{\Omega_j}(x, t) := \frac{\Psi_{\Omega_j}(x, t)}{\sum_{\Omega_k \in \Lambda} \Psi_{\Omega_k}(x, t)}.$$

Example normalized blending weights are shown in Figure 6.

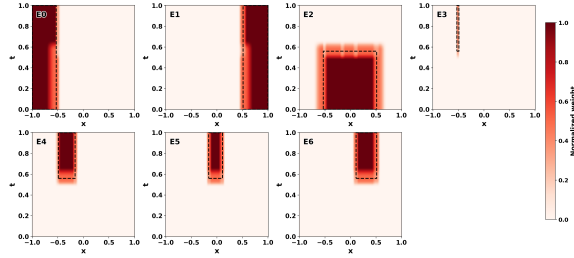


FIGURE 6. Example normalized blending weights for the ATtoE leaf composition. Dashed rectangles indicate expert region bounds. The leaf experts form a POU over the domain, each covering its region with smooth transitions in the overlap zones.

Finally, the full composition (14) is jointly fine-tuned under the global PDE loss. This last stage reconciles the independently trained experts in the overlap zones, producing a single, globally consistent solution. The fine-tuning stage consists of 40k SSBroyden iterations with no Adam warm-up for the second order benchmarks, and a short L-BFGS run with learning rate 10^{-4} for the higher order ones. In the single precision regime this final phase is less effective: the strong Wolfe line search

accepts a step only if it sufficiently decreases the training loss, and on the already accurate composed model it quickly stops finding such steps, so training terminates after few iterations. We found SOAP unsuitable for this final stage. Fine-tuning uses a sampling scheme that draws a fixed fraction of the collocation points from within the overlap regions; we use an overlap sampling fraction of 0.3, with an overlap fraction $\kappa = 0.03$ for all benchmarks.

Concentrating samples in the overlaps is motivated by two observations. First, these are the regions where the expert sub-networks trained on the boundary conditions provided by the initial coarse approximation u_0 . Second, typically, the tree approximation produces overlaps that are located exactly on sharp transitions or shocks of the solution. Thus, increased concentrated sampling at the overlaps produces a loss function that guides the training process to optimize the error on the difficult regions. This is a solution-driven counterpart to residual-based adaptive sampling methods such as RAD [29], with the sampling density fixed in advance by the decomposition rather than updated from residual during training.

4. ADDITIONAL METHODS AND VARIANTS

This section outlines alternative composition strategies and an alternative tree construction procedure that may replace or modify the corresponding components of the main algorithm of §3. In addition, we list established techniques from the PINN literature (see §1), that may improve the performance of the AToE method through enhanced training schemes and network design choices.

4.1. Alternative Compositions.

4.1.1. *Additive AToE (AAToE)*. A direct alternative strategy treats each expert as an additive residual correction on top of the root network u_0 , using all nodes of the trimmed tree rather than the leaves alone. This level-wise additive structure is related to the multilevel FBPINN of [5], with the difference that our levels are the nodes of a solution-adapted tree rather than a prescribed hierarchy of uniform grids.

Let D denote the depth of the trimmed tree, and for a node i let $\text{level}(i) \in \{0, \dots, D\}$ denote its level, with $\text{level}(i) = 0$ for the root. The approximate additive NN approximation u_θ is

$$(16) \quad u_\theta(x, t) = u_0(x, t) + \sum_{\ell=1}^D \sum_{i: \text{level}(i)=\ell} w_i(x, t) u_i(x, t)$$

where the per-level weights form a partition of unity over the experts at that level together with the root, whose constant background weight carries the remainder,

$$(17) \quad w_i(x, t) = \frac{\Psi_{\Omega_i}(x, t)}{1 + \sum_{k: \text{level}(k)=\text{level}(i)} \Psi_{\Omega_k}(x, t)}.$$

Because the root is retained in the composition, the siblings of the selected nodes need not be kept and any region left without a dedicated expert at a given level is approximated using the fine-tuning of u_0 . The tree may therefore be trimmed to the selected nodes alone, so that a level ℓ covers only part of the domain. Because the correction is additive and organized by level, this variant naturally admits a coarse-to-fine schedule in which one level of experts is trained at a time with the previous levels frozen, stopping the refinement once the target accuracy is reached. Alternatively, one may normalize the windows over the whole tree and train all

experts jointly in a single stage. Example normalized blending weights for the additive normalization are shown in Figure 7.

