HyResPINNs: A Hybrid Residual Physics-Informed Neural Network Architecture Designed to Balance Expressiveness and Trainability

Anonymous authors
Paper under double-blind review

Abstract

Physics-informed neural networks (PINNs) have emerged as a powerful approach for solving partial differential equations (PDEs) by training neural networks with loss functions that incorporate physical constraints. In this work, we introduce HyResPINNs, a two-level convex-gated architecture designed to maximize approximation expressiveness for a fixed number of degrees of freedom (DoF). The first level involves a trainable, per-block combination of smooth basis functions with trainable sparsity, and deep neural networks; the second involves the ability to gate entire blocks (much like in ResNets or Highway Nets), allowing for expressivity along the depth dimension of the architecture. Our empirical evaluation on a diverse set of challenging PDE problems demonstrates that HyResPINNs consistently achieve superior accuracy to baseline methods while remaining competitive relative to training times. These results highlight the potential of HyResPINNs to combine desirable features from traditional scientific computing methods and modern machine learning, paving the way for more robust and expressive approaches to physics-informed modeling.

1 Introduction

Partial differential equations (PDEs) underpin the simulation of complex physical phenomena across physics and engineering, from turbulent flows to electromagnetic waves and quantum systems (Evans, 2010; LeVeque, 2007). However, the interplay of nonlinearities, intricate geometries, and temporal dynamics often renders analytic solutions intractable, motivating a need for robust and adaptable numerical methods (Strang et al., 1974; LeVeque, 2007). While classical solvers—such as finite element, finite difference, and spectral methods—have established a foundation for computational science, recent advances in machine learning have led to physics-informed neural networks (PINNs), which offer a flexible, data-driven approach to PDE approximation (Raissi et al., 2019a; Karniadakis et al., 2021).

Despite these advances, a central challenge remains: how can we balance expressivity, adaptivity, and efficiency to obtain accurate, tractable solutions for increasingly complex problems? The impact of these three factors differs fundamentally between classical and machine learning (ML) paradigms, creating complementary strengths and weaknesses. In this work, we decompose expressivity into three interrelated components: *function space richness*—the breadth and diversity of approximable functions; *adaptivity*—the ability to allocate expressive power selectively (whether across input regions or within the model's internal structure); and *efficiency per DoF*—the contribution of each parameter to the overall approximation. These criteria jointly determine a solver's ability to capture complex solution structures while maintaining computational tractability. The central motivation of this study is to combine the controllable adaptivity and efficiency of classical methods with the richness and flexibility of neural networks, unifying their advantages within a hybrid framework.

Classical numerical methods offer rigorous convergence guarantees and explicit, controllable adaptivity—allowing targeted mesh or basis refinement (Ainsworth & Oden, 1997). Such refinement is often guided by prior knowledge of solution smoothness or domain geometry (Brenner & Scott, 2008; Strang et al., 1974), leading to high efficiency per DoF, where each additional parameter meaningfully improves accuracy. However, their function space richness is constrained to the chosen basis or kernel, limiting the diversity of representable functions without significant model redesign. Adaptive extensions—such as *hp*-adaptive finite elements (Schwab, 1998) or greedy kernel selection (Schaback & Wendland, 2000)—can expand or restructure the function space, but operate through discrete, manually designed rules within a fixed basis family. In contrast, ML-based neural architectures adjust capacity continuously during training, reweighting and recombining learned features without altering the underlying discretization. Further, maintaining accuracy in complex scenarios, such as nonsmooth features, irregular geometries, or high-dimensional interactions, escalates the required DoFs, resulting in increased computational costs (Babu**v**ska & Suri, 1994). Thus, while classical methods excel in efficiency and controllable adaptivity, they are less inherently flexible in representing highly varied solution behaviors.

Conversely, PINNs offer a mesh-free, data-driven approach that embeds physical laws directly into the neural network training. PINN approaches are promising due to their rich function spaces, enabling the approximation of complex, high-dimensional problems without the need for hand-crafted bases (Hornik et al., 1989), and without requiring a priori knowledge of solution structure. The overparameterization of neural architectures enables this flexibility, but concurrently reduces the efficiency per DoF such that many parameters contribute marginally to the final accuracy (Adcock & Dexter, 2021). Unlike classical methods, where DoF efficiency is linked to convergence theory, its quantification in neural architectures remains largely empirical. Moreover, their adaptivity is largely implicit, governed by global optimization rather than targeted refinement, making it challenging to resolve localized features, sharp gradients, or multiscale phenomena (Wang et al., 2021b). While adaptive PINN methods exist, achieving architecture-level, localized adaptivity comparable to classical numerical methods remains an open challenge.

A growing body of research seeks to combine the complementary strengths of classical and ML paradigms for PDE approximation. One class enriches neural networks with fixed structured bases—such as Fourier, polynomial, or radial basis functions (RBFs)—to improve efficiency per DoF or representation quality (Cooley et al., 2025b; Wang et al., 2021b; 2023b). While some adaptive basis methods exist, such as trainable modes in random Fourier feature embeddings (Wang et al., 2021b), most basis-enrichment strategies fix the basis set in advance and introduce adaptivity through pruning or regularization. This imposes a predefined representational structure that the model can only reduce rather than expand. In contrast to the incremental refinement strategies of classical methods, this separation of adaptivity from expressivity can limit efficiency per DoF, particularly early in training, when many basis functions contribute little to the solution, or when pruning strategies are insufficient. Another class draws on mesh-refinement ideas, reallocating model effort toward difficult regions via adaptive sampling (Zeng et al., 2022), domain decomposition (Jagtap & Karniadakis, 2020), or architecture adjustments (Luo et al., 2025). These approaches reallocate expressive power locally—either by concentrating the training signal in regions of high residual (adaptive sampling) or by statically partitioning the domain so that subregions have dedicated parameters (domain decomposition and mesh-inspired designs). As a result, function space richness remains unchanged, and adaptivity is achieved only indirectly through where and how the fixed capacity is applied—leaving a gap for methods that can dynamically expand and redistribute capacity while blending complementary function spaces throughout training.

We address this gap with HyResPINNs—a hybrid deep residual architecture that unifies function space richness, adaptivity, and efficiency per DoF in a single framework. Each residual block is a hybridization of two complementary function spaces: a trainable RBF neural network (RBFNN) that leverages compactly-supported RBFs for localized, structured refinement, and a deep neural network (DNN) for flexible, compositional expressivity. Their outputs are merged via a convex combination controlled by a trainable, interpretable gating parameter, allowing the model to adaptively shift between global and local modes over the course of training. A second gating parameter modulates the residual (skip) connection, starting at identity and progressively "opening" to activate additional depth only when needed, reducing early overparameterization. Stacking these hybrid blocks—that is, composing multiple blocks sequentially in depth—incrementally enriches the function space while concentrating refinement in specific solution regions, enabling fine-grained adaptivity and efficient parameter use—without reliance on static bases, coarse pruning, or fixed architectural capacity.

Our key contributions are as follows:

- **Expanded function space richness through hybridized composition.** We combine the global, compositional expressivity of deep neural networks with the localized approximation power of compactly-supported RBFs, enabling accurate representation of both smooth global structures and highly localized or discontinuous features across diverse PDE problems.
- **Fine-grained adaptivity via global–local blending and progressive depth activation.** Our architecture dynamically reallocates representational power during training—adjusting the balance between global and local modes and activating additional depth only when needed—to target refinement where it is most beneficial.
- **Improved efficiency per DoF through targeted capacity growth and localized enrichment.** HyResPINNs achieve competitive or superior accuracy with fewer, more effectively utilized parameters compared to static overparameterized neural PDE solvers by starting with minimal active capacity and selectively expanding it in response to solution complexity.

Through these contributions, we establish HyResPINNs as a practical approach for learning solutions to PDEs, providing a flexible framework that generalizes well across different problem domains. The remainder of this paper is organized as follows. In Section 2, we discuss related works and in Section 3, we review the relevant background. Then, in Section 4, we describe the mathematical formulation and architectural details of HyResPINNs. Next, in Section 5, we present numerical experiments comparing HyResPINNs to a suite of baseline neural PDE solvers, on a series of challenging PDE problems. We summarize our results and discuss future work in Section 6.