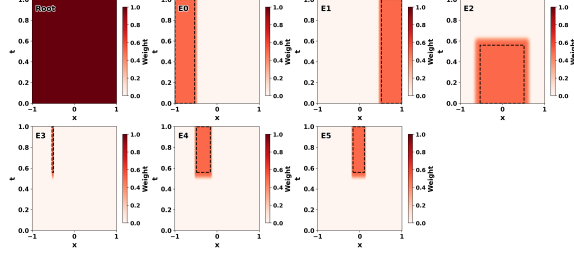


FIGURE 7. Example normalized blending weights for the AAToE composition. Dashed rectangles indicate expert region bounds. The base model retains a uniform contribution (Ψ_0 constant across the domain) while experts add localized corrections, each level normalized against a background of 1 to allow partial coverage.

4.1.2. Hierarchical Activation Propagation: ANT (Adaptive Neural Tree). A more expressive strategy lets information flow along the tree, following the Adaptive Neural Tree construction of Tanno et al. [17]. Instead of the raw coordinates, each expert receives the last hidden layer activation of its parent, $A_{\text{parent}(i)}$, and returns both a prediction u_i^* and an activation A_i passed on to its own children:

$$(18) \quad u_i(A_{\text{parent}(i)}(x, t)) = (u_i^*(x, t), A_i(x, t)), \quad A_{\text{parent}(\text{root})}(x, t) := (x, t).$$

The global solution is the POU combination over the leaves Λ ,

$$(19) \quad u_\theta(x, t) = \sum_{j \in \Lambda} \tilde{\Psi}_{\Omega_j}(x, t) u_j^*(x, t),$$

with the leaf normalization of (15).

By propagating activations down the tree, deeper regions are effectively represented by deeper, composed networks. The non-smooth, complex regions that attract additional refinement are therefore modeled by more expressive functions, while smoother regions remain shallow. This ties model capacity to local complexity. As in the additive case, one may also train the tree level by level: at each stage the current leaves act as the outputs, the levels that have finished training are frozen, and the next level of experts is grown on top of them. Unlike adaptive neural trees, where the routing of an input to a leaf is itself learned, here the assignment is fixed by the geometric decomposition and the leaves are combined through the partition of unity of §3.6. Visualizations of the AToE, AAToE, and ANT formulations are shown in Figure 8.

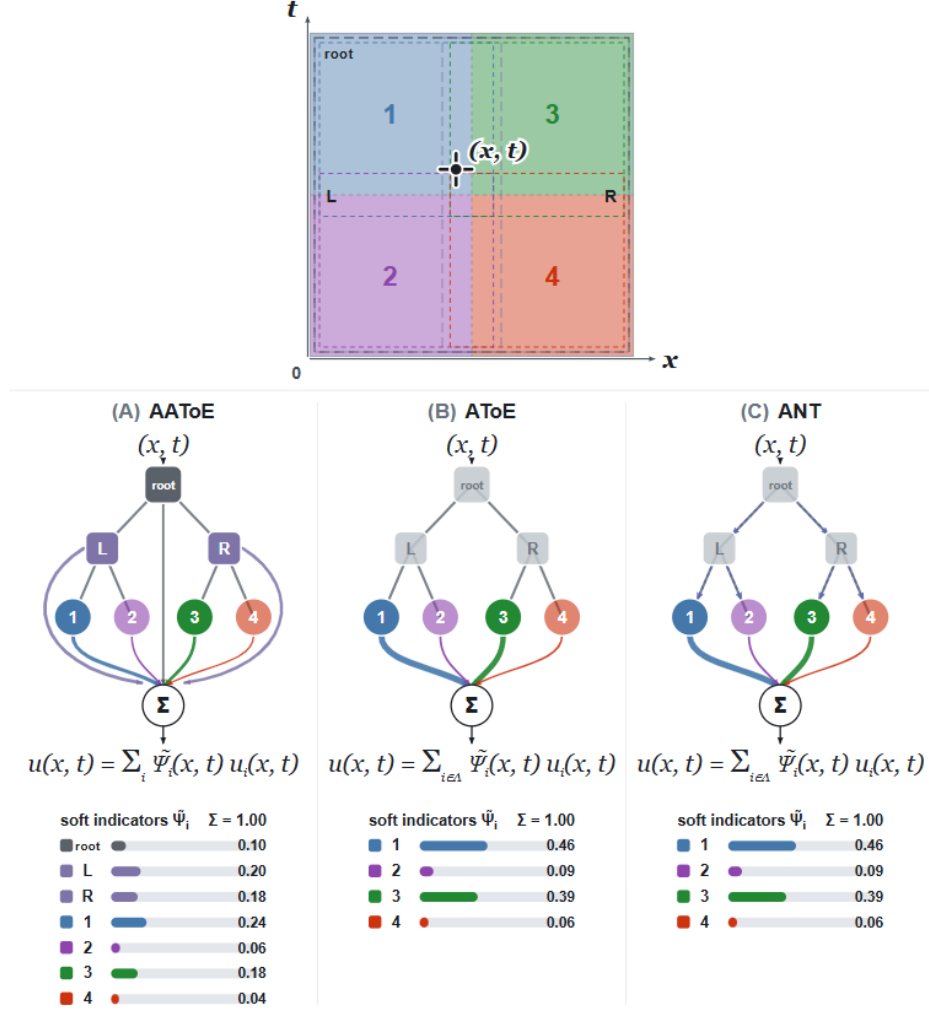


FIGURE 8. The AAToE (A), AToE (B), and ANT (C) formulations. Top: the tree decomposition of the domain, shared by all variants, with an evaluation point (x, t) in an overlap region. Bottom: for each variant, the flow from the evaluation point through the tree to the blended prediction, with the resulting normalized weights $\tilde{\Psi}_i(x, t)$. In AAToE all nodes contribute to the sum; in AToE only the leaves contribute; in ANT only the leaves contribute, each receiving its input through the chain of its ancestors.

4.1.3. Incremental Splitting. As an alternative to the M -term construction of §3.3, the tree may be grown incrementally during training. Training is performed for a fixed number of epochs (or until a convergence criterion), after which a single split is introduced by fitting a shallow decision tree of depth one. The split is accepted if it satisfies a norm based criterion motivated by the theory. This approach may be advantageous when an initial vanilla PINN is unable to capture sufficient structure of the solution to reliably estimate a complete tree.

This incremental approach with iterations between training and tree construction, is slower than the M -term construction over the initial solution u_0 . We also noticed

that the initial solution u_0 captures the main geometry of the ground truth solution u , providing a sufficient basis for the M -term method to produce good tree partitions. For these reasons, the M -term construction is adopted as the default in this work.

4.2. Adaptive Allocation of Expert Capacity. While the experiments in this work use a fixed MLP expert architecture, alternative backbones such as residual networks or PirateNet [22] may be used instead. More importantly, the size of each expert may differ between regions. The connection between geometric wavelet coefficients and smoothness suggests a second form of adaptivity beyond domain decomposition. Neural network capacity can be allocated according to local complexity, assigning experts with more capacity to regions associated with slower wavelet norm decay and smaller experts to smoother regions, as discussed in §3.4.

This behavior is analogous to classical approximation settings in which the local approximation order, such as the polynomial degree or the number of retained terms, is adapted according to the smoothness of the target function. Adaptive per-expert capacity allocation is left for future work.

4.3. Training Enhancements. The training scheme of §3 is deliberately plain. The following techniques from the PINN literature, introduced in §1.2, may be applied on top of it. Some are straightforward add-ons that do not interact with our algorithm in either direction. Others may combine with the decomposition in useful ways, and results may improve when they are applied at the per-expert level. In causal training, for example, each expert can maintain its own weighting of training samples, independently of the others. Exploring such combinations, and quantifying the effect of these techniques on top of the adaptive decomposition, is left for future work.

- Adaptive sampling strategies [29].
- Causal training for time-dependent problems [23].
- Sequential time-marching over long horizons [28, 13].
- Adaptive loss weighting (e.g., LRA) [25].
- Random weight factorization [26].