2 Related Work

In this section, we outline in more depth the approaches of classical numerical solvers and traditional PINNs. Then we review prior work in hybrid architectures that combine classical and ML paradigms, residual and stacked designs for progressive refinement, and radial basis function (RBF) methods and related kernel approaches. We organize this discussion to highlight how each class of methods addresses—or falls short of—the three critical components of expressive PDE solvers: (i) function space richness, (ii) adaptivity, and (iii) efficiency per degree of freedom. By situating HyResPINNs in relation to these prior works, we clarify both the limitations that motivate our framework and the distinct design principles that set it apart.

Classical Numerical Solvers. Classical numerical methods such as finite element (FEM) (Strang et al., 1974), finite difference (FDM) (LeVeque, 2007), and spectral approaches (Shen et al., 2011) have long provided a foundation for PDE solvers. These approaches employ carefully engineered discretizations or basis functions informed by a priori knowledge of solution smoothness and domain geometry (Brenner & Scott, 2008; Strang et al., 1974). They offer provable convergence rates and explicit adaptivity, allowing refinement to be directly targeted to specific regions of complexity. For instance, they can concentrate additional DoFs in regions with singularities or sharp gradients by refining the approximation through mesh, basis, or combined strategies (Ainsworth & Oden, 1997; Melenk, 1997). However, the required DoF can escalate rapidly in the presence of nonsmooth features, irregular geometries, or high-dimensional interactions (Schwab, 1998; Babuvska & Suri, 1994; Demkowicz et al., 2002; Bungartz & Griebel, 2004), resulting in significant computational costs.

Physics-Informed Neural Networks (PINNs). PINNs (Raissi et al., 2019a) approximate PDE solutions by incorporating physical laws into the neural network's loss function. PINNs achieve expressivity through learned, mesh-free representations, with function space richness emerging from network architecture, activation functions, and optimization processes (Hornik et al., 1989; Cybenko, 1989). Universal approximation theorems guarantee that, under certain architectural assumptions, neural networks can approximate any function arbitrarily well, suggesting their theoretical suitability for modeling complex PDEs. However, realizing this expressivity in practice is often difficult, with performance closely tied to the specific network architecture and optimization strategy (DeVore et al., 2020; Adcock & Dexter, 2021; Marwah et al., 2021). Standard PINNs exhibit implicit, global adaptivity: the network parameters are updated in response to the overall loss computed across the full domain, rather than through explicit, spatially localized refinement mechanisms typical in classical methods. Further, PINNs often perform poorly in regimes with steep gradients or multiscale features (Wang et al., 2022b). Recent approaches to ameliorating optimization issues include domain decomposition (X-PINNs) (Jagtap & Karniadakis, 2020), gradient-enhanced training (G-PINNs) (Yu et al., 2022), or discretely-trained PINNs using RBF-FD approximations in place of automatic differentiation (DT-PINNs) (Sharma & Shankar, 2022).

Hybrid Architectures. Efforts to hybridize PINNs with classical ideas often fall into two broad categories. The first are basis-enrichment methods which embed structured bases directly into neural architectures. Embedding known spectral or polynomial structure can yield high efficiency per DoF: each parameter has a targeted, interpretable role, and explicit representation of low- or high-frequency modes can offset the spectral bias of standard neural networks (Tancik et al., 2020; Cooley et al., 2025a). RBF-based methods, in contrast, target localized approximation and can provide improved computational efficiency when compactly supported kernels limit the region of influence (Wang et al., 2023b; Widrow & Lehr, 2002; Fasshauer, 2007). While some adaptive basis methods exist, such as trainable modes in random Fourier feature embeddings (Wang et al., 2021b), most basis-enrichment strategies fix the basis set in advance and introduce adaptivity through pruning or regularization. Across these methods, fixed, overcomplete basis sets are common, with adaptivity introduced only through pruning or regularization, which can decouple adaptivity from expressivity and limit efficiency per DoF when pruning is coarse.

Adaptivity-focused hybrids, in contrast, modify PINNs through adaptive sampling, domain decomposition, or architecture-level refinements inspired by mesh adaptation. These approaches reallocate expressive power locally—either by concentrating the training signal in regions of high residual (adaptive sampling) or by statically partitioning the domain so that subregions have dedicated parameters (domain decomposition and mesh-inspired designs). Recent efforts to introduce local adaptivity—including adaptive sampling (Zeng et al., 2022; Lu et al., 2021), loss reweighting (McClenny & Braga-Neto, 2020), and domain decomposition (Shukla et al., 2021; Jagtap & Karniadakis, 2020)—have improved performance in challenging regions, but these strategies still operate primarily at the data or loss level; the underlying parameter updates remain global, and truly local, mesh-like refinement within the network architecture remains largely unaddressed. Overall, most hybrids address only one axis of the expressivity–adaptivity–efficiency tradeoff at a time.

Residual and Stacked Architectures. Deep residual learning and stacked architectures have been recently introduced in scientific machine learning to overcome optimization challenges and enhance the expressivity of DNNs for solving PDEs. In particular, by enabling the training of deeper models and facilitating the learning of multi-scale phenomena through incremental refinement or additive corrections (He et al., 2016; Noorizadegan et al., 2024b). Residual connections allow deeper networks by alleviating gradient degradation, while stacked or multi-stage approaches iteratively refine solutions. These designs can increase function space richness without dramatically widening single models.

For instance, PirateNets (Wang et al., 2024) use multiple residual blocks with learned skip weights that modulate each block's influence. However, these residual weights are unconstrained and can grow arbitrarily large, complicating the interpretation of each block's non-linear transformation and potentially potentially destabilizing training. Stacked PINNs (Howard et al., 2025) introduce a multifidelity paradigm in which the output of each trained network serves as a low-fidelity input for the next, iteratively building up model accuracy while reducing the need for excessively large single networks. This framework incrementally increases depth by stacking new PINNs, but it does so through a sequential training process that requires manual intervention at each stage. Both methods introduce partial adaptivity—PirateNets modulate block contributions; stacked PINNs increase depth over stages—but neither couples adaptivity, efficiency, and richness in a principled way. Crucially, neither integrates complementary function spaces (e.g., local RBFs with global DNNs) to exploit different inductive biases within a unified refinement framework.

Radial Basis Function Networks and Kernel Methods. Radial basis functions (RBF) methods have long been used in scientific computing as a flexible, mesh-free alternative to grid-based discretizations, especially for scattered or irregular domains (Wu et al., 2012; Fasshauer, 2007). Many RBFs—especially those with compact support or rapidly decaying influence—enable local control in function approximation, as each basis function primarily affects a specific region of the domain. This locality allows RBFs to accurately represent local solution features with minimal disturbance to other regions (Wu et al., 2012; Widrow & Lehr, 2002; Chen et al., 2023). As a result, RBFs are particularly advantageous in problems where local refinement is critical, whereas global bases such as Fourier or polynomial functions are often more effective for smooth, globally distributed features but can struggle near sharp transitions due to the Gibbs phenomenon.

Despite their strengths for localized approximation, traditional RBFNNs, particularly those with fixed centers, encounter the curse of dimensionality when approximating general smooth functions in high dimensions (Wu et al., 2012). The number of required basis functions can increase exponentially with input dimension. This contrasts with three-layer multilayer perceptrons, where the number of hidden units grows polynomially due to their global parameterization (Barron, 2002; Wu et al., 2012). This limitation highlights a broader challenge in approximating complicated functions: balancing local accuracy with efficient global representation. While RBFs offer locality, the field has also explored how other paradigms, such as neural networks and kernel methods, expand representational capacity. Specifically, theoretical and practical connections have been established between neural networks and kernel methods (Arora et al., 2019; Li et al., 2020), inspiring feature lifting architectures such as random Fourier embeddings (Wang et al., 2021b; Li et al., 2020) that aim to efficiently expand representational capacity. Crucially, while RBFNN-based models are valued for their mesh-free construction and strong locality, they generally lack the hierarchical, compositional structure and global parameter sharing that characterize DNNs—a key advantage shared by many feature lifting methods in expanding representational capacity efficiently. Yet most retain a bias toward either predominantly local (RBF-like) or predominantly global (DNN-like) representation.