5. NUMERICAL EXPERIMENTS

5.1. Experimental Setup and Results. Tables 3 and 4 summarize the experimental configuration and results for each benchmark PDE, alongside a vanilla PINN baseline architecture for comparison. Accuracy is reported as the relative L_2 error,

$$\text{Rel-}L_2 = \frac{\|u_\theta - u\|_{L_2}}{\|u\|_{L_2}},$$

where u is the reference solution and the norms are evaluated on the reference solution grid.

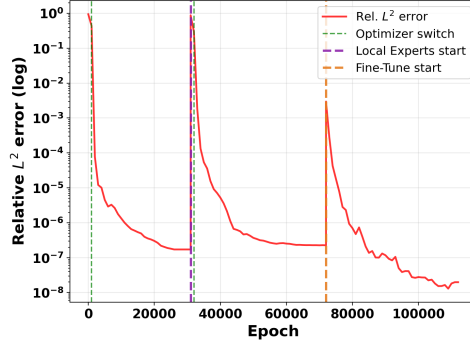
All runs follow the plain training scheme of §3; for the higher order benchmarks, the time tiling of §3.3 is used. The tables report all benchmarks; Figures 9 and 10 and Figure 11 show representative examples.

Ground-truth reference solutions are computed with high-accuracy spectral solvers: a Fourier pseudo-spectral discretization with ETDRK4 time-differencing for Allen–Cahn ($1,024 \times 500$ grid), KdV (512×500), and KS (512×500), a split-step Fourier scheme for NLS ($2,048 \times 1,000$) and the closed-form Cole–Hopf integral solution, evaluated by high-resolution quadrature, for Burgers. The accuracy of

the Burgers reference, estimated by quadrature refinement and by an independent extended-precision evaluation of the Cole–Hopf series, is at the level of double precision round-off, well below the errors reported below. All results were generated on an NVIDIA A10G GPU (23 GB VRAM). Both the vanilla and the AToE neural networks used 20,512 samples. However, as explained in §3.6, in our adaptive scheme, the collocation points were drawn randomly with a weighted probability giving more emphasis to the overlap regions of the experts where we expect the solution to have shocks, discontinuity, etc.

PDE	Vanilla Architecture (# Total Params)	AToE Architecture (# Total Params)	Vanilla Rel- L_2	AToE Rel- L_2
Burgers	MLP (4×60) (11,221)	6 × MLP (3×24) (7,782)	1.71×10^{-7}	1.30×10^{-8}
NLS	MLP (4×60) (11,282)	10 × MLP (3×20) (9,420)	3.85×10^{-4}	3.62×10^{-4}
Allen–Cahn	MLP (4×60) (11,221)	7 × MLP (3×24) (9,079)	5.24×10^{-5}	9.54×10^{-6}

TABLE 3. Experimental configuration and results for the benchmark PDEs, alongside a vanilla PINN baseline.



(A) Relative L_2 error during training, showing the three stages: the root u_0 , the local experts, and the composed model u_θ .

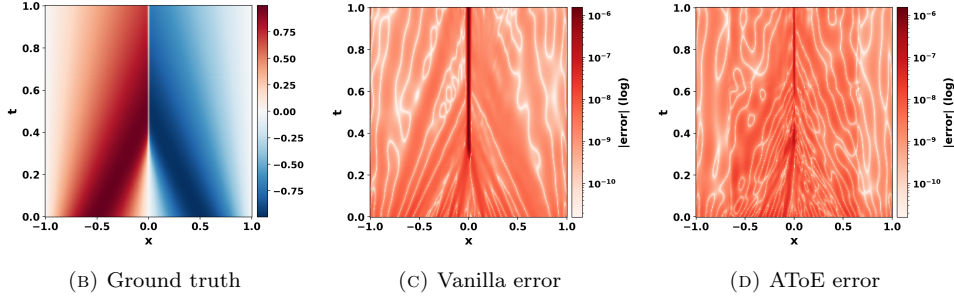
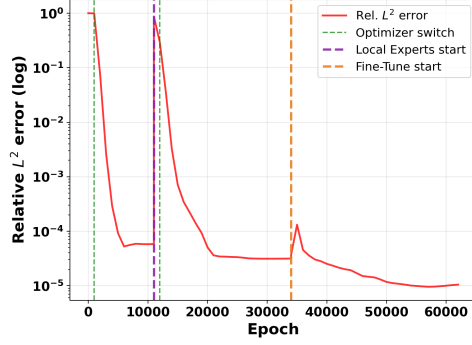


FIGURE 9. Burgers equation. Top: relative L_2 error during training. Bottom: the reference solution alongside the pointwise absolute errors of the vanilla model and the AToE model.