Positioning of HyResPINNs. HyResPINNs directly address the limitations identified above by embedding trainable, compact-kernel RBFNNs and DNNs within a unified residual block architecture. Two interpretable gating parameters per block control (i) the convex combination between local (RBF) and global (DNN) modes, and (ii) the activation of the residual (skip) path, allowing progressive depth activation during training. This enables HyResPINNs to dynamically grow and redistribute capacity while blending complementary function spaces—coupling function space richness, adaptivity, and efficiency per DoF in a single architecture. Unlike prior hybrids, this approach jointly targets all three axes, enabling fine-grained, architecture-level refinement without manual basis selection, coarse pruning, or static overparameterization.

3 Background

In this section, we review the core components underlying HyResPINNs: physics-informed neural networks (PINNs) as the baseline neural PDE framework, residual network architectures for progressive refinement, and radial basis function neural networks (RBFNNs) for localized approximation.

3.1 Physics-Informed Neural Networks.

Given a spatio-temporal domain $\mathcal{D} = [0, T] \times \Omega \subset \mathbb{R}^{1+d}$, where Ω is a bounded spatial domain in $\mathbb{R}^d$ with boundary $\partial\Omega$, the general form of a parabolic PDE is:

$$u_t + \mathcal{F}(u, \nabla u, \dots) = f, \tag{1}$$

where $\mathcal{F}$ is a linear or nonlinear differential operator, and u denotes the unknown solution. This PDE is subject to an initial condition $u(0, \mathbf{x}) = g(\mathbf{x})$ for $\mathbf{x} \in \Omega$, and boundary conditions $\mathcal{B}(u(t, \mathbf{x})) = 0$ for $t \in [0, T]$ and $\mathbf{x} \in \partial\Omega$. Here, f and g are given functions, and $\mathcal{B}$ denotes an abstract boundary operator.

The general formulation of PINNs (Raissi et al., 2019a) aim to approximate the unknown solution $u(t, \mathbf{x})$ using a representation model $u_\theta(t, \mathbf{x})$, such that $u_\theta(t, \mathbf{x}) \approx u(t, \mathbf{x})$ and θ denotes the set of all trainable parameters of the network. The parameters θ, are optimized to minimize the composite loss function:

$$L(\theta) = \lambda_{ic} L_{ic} + \lambda_{bc} L_{bc} + \lambda_r L_r, \tag{2}$$

where L_{ic}, L_{bc}, and L_r represent the loss components associated with the initial conditions, boundary conditions, and the residual of the PDE, respectively; and λ_{ic}, λ_{bc}, and λ_r are corresponding weighting coefficients that balance the contribution of each loss term. These terms can be set as static weights or adapted during training using various techniques such as NTK weighting (Wang et al., 2021b), gradient annealing (Wang et al., 2021a) among others.

Each loss term in Equation 2 is computed as the mean squared error of the initial condition, boundary, and PDE residuals. Specifically, each loss term is defined as:

$$L_{ic} = \frac{\lambda_{ic}}{N_{ic}} \sum_{i=1}^{N_{ic}} \left| u_\theta(0, \mathbf{x}_{ic}^i) - g(\mathbf{x}_{ic}^i) \right|^2, \tag{3}$$

$$L_{bc} = \frac{\lambda_{bc}}{N_{bc}} \sum_{i=1}^{N_{bc}} \left| \mathcal{B}[u_\theta](t_{bc}^i, \mathbf{x}_{bc}^i) \right|^2, \tag{4}$$

$$L_r = \frac{\lambda_r}{N_r} \sum_{i=1}^{N_r} \left| \frac{\partial u_\theta}{\partial t}(t_r^i, \mathbf{x}_r^i) + \mathcal{F}[u_\theta](t_r^i, \mathbf{x}_r^i) - f(t_r^i, \mathbf{x}_r^i) \right|^2. \tag{5}$$

The training data points $\{\mathbf{x}_{ic}^i\}_{i=1}^{N_{ic}}$, $\{t_{bc}^i, \mathbf{x}_{bc}^i\}_{i=1}^{N_{bc}}$ and $\{t_r^i, \mathbf{x}_r^i\}_{i=1}^{N_r}$ can be fixed mesh or points randomly sampled at each iteration of a gradient descent algorithm.

Variants of PINNs extend this basic formulation to improve stability, scalability, or compatibility with specific PDE structures. Specifically, Karniadakis and collaborators have extended these methods to conservative PINNs (cPINNs) (Jagtap et al., 2020), variational PINNs (vPINNS) (Kharazmi et al., 2019), parareal PINNs (pPINNs) (Meng et al., 2020), stochastic PINNs (sPINNs) (Zhang et al., 2019), fractional PINNs (fPINNs) (Pang et al., 2018), LesPINNs (Yang et al., 2019), non-local PINNs (nPINNs) (Pang et al., 2020) and eXtended PINNs (xPINNs) (Jagtap & Karniadakis, 2020). While differing in implementation details, all variants rely on the ability of neural networks to represent complex solution spaces without manually designed bases, and all face the challenge of efficiently directing model capacity toward localized, high-complexity regions. In this work, we will focus on application of the original collocation PINNs approach; however, the work presented herein can be applied to many if not all of these variants.

3.2 Residual Networks

DNNs often encounter optimization challenges in deep architectures such as vanishing or exploding, resulting training performance degradation. Residual Networks (ResNets) address these issues by introducing skip connections (He et al., 2016), which allow information to flow directly from an earlier layer to a later layer, creating an identity mapping that ensures information is preserved even as the network deepens (Noorizadegan et al., 2024a). Formally, a standard residual block is expressed as:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \mathbf{W}_i) + \mathbf{x}, \tag{6}$$

where $\mathbf{x}$ represents the input to the block, $\mathcal{F}$ is the learned residual mapping, and $\mathbf{y}$ is the block output.

These connections improve gradient flow, enable stable training of deeper models, and allow the network to learn incremental refinements rather than full transformations from scratch (Noorizadegan et al., 2024a). In the PDE context, this incremental refinement aligns naturally with multi-scale and hierarchical solution structures, making residual architectures a useful foundation for adaptivity in depth. Recent work has explored variants with trainable skip weights, allowing the effective network depth to evolve during training, though often without constraints that ensure interpretability or stability (Wang et al., 2024).

3.3 Radial Basis Function Neural Networks

A Radial Basis Function Neural Network (RBFNN) is typically a shallow network with a single hidden layer of RBF units followed by a linear output layer. The output of the network is a linear combination of the

Name	**Formula** $\psi(r)$	**Support**	**Smoothness**	**Notes**
Gaussian	$\exp\left(-\frac{1}{2}r^2\right)$	Global	C^∞	σ controls width
Multiquadric	$\sqrt{r^2+c^2}$	Global	C^∞	c controls shape; non-decaying
Inverse MQ	$\frac{1}{\sqrt{r^2+c^2}}$	Global	C^∞	Long-range; c sets shape
Matérn-5/2	$\left(1+\sqrt{5}r/\ell+\frac{5}{3}(r/\ell)^2\right)e^{-\sqrt{5}r/\ell}$	Global	C^2	ℓ is the lengthscale
Wendland C^2	$(1-\frac{r}{\rho})^6(35(\frac{r}{\rho})^2+18(\frac{r}{\rho})+3),\ r<\rho$	Compact	C^2	ρ sets support radius

Table 1: Common radial basis functions $\psi(r)$ for RBFNNs. The distance $r(\mathbf{x}, \mathbf{x}^c; \omega_j)$ may be isotropic, anisotropic diagonal, or fully anisotropic (Mahalanobis), with ω_j denoting shape parameters (e.g., σ, c, ℓ, ρ) for the j-th RBF.