(A) Relative L_2 error during training, showing the three stages: the root u_0 , the local experts, and the composed model u_θ .

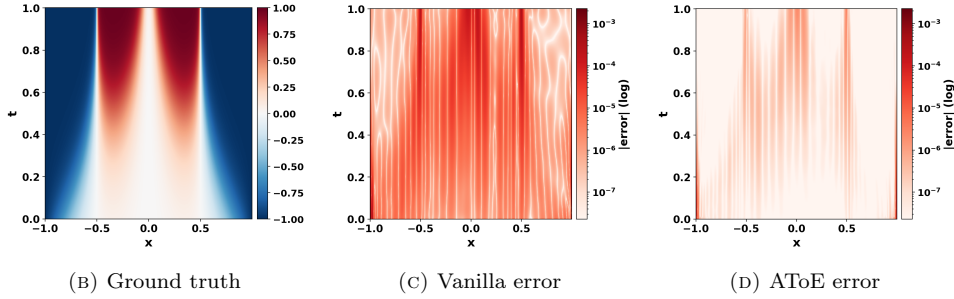


FIGURE 10. Allen-Cahn equation. Top: relative L_2 error during training. Bottom: the reference solution alongside the pointwise absolute error of the vanilla and the AToE models.

PDE	Optimizer (# Iterations)	Vanilla Architecture (# Total Params)	AToE Architecture (# Total Params)	Vanilla Rel- L_2	AToE Rel- L_2
KdV	SOAP (100k)	$3 \times \text{MLP} + \text{RFF}$ (6×128) (297,603)	$24 \times \text{MLP} + \text{RFF}$ (3×45) (149,064)	1.06×10^{-3}	6.26×10^{-4}
KS	SOAP (100k)	$5 \times \text{MLP} + \text{RFF}$ (6×128) (496,005)	$27 \times \text{MLP} + \text{RFF}$ (3×45) (167,697)	2.71×10^{-4}	2.70×10^{-4}

TABLE 4. Experimental configuration and results for the higher order benchmark PDEs, alongside a vanilla PINN baseline (the root model u_0) for comparison.

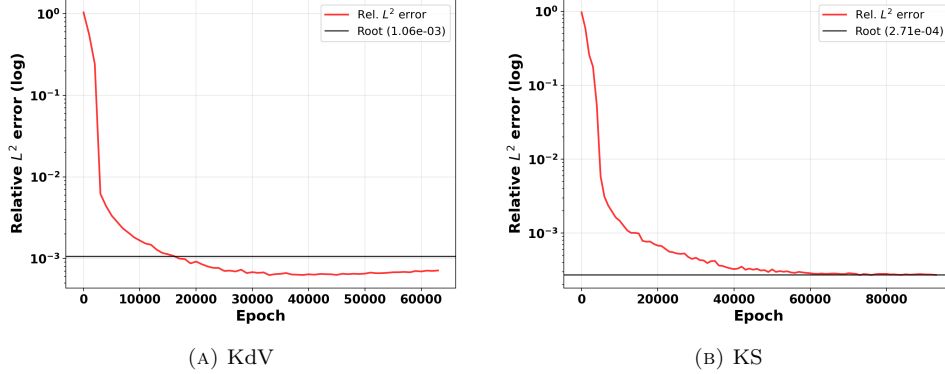


FIGURE 11. Relative L_2 error during the local expert training phase for the KdV and KS benchmarks. The black horizontal line marks the accuracy of the root model u_0 .

6. DISCUSSION

6.1. Summary of Findings. The experiments show that the AToE method allocates network capacity according to the local structure of the solution, attaining comparable or higher accuracy with less total network capacity than the non-adaptive baseline.

Furthermore, the AToE method is interpretable: the partition is neither prescribed by hand nor learned. This stands in contrast to methods with a prescribed partition, where the regions cannot follow the structure of the solution, and to methods that learn the partition through a gating network, where the resulting regions are encoded implicitly in trained weights. Here the criterion that selects the regions is the same wavelet decay that measures local smoothness, so the partition itself carries meaning.

Finally, the method can be viewed as a distillation-like compression scheme. The construction of the tree and of the interface data requires only the predictions of the initial model, not its architecture or training history, so any pretrained model of the solution can serve as the teacher. The result is a lighter adaptive architecture whose structure follows the local complexity of the teacher's prediction.

6.2. Future Work. Several promising research directions naturally arise from the proposed framework.

6.2.1. Architectural Variants and Training Strategies. Section 4 outlines several possible formulations, including AAToE and ANT, each inducing a different flow of information through the decomposition tree. Beyond the architectural design itself, these variants naturally motivate different training strategies, including staged, hierarchical, localized, or joint optimization schemes. A systematic study of the interaction between architecture and training methodology remains an important direction for future work.

6.2.2. Adaptive Capacity Allocation. While the present work employs a fixed expert architecture across all subdomains, the theoretical connection between geometric wavelet coefficients, smoothness, and approximation capacity suggests adapting the size of each expert according to the local complexity of the solution.

6.2.3. *Computational Efficiency.* The locality induced by the adaptive decomposition provides opportunities for significant computational savings. In particular, many formulations naturally localize the influence of each expert to a small subset of the computational domain, suggesting training and inference schemes in which only the relevant experts are evaluated for each sample. Such locality also opens the door to parallel execution of independent experts, improving scalability on modern parallel hardware.

7. CONCLUSION

We introduced the Adaptive Tree of Experts (AToE), a PINN method that adapts the network architecture to the local regularity of the PDE solution. Guided by nonlinear approximation theory, the method constructs a hierarchical decomposition of the space–time domain and assigns local expert networks to its regions, so that network capacity is allocated according to local solution complexity. The resulting decomposition is explicit and interpretable, and the numerical experiments show that it attains comparable or higher accuracy than a non-adaptive network of larger total capacity. Future work will focus on the architectural variants of §4, adaptive capacity allocation, and computational efficiency.

REFERENCES

1. I. Ben-Shaul, S. Dekel, and O. Elisha, *Sparse Besov space analysis of deep learning representation layers in high dimensions*, Pure and Applied Functional Analysis **10** (2025), 509–534.
2. A. Bonfanti, I. Medina, R. List, B. Staeves, R. Santana, and M. Ellero, *PINN Balls: Scaling second-order methods for PINNs with domain decomposition and adaptive sampling*, Advances in Neural Information Processing Systems (NeurIPS), 2025.
3. S. Dekel and D. Leviatan, *Adaptive multivariate approximation using binary space partitions and geometric wavelets*, SIAM Journal on Numerical Analysis **43** (2005), no. 2, 707–732.
4. Ronald A. DeVore, *Nonlinear approximation*, Acta Numerica **7** (1998), 51–150.
5. 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** (2024), 117116.
6. Z. Hao, J. Yao, C. Su, H. Su, Z. Wang, F. Lu, Z. Xia, Y. Zhang, S. Liu, L. Lu, and J. Zhu, *PINNacle: A comprehensive benchmark of physics-informed neural networks for solving PDEs*, Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, 2024.
7. Z. Hu, A. D. Jagtap, G. E. Karniadakis, and K. Kawaguchi, *Augmented physics-informed neural networks (APINNs): A gating network-based soft domain decomposition methodology*, Engineering Applications of Artificial Intelligence **126** (2023), 107183.
8. A. D. Jagtap and G. E. Karniadakis, *Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations*, Communications in Computational Physics **28** (2020), no. 5, 2002–2041.
9. A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, *Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems*, Computer Methods in Applied Mechanics and Engineering **365** (2020), 113028.
10. E. Kharazmi, Z. Zhang, and G. E. Karniadakis, *hp-VPINNs: Variational physics-informed neural networks with domain decomposition*, Computer Methods in Applied Mechanics and Engineering **374** (2021), 113547.
11. E. Kiyani, K. Shukla, J. F. Urbán, J. Darbon, and G. E. Karniadakis, *Optimizing the optimizer for physics-informed neural networks and Kolmogorov–Arnold networks*, Computer Methods in Applied Mechanics and Engineering **446** (2025), 118308.
12. D. Luo, S.-H. Jo, and T. Kim, *Progressive domain decomposition for efficient training of physics-informed neural network*, Mathematics **13** (2025), no. 9, 1515.