kernel activations:

$$f(\mathbf{x}) = \sum_{j=1}^{N_c} c_j \ \psi(r(\mathbf{x}, \mathbf{x}_j^c;\ \omega_j)), \tag{7}$$

where $\mathbf{x}_j^c$ are center points, c_j are coefficients, and ψ is a chosen radial basis function kernel (see Table 1 for examples). The influence of each RBF is determined by a distance metric r between the input $\mathbf{x}$ and the j-th center $\mathbf{x}_j^c$, with additional shape parameters ω_j (e.g., width or lengthscale). One metric is the isotropic Euclidean distance:

$$r = \frac{||\mathbf{x} - \mathbf{x}_j^c||}{\sigma_j}, \tag{8}$$

which applies a uniform scale in all directions. Anisotropic variants allow direction-dependent scaling, such as:

$$r = \sqrt{\sum_{i=1}^{d} \frac{(x_i - x_{j,i}^c)^2}{\sigma_{j,i}^2}}, \tag{9}$$

for axis-aligned ellipsoidal support (per-dimension widths $\sigma_{j,i}$), or more generally the Mahalanobis distance:

$$r = \sqrt{(\mathbf{x} - \mathbf{x}_j^c)^\top A_j (\mathbf{x} - \mathbf{x}_j^c)}, \tag{10}$$

using a symmetric positive-definite matrix A_j for arbitrary orientation.

The specific choice of ψ determines each basis function's support, smoothness, and decay—for example, Gaussians are globally supported and C^∞ smooth, while Wendland functions are compactly-supported and less smooth. The shape parameters (σ, c, A, etc.) control locality and directional influence, and are often initialized to reasonable heuristics (e.g., based on data spread or uniformly), but can be further learned from data during training. RBFs with rapid decay or compact support intrinsically localize their influence, enabling accurate modeling of fine-scale features while minimizing impact on distant regions (Wu et al., 2012).

Summary. PINNs provide flexible, mesh-free function approximations; ResNets offer a mechanism for incremental refinement and stable training at depth; and RBFNNs give localized, interpretable basis functions well matched to targeted refinements. Together, these components form a foundation for architectures that can represent both global and local solution structures while controlling where and how model capacity is used.

4 Hybrid Residual PINNs (HyResPINNs)

We propose HyResPINNs, a deep learning framework that integrates radial basis function neural networks (RBFNNs) and deep neural networks (DNNs) within a hybrid residual architecture. This hybridization is designed to enhance function approximation for PDEs by combining the adaptive locality potential of RBFs with the expressive, hierarchical modeling capacity of DNNs.

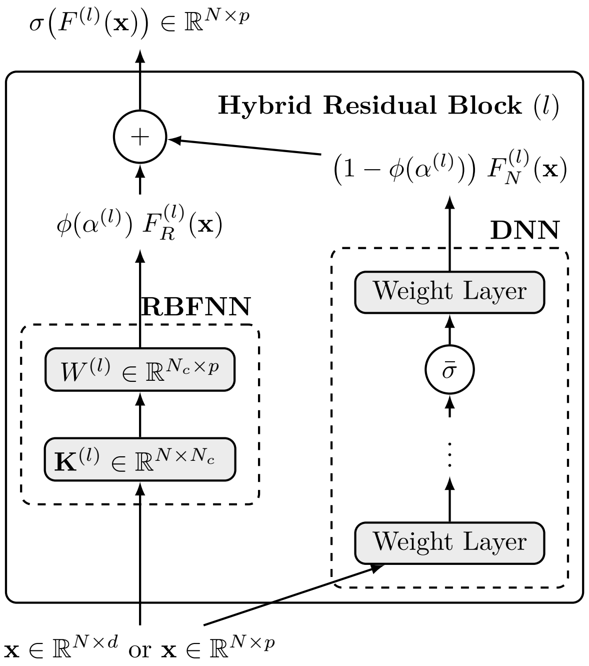

$\sigma\big(F^{(l)}(\mathbf{x})\big) \in \mathbb{R}^{N\times p}$
Hybrid Residual Block (l)
$+$
$\big(1-\phi(\alpha^{(l)})\big)\, F_N^{(l)}(\mathbf{x})$
$\phi(\alpha^{(l)})\, F_R^{(l)}(\mathbf{x})$
DNN
Weight Layer
RBFNN
$W^{(l)} \in \mathbb{R}^{N_c\times p}$
$\bar{\sigma}$
$\mathbf{K}^{(l)} \in \mathbb{R}^{N\times N_c}$
Weight Layer
$\mathbf{x} \in \mathbb{R}^{N\times d}$ or $\mathbf{x} \in \mathbb{R}^{N\times p}$

Figure 1: Illustration of the hybrid residual block (l), which combines contributions from RBFNN and DNN components. The input, $\mathbf{x} \in \mathbb{R}^{N\times d}$ for the first block ($\ell = 1$), or $\mathbf{x} \in \mathbb{R}^{N\times p}$ for inner blocks ($\ell > 1$), is first processed by the RBFNN through the kernel $\mathbf{K}^{(l)} \in \mathbb{R}^{N\times N_c}$, followed by a linear weight layer $W^{(l)} \in \mathbb{R}^{N_c\times p}$ to produce $F_R^{(l)}(x)$. In parallel, the DNN branch applies a sequence of weight layers and nonlinearities $\bar{\sigma}$ to yield $F_N^{(l)}(x)$. The two branches are adaptively fused through a trainable gating parameter $\alpha^{(l)}$, producing the weighted sum $\sigma\big(F^{(l)}(x)\big) = \phi(\alpha^{(l)})F_R^{(l)}(x) + \big(1-\phi(\alpha^{(l)})\big)F_N^{(l)}(x)$, which is then passed forward through the residual connection. This design allows the model to balance kernel-based representations from the RBFNN with expressive features from the DNN, enhancing flexibility.

4.1 Hybrid Residual Block: Dual Gating for Adaptive Representation

The core component of HyResPINNs is the hybrid residual block (see Figure 1), which combines two representational paradigms: an RBFNN component (F_R) and a DNN component (F_N). These are combined using two trainable gating parameters—one internal to each block and one external across blocks—to flexibly balance local and global contributions and to enable dynamic depth adaptation. We note that the degree of locality or globality realized in practice depends on the kernel choice and initialization strategies in the RBFNN component, and the chosen activation function in the DNN component.

4.1.1 Internal Gating: Balancing Local and Global Features

Within each block, the outputs of the RBFNN and DNN components are merged via a trainable scalar parameter $\alpha^{(l)}$, transformed by a tempered sigmoid (Papernot et al., 2020) $\phi(z) = \phi_\tau(z/\tau)$, where $\tau > 0$ is a hyperparameter controlling gate sharpness and enforces the weights in $[0, 1]$. This ensures a convex combination:

$$F^{(l)}(\mathbf{x}) = \sigma\Bigg(\ \phi(\alpha^{(l)})F_R^{(l)}(\mathbf{x}) + \Big(1-\phi(\alpha^{(l)})\Big)\, F_N^{(l)}(\mathbf{x})\ \Bigg), \tag{11}$$

where $F_R^{(l)}(\mathbf{x})$ and $F_N^{(l)}(\mathbf{x})$ are both projected to a common output dimension p, and $\sigma = \tanh$ ensures bounded, nonlinear transformations. Low values of τ yield smooth mixing; high values drives near binary selection between the component outputs. This mechanism enables the network to dynamically interpolate between localized adaptation provided by RBFs—especially with compact support or rapid decay—and global, compositional trends from DNNs.

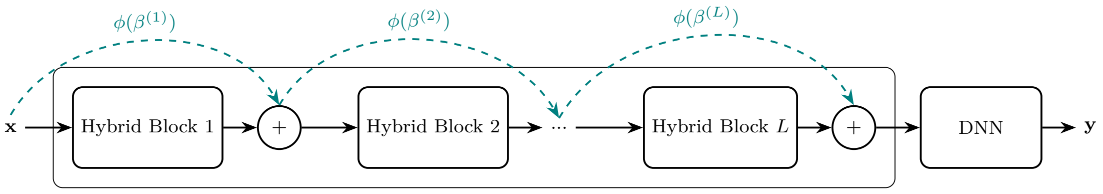


Figure 2: Illustration of the HyResPINN architecture using L hybrid residual blocks. Each block combines outputs from an RBFNN component and a DNN component (see Figure 1). The trainable skip connections $\phi(\beta^{(\ell)})$, modulate the residual flow of information across blocks. These gated skip parameters enable the network to adaptively balance local kernel-based features and global neural representations at different depths, improving flexibility during training. The final output is obtained by passing the aggregated representation through a terminal DNN layer to produce the final prediction $\mathbf{y}$.

The p Parameter and Kernel Method Connection. The block output dimension p defines the feature lifting space where both components operate. This aligns with kernel methods, where data is explicitly mapped into a higher-dimensional feature space to enhance separability or approximation capabilities. As discussed in Section 2, theoretical and practical connections between neural networks and kernel methods inspire such architectures, aiming to efficiently expand representational capacity. In our design, the RBFNN and DNN components leverage this principle, enabling the subsequent layers to operate on more expressive features.

4.1.2 External Gating: Adaptive Residual Depth

A second gating parameter, $\beta^{(l)}$, is likewise passed through a temperature-controlled sigmoid to modulate the residual connection:

$$H^{(l)} = \phi(\beta^{(l)})F^{(l)} + (1 - \phi(\beta^{(l)}))H^{(l-1)}, \tag{12}$$

where $H^{(l-1)}$ is the previous block's output (or the original input for $l = 1$). This adaptive gating enables the network to learn how much each block should transform the representation, thus supporting dynamic depth during training.

4.1.3 Regularization and Initialization

We regularize gating parameters to avoid over-reliance on a single pathway similar to (Howard et al., 2025):

$$\mathcal{L} = \lambda_{ic}\mathcal{L}_{ic} + \lambda_{bc}\mathcal{L}_{bc} + \lambda_r\mathcal{L}_r + \lambda_g \sum_{i=1}^{L} \left(\alpha_i^2 + \beta_i^2\right), \tag{13}$$

such that λ_g is a weight modulating the strength of the regularization.

We initialize $\phi(\beta^{(l)}) = 0$ for all l, biasing the network toward the residual (identity) mapping at the start (Hauser, 2019), and $\phi(\alpha^{(l)}) = 0.5$ to initially balance RBFNN and DNN contributions. DNN weights and RBF coefficients use Xavier (Glorot & Bengio, 2010) initialization, while the RBF shape parameters are initialized using a quantile distance heuristic between the center points in each block. The input center points are initialized using the Poisson sampling method of (Shankar, 2017), and internal block center points are initialized using k-means clustering of the subsequent block's output following (Wurzberger & Schwenker, 2024).

4.2 RBFNN Block Component

For input $\mathbf{x} \in \mathbb{R}^{N\times d}$, the l-th RBFNN constructs:

$$\mathbf{K}_{ij}^{(l)} = \psi^{(l)}\left(r\left(\mathbf{x}_i, \mathbf{x}_j^c; \omega_j^{(l)}\right)\right). \tag{14}$$

Here, $\psi^{(l)}$ denotes the selected RBF (see Table 1), and $r(\cdot, \cdot;\ \omega_j^{(l)})$ is a distance metric parameterized by trainable shape parameters $\omega_j^{(l)}$, which control the width or anisotropy of each basis. The kernel matrix is multiplied by a trainable weight matrix $W^{(l)} \in \mathbb{R}^{N_c \times p}$, yielding the final RBFNN output:

$$F_R^{(l)}(\mathbf{x}) = \mathbf{K}^{(l)}(\mathbf{x}) \cdot W^{(l)} \in \mathbb{R}^{N \times p}. \tag{15}$$

Each hybrid block maintains its own set of RBF center points $\{\mathbf{x}_j^c\}^{(l)}$, allowing block-specific local refinements that adapt independently across network depth.

Wendland C^2 Kernel Selection. In this work, we primarily use the $C^2(\mathbb{R}^3)$ Wendland kernel defined as:

$$\psi_{\text{Wend},C^2}(r) = (1 - r)_+^6\,(35r^2 + 18r + 3), \tag{16}$$

where $(\cdot)_+ = \max\{0, \cdot\}$ ensures compact support and $\mathbf{x}_i^c$ is the center of the i-th RBF. The variable r denotes the distance metric, which may be either isotropic (Equation 8) or anisotropic (Equation 9 or Equation 10). We implement the Wendland kernels with unit support ($\rho = 1$ from Table 1), such that the kernel evaluates to zero whenever $r \geq 1$, ensuring strict locality and sparse kernel matrices. Fixing $\rho = 1$ lets the anisotropy control the effective radii directly, simplifying the parameterization without sacrificing expressivity. For the fully anisotropic case in Equation 10, we parameterize each matrix A_j in Cholesky form, $A_j = M_j^\top M_j$, where M_j is a trainable lower-triangular matrix with positive diagonal entries. This guarantees that A_j remains symmetric positive-definite during training, which ensures well-defined distances, avoids degeneracies, and is practically efficient (Wurzberger & Schwenker, 2024). Empirically, ablation studies confirmed that the C^2 Wendland RBF achieves the lowest approximation error compared to alternative kernels, justifying its use in our architecture.

4.3 DNN Block Component

Each DNN block component computes:

$$F_N^{(l)}(\mathbf{x}) = \mathcal{NN}(\mathbf{x}; \theta^{(l)}) \in \mathbb{R}^{N \times p}, \tag{17}$$

where $\theta^{(l)}$ represents the set of trainable parameters in the network. The final output is projected to the p-dimensional feature space shared by the RBFNN component. By design, this block can use any standard or advanced DNN architecture; in some experiments, we employ Modulated Multi-Layer Perceptrons (ModMLPs) with random weight factorization (Wang et al., 2021a), which are an extension to standard fully connected networks inspired by neural attention mechanisms. These modifications incur relatively small computational and memory overhead while leading to significant improvements in predictive accuracy (Wang et al., 2023a).

4.4 Full Model Composition

The complete HyResPINN model (see Figure 2) consists of L stacked hybrid residual blocks. After the final block, a standard DNN head further refines the representation into the final prediction:

$$u_\theta = \text{DNN}_{out}(H^{(L)}). \tag{18}$$

In some examples, we initialize the final layer of the final DNN using the physics-informed approach of (Wang et al., 2024), a least-squares fit to the PDE residuals at collocation points before gradient training which embeds physics priors and accelerating convergence.

Summary. By fusing RBFNNs with DNNs in a residual, gated architecture, HyResPINNs achieve function space richness, fine-grained adaptivity, and improved parameter efficiency. Our experiments demonstrate that this approach generalizes robustly across canonical and irregular PDE benchmarks, outperforming state-of-the-art neural PDE solvers in both accuracy and parameter efficiency.