13. R. Matthey and S. Ghosh, *A novel sequential method to train physics informed neural networks for Allen–Cahn and Cahn–Hilliard equations*, Computer Methods in Applied Mechanics and Engineering **390** (2022), 114474.
14. B. Moseley, A. Markham, and T. Nissen-Meyer, *Finite basis physics-informed neural networks (FBPINNs): A scalable domain decomposition approach for solving differential equations*, Advances in Computational Mathematics **49** (2023), no. 4, 62.
15. M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics **378** (2019), 686–707.
16. M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, *Fourier features let networks learn high frequency functions in low dimensional domains*, Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020.
17. R. Tanno, K. Arulkumaran, D. C. Alexander, A. Criminisi, and A. Nori, *Adaptive neural trees*, Proceedings of the 36th International Conference on Machine Learning (ICML), Proceedings of Machine Learning Research, vol. 97, 2019, pp. 6166–6175.
18. J. D. Toscano, D. T. Chen, V. Oommen, J. Darbon, and G. E. Karniadakis, *A variational framework for residual-based adaptivity in neural PDE solvers and operator learning*, npj Artificial Intelligence **2** (2026), no. 1, 32.
19. J. F. Urbán, P. Stefanou, and J. A. Pons, *Unveiling the optimization process of physics-informed neural networks: How accurate and competitive can PINNs be?*, Journal of Computational Physics **523** (2025), 113656.
20. N. Vyas, D. Morwani, R. Zhao, M. Kwun, I. Shapira, D. Brandfonbrener, L. Janson, and S. Kakade, *SOAP: Improving and stabilizing Shampoo using Adam*, arXiv preprint arXiv:2409.11321 (2024).
21. S. Wang, A. K. Bhartari, B. Li, and P. Perdikaris, *Gradient alignment in physics-informed neural networks: A second-order optimization perspective*, arXiv preprint arXiv:2502.00604 (2025).
22. S. Wang, B. Li, Y. Chen, and P. Perdikaris, *PirateNets: Physics-informed deep learning with residual adaptive networks*, Journal of Machine Learning Research **25** (2024), no. 402, 1–51.
23. S. Wang, S. Sankaran, and P. Perdikaris, *Respecting causality for training physics-informed neural networks*, Computer Methods in Applied Mechanics and Engineering **421** (2024), 116813.
24. S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, *An expert’s guide to training physics-informed neural networks*, arXiv preprint arXiv:2308.08468 (2023).
25. S. Wang, Y. Teng, and P. Perdikaris, *Understanding and mitigating gradient flow pathologies in physics-informed neural networks*, SIAM Journal on Scientific Computing **43** (2021), no. 5, A3055–A3081.
26. S. Wang, H. Wang, J. H. Seidman, and P. Perdikaris, *Random weight factorization improves the training of continuous neural representations*, arXiv preprint arXiv:2210.01274 (2022).
27. Y. Wang, Y. Peng, Z. Hu, and Y. Li, *Exploring prototype-guided strategy for domain decomposition in physics-informed neural network*, Nonlinear Dynamics **113** (2025), 12487–12514.
28. C. L. Wight and J. Zhao, *Solving Allen–Cahn and Cahn–Hilliard equations using the adaptive physics informed neural networks*, Communications in Computational Physics **29** (2021), no. 3, 930–954.
29. C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. 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** (2023), 115671.

SCHOOL OF MATHEMATICAL SCIENCES, TEL AVIV UNIVERSITY, TEL AVIV, ISRAEL
 Email address: `assafzimand@gmail.com`

SCHOOL OF MATHEMATICAL SCIENCES, TEL AVIV UNIVERSITY, TEL AVIV, ISRAEL
 Email address: `shai@tauex.tau.ac.il`