Problem	PINN	ResPINN	ExpertPINN	StackedPINN	PirateNet	**HyResPINN**
Allen–Cahn (Dirchlet BCs)	2.65e-1	6.82e-2	9.66e-3	6.11e-2	1.10e-2	**2.39e-3**
Allen–Cahn (Periodic BCs)	5.26e-1	2.70e-3	3.86e-3	5.87e-3	2.62e-5	**1.19e-5**
DarcyFlow (2D Dirichlet BCs)	7.50e-4	4.60e-4	5.00e-4	9.00e-4	8.71e-5	**5.44e-5**
(2D Neumann BCs)	2.00e-3	1.40e-3	1.20e-4	4.10e-3	1.70e-4	**6.00e-5**
(3D Dirichlet BCs)	2.20e-3	1.20e-2	2.10e-2	5.40e-2	3.90e-2	**1.20e-3**
(3D Neumann BCs)	8.50e-3	6.10e-3	1.10e-3	3.90e-2	1.30e-3	**1.10e-3**
DarcyFlow (rough)	6.85e-5	2.69e-5	1.10e-4	1.50e-4	5.44e-5	**1.05e-5**
Kuramoto–Sivashinsky	1.42e-1	1.29e-2	1.57e-3	1.86e-1	2.00e-3	**8.12e-4**
Korteweg–De Vries	8.17e-1	5.72e-1	6.86e-4	2.98e-1	5.61e-4	**4.09e-4**
Grey–Scott (u)	7.54e-1	9.87e-3	4.79e-3	1.21e-3	**3.61e-3**	6.43e-3
Grey–Scott (v)	9.56e-1	1.10e-2	8.98e-3	6.90e-2	**9.39e-3**	1.31e-2

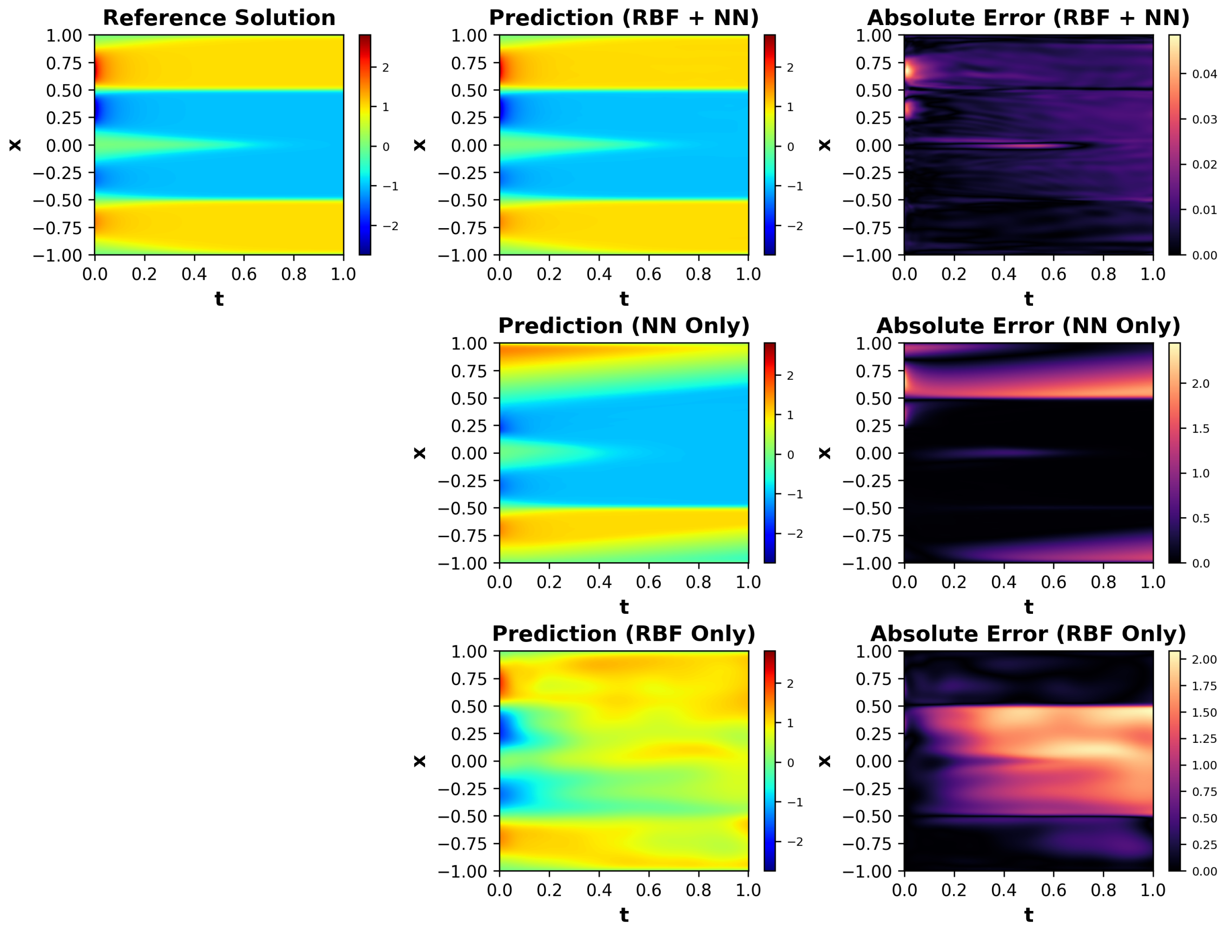
Table 2: Relative L^2 test error results across a range of PDE benchmarks under different boundary conditions. For the Grey–Scott PDE, errors are reported separately for both components (u and v). The lowest error for each problem is highlighted in bold.

5 Experimental Results and Discussion

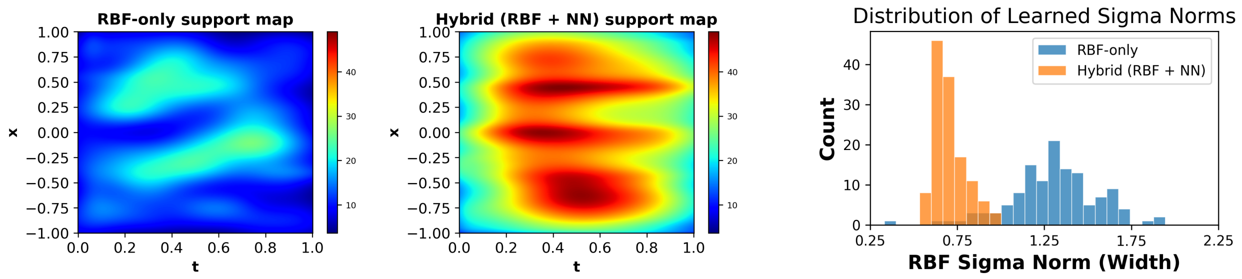
We evaluate HyResPINNs on a suite of PDE benchmark problems designed to test its function space richness, adaptivity, and efficiency per DoF. These benchmarks have been selected to either model solution features that present significant challenges for computational models or are recognized as established benchmark problems. First, to evaluate localized adaptivity, we consider two 1D non-linear hyperbolic Allen-Cahn equations, one with Dirichlet boundary conditions in Section 5.1, and periodic boundary conditions in Section 5.2, each containing sharp solution features. In Section 5.3, we test 2D and 3D Darcy flow problems on annulus and extruded annulus domains to evaluate generalization to non-standard geometries using various boundary conditions. Then, Section 5.4 details a 2D Darcy flow problem with rough, discontinuous coefficients to examine robustness under discontinuities. To further explore the function space richness of our methods, we evaluate the chaotic and multi-scale systems given by the Kuramoto–Sivashinsky equation in Section 5.5, the Korteweg-de Vries equation in Sections 5.6, and the Grey–Scott equation in Section 5.7. We assess predictive accuracy and efficiency per DoF throughout all experiments, comparing against state-of-the-art baselines. Table 2 summarizes the top results from each test.

Baseline Methods We compare the proposed HyResPINNs against a suite of baseline methods, including 1) the standard PINN as originally formulated in (Raissi et al., 2019b) (PINN), 2) a PINN based on the architectural guidelines proposed in (Wang et al., 2023a) (ExpertPINNs), 3) PINNs with residual connections (ResPINNs), 4) PirateNets (Wang et al., 2024), and 5) the stacked PINN approach of (Howard et al., 2025) (StackedPINNs). To ensure a fair comparison, we implemented each approach following the architectural details provided in each.

Experimental Setup We follow similar experimental design procedures as those described in (Wang et al., 2022a; 2023a; 2024). We employ mini-batch gradient descent for most experiments, with collocation points randomly sampled during each training iteration. In the Darcy Flow experiments, we use full-batch gradient descent, with collocation point sets generated using the efficient Poisson sampling technique from (Shankar, 2017). For training, we use the Adam optimizer Kingma & Ba (2015), and follow the learning rate schedule of (Wang et al., 2023a) which starts with a linear warm-up phase of $5,000$ iterations, starting from zero and gradually increasing to 10^{-3}, followed by an exponential decay at a rate of 0.9. We also employ a learning rate annealing algorithm (Wang et al., 2023a) to balance each loss term and causal training (Wang et al., 2022a; 2023a) to mitigate causality violation in time-dependent PDEs. Further, we apply exact periodic or Dirichlet boundary conditions (Dong & Ni, 2021) when applicable. We use the hyperbolic tangent activation functions in all DNNs and initialize each network's parameters using the Glorot normal scheme (Glorot & Bengio, 2010). We ran five random trials for each test and report the mean values achieved in each plot and table. We measure the quality of each model's predicted solution u_θ using the relative L^2 error, defined



(a) Comparison of the three variants: HyResPINN (RBFNN+DNN), DNN-only, and RBF-only's predicted solutions (middle) and the absolute errors (right) compared to the true solution (lef).



(b) (Left) RBF Support Map. Hybrid model exhibits stronger kernel coverage at sharp solution features. (Right) Sigma Norms. Distribution of learned RBF widths. Hybrid learns broader, smoother kernels.

Figure 3: *(Allen–Cahn Equation)* Comparison of model outputs and internal representations across variants.

as $\frac{\|u_\theta - u^*\|_2}{\|u^*\|_2}$, where u^* denotes the reference solution (either exact or computed using a high-fidelity solver). Here $\|\cdot\|_2$ refers to the discrete ℓ_2 norm evaluated over a set of equispaced test points in the domain, which serves as an approximation of the continuous $L^2(\Omega)$ norm. The code for our methods and all compared baseline approaches is written in Jax 1.11 and will be publicly available upon publication. We train all models on an Nvidia H100 GPU running CentOS 7.2.

5.1 1D Allen-Cahn with Dirichlet Boundary Conditions

We start by considering the one-dimensional Allen–Cahn equation defined on $x \in [-1, 1]$, $t \in [0, 1]$ given as:

$$\frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial x^2} - 5u^3 + 5u,$$

where $u(x, t)$ is the solution and $\varepsilon = 0.001$ is the diffusion coefficient. We impose the homogeneous Dirichlet boundary conditions:

$$u(-1, t) = 0, \quad u(1, t) = 0, \quad \text{for all } t \in [0, 1],$$

and the initial condition is:

$$u(x, 0) = -2\sin(2\pi x)\left(2e^{-8(x-0.5)^2} - e^{-5(x+0.5)^2}\right).$$

This nonlinear, reaction-diffusion PDE's solution exhibits both smooth, global variation and sharp, localized peaks (centered at $x = \pm 0.5$). This problem is designed to evaluate a model's capacity to accurately capture these localized features while simultaneously preserving the global solution structure.

We first perform ablation studies to systematically investigate the individual contributions of the RBFNN and DNN components within the HyResPINN residual blocks. Next, to measure the DoF efficiency, we benchmark the complete HyResPINN model against baseline solvers with comparable model complexity, quantified by parameter count and network depth. Appendix A details ablation studies measuring the accuracy and computational cost of the HyResPINN architecture under varying RBF kernel types, numbers of center points, and deepening residual blocks. For the remainder of the experiments, we fix the RBF kernel to Wendland C^2—the configuration our ablation studies identify as the most effective concerning accuracy, computational efficiency, while aligning with our goal of inherent locality.

Expressivity: Local and Global Representations. We first examine the function space richness of HyResPINNs by isolating local (RBFNN), global (DNN), and hybrid (RBFNN+DNN) representations, testing whether their combination enables more expressive solution approximation than either alone. To isolate the contributions of each hybrid component, we conduct ablation experiments comparing RBFNN-only, DNN-only, and full HyResPINN configurations. In the RBFNN-only configuration, the DNN outputs are fixed to zero, isolating localized basis function representation. In the DNN-only configuration, the RBFNN outputs are fixed to zero, isolating global neural representation. In the full HyResPINN, both components are active and combined via learned gating. All models are trained with Adam for 100,000 iterations, with a batch size of 1024 collocation points randomly sample each iteration. The DNNs use three layers of 64 neurons, RBFNNs use 64 centers, and HyResPINNs use both and one block.

As shown in Figure 3a, the full hybrid model achieves the lowest relative error (2.4×10^{-3}), accurately resolving both the steep gradients and the global structure. The DNN-only baseline captures the overall shape but fails to resolve sharp peaks, resulting in localized errors above 1.0 (relative error 2.65×10^{-1}). The RBFNN-only baseline captures localized peaks but struggles to maintain smoothness elsewhere (4.6×10^{-3}). Neither baseline achieves the accuracy or robustness of the hybrid. These results directly confirm that hybrid local–global representation enables superior function approximation on challenging, multiscale PDEs.

Adaptivity: Solution Refinement. We next examine how HyResPINNs achieve fine-grained adaptivity in the width of RBF kernels. Kernel support maps (Fig. 3b, left), computed by summing RBF activations across the domain, reveal that the hybrid model develops broad, overlapping support, densely covering regions with sharp features. In contrast, the RBFNN-only model forms narrowly focused kernels with little overlap. Furthermore, the distribution of learned RBF widths (Fig. 3b, right) demonstrates that HyResPINNs favor wider, smoother kernels compared to RBFNNs alone. This broader support reflects the model's ability to blend basis functions, promoting both smooth transitions and precise local refinement—key ingredients of adaptive expressivity.

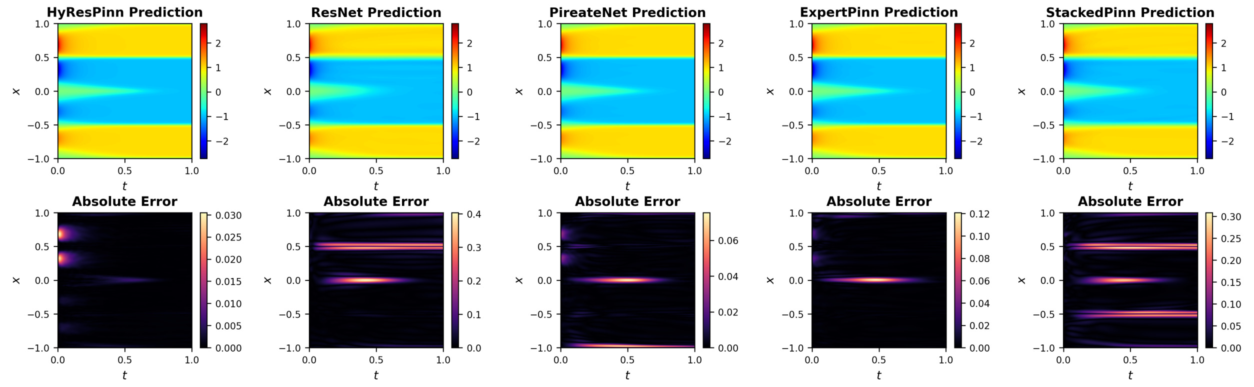


Figure 4: *Allen–Cahn equation:* Predicted solutions (top row) and corresponding absolute errors (bottom row) for various baseline models. Each column shows the model's prediction across the full domain, followed by the absolute errors relative to the reference solution.

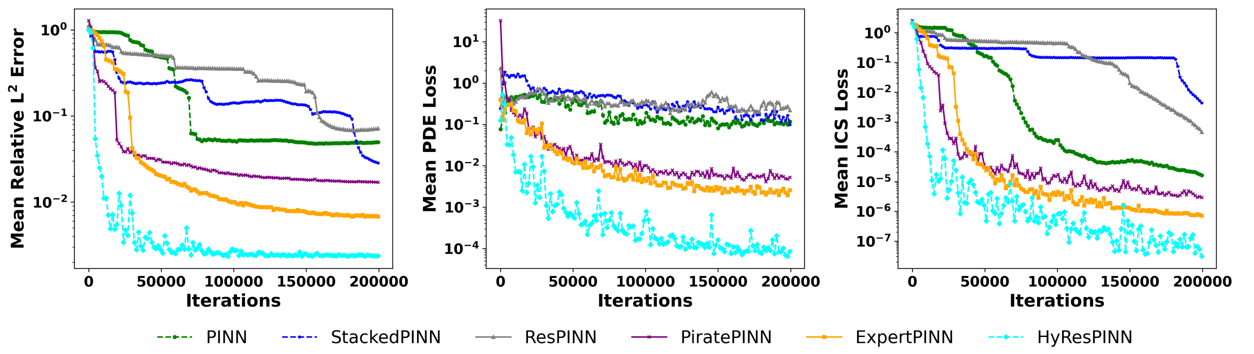


Figure 5: *Allen–Cahn equation:* (Left) Relative L^2 errors, (middle) PDE loss, (right) initial condition loss, across all baselines of similar parameter numbers as a function of training iteration.

Model	# of Parameters	Computational Cost
PINN	17025	0.0121 (0.0003)
StackedPINN	17157	0.0175 (0.0001)
ResPINN	17157	0.0148 (0.0003)
PiratePINN	17926	0.0216 (0.0001)
ExpertPINN	17793	0.0185 (0.0002)
HyResPINN	17667	0.0180 (0.0001)

Table 3: Comparison of model complexity and efficiency across PINN variants. The table reports the total number of trainable parameters and the average wall-clock time (in seconds) required to perform 100 iterations, with standard deviations in parentheses.

Efficiency: Accuracy and Convergence per Degree of Freedom. Finally, we assess efficiency per DoF by comparing HyResPINNs to state-of-the-art neural PDE solvers under matched parameter budgets, focusing on both final accuracy and convergence speed. As summarized in Figure 5 and Table 3, HyResPINNs consistently outperform all baselines, achieving lower mean relative L^2 error, faster convergence, and sharper resolution of transition regions. The final relative errors are 6.82×10^{-2} for ResNet, 9.66×10^{-3} for ExpertPINN, 1.09×10^{-2} for PirateNet, and 6.12×10^{-2} for StackedPINN, compared to just 2.39×10^{-3} for

HyResPINNs. Qualitative comparisons (Fig. 4) further show that only HyResPINNs robustly resolve both sharp and smooth solution features.

Summary. These analyses highlight the benefit of hybridization: the RBFNN component provides localized adaptivity, while the DNN component ensures global coherence. HyResPINN's lower error and more adaptive internal structure demonstrate that coordinated local–global representation is essential for robust and expressive PDE learning.

5.2 1D Allen-Cahn with Periodic Boundary Conditions

Next, we present results for 1D the Allen-Cahn problem with periodic boundary conditions, a well-known challenge for conventional PINN models and a widely studied benchmark in recent literature (Wight & Zhao, 2020; Wang et al., 2022a; Daw et al., 2022). For simplicity, we consider the one-dimensional case with a periodic boundary condition with $t \in [0, 1]$, and $x \in [-1, 1]$:

$$\begin{aligned} &u_t - 0.0001u_{xx} + 5u^3 - 5u = 0\,, \\ &u(0, x) = x^2 \cos(\pi x)\,, \\ &u(t, -1) = u(t, 1)\,,\; u_x(t, -1) = u_x(t, 1)\,. \end{aligned}$$

This problem is particularly difficult for two reasons. First, the Allen-Cahn PDE requires PINNs to approximate solutions with sharp transitions in space and time, posing challenges for standard neural network architectures. Second, there is an inherent incompatibility between the initial and boundary conditions, leading to a weak discontinuity at time $t = 0$. Traditional numerical methods would likely produce oscillations near the discontinuity due to the mismatch between the initial and boundary conditions. However, unlike traditional solvers, PINNs do not rely on discretization and instead learn their model parameters through optimization. This optimization process may mitigate spurious oscillations, as it relies on stochastic updates and continuous function approximations. Specifically, implicit regularization from both mini-batch training and stochastic gradient descent (SGD) may contribute to stabilizing optimization and reducing any spurious oscillations (Sekhari et al., 2021). Full experimental details are provided in Appendix B.

Expressivity: Capturing Periodic and Discontinuous Solutions. Similar to the previous section, we compare HyResPINNs to DNN-only and RBFNN-only variants. Figure 6 demonstrates the advantages of the hybrid residual block structure in capturing both smooth and sharp features in the 1D Allen-Cahn solution. The absolute error plots show that HyResPINN exhibits significantly lower errors than standard DNN-only and RBFNN-only, validating its improved accuracy.

Adaptivity: Depth Ablation and Solution Refinement. To test architectural adaptivity, we vary the number of residual blocks. Figure 7 demonstrates that increasing the number of residual blocks in HyResPINNs leads to consistently lower errors, outperforming the competing methods. Residual learning allows deeper networks to refine predictions iteratively, progressively improving both global structure and sharp transitions. The competing methods struggle to maintain accuracy as depth increases, whereas HyResPINNs remains robust, demonstrating the effectiveness of deep hybrid residual learning. Specifically, the standard PINNs prediction error increases as the network becomes deeper, eventually failing to capture any part of the solution. ResNets errors also increase with increased depth, though at a slower rate than the standard PINN. This is likely due to the static skip connection in the standard residual architecture design preventing the adaptive learning of residual connections.

Efficiency: Accuracy and Convergence per Degree of Freedom. We next assess the efficiency per DoF by comparing HyResPINNs to state-of-the-art neural PDE solvers under matched parameter budgets, focusing on both final accuracy and convergence speed (see Figure 8). Training times reported in Table 4 show that HyResPINN has comparable training costs to other methods, while exhibiting consistently lower error rates. Together, these results show that HyResPINN's hybrid, gated design delivers the function space richness, adaptivity, and efficiency required for challenging periodic PDEs, outperforming conventional architectures across both local and global metrics.

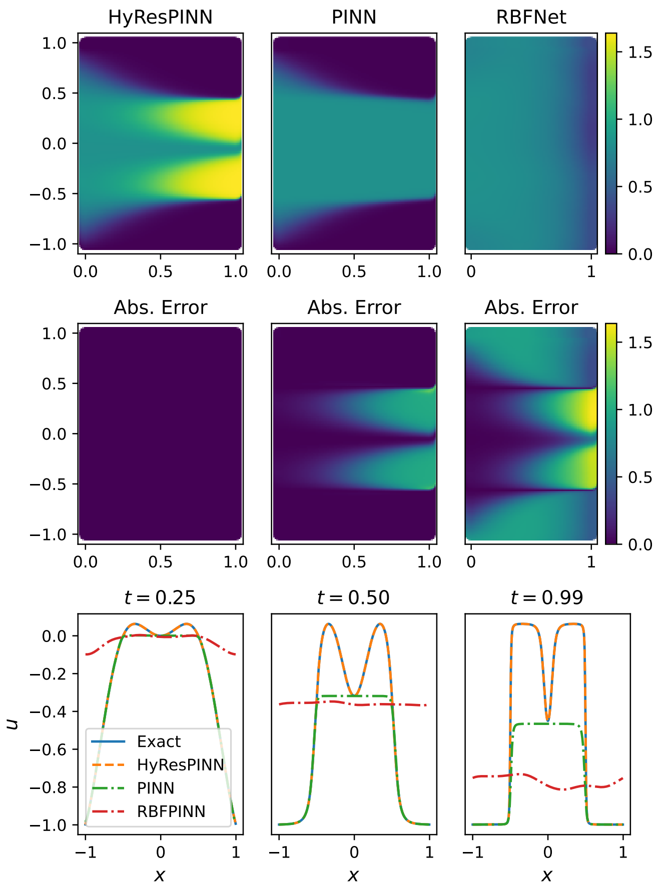


Figure 6: *Allen-Cahn equation:* Comparison of the predicted solutions for the Allen-Cahn equation using HyResPINN, standard PINN, and RBF PINN models. The top row shows the predicted solutions for HyResPINN (left), standard PINN (center), and RBF PINN (right). The second row shows the absolute error between the predicted and true solutions. The bottom row shows the predicted solutions for time steps ($t = 0.25, 0.5, 0.99$) compared to the true solution.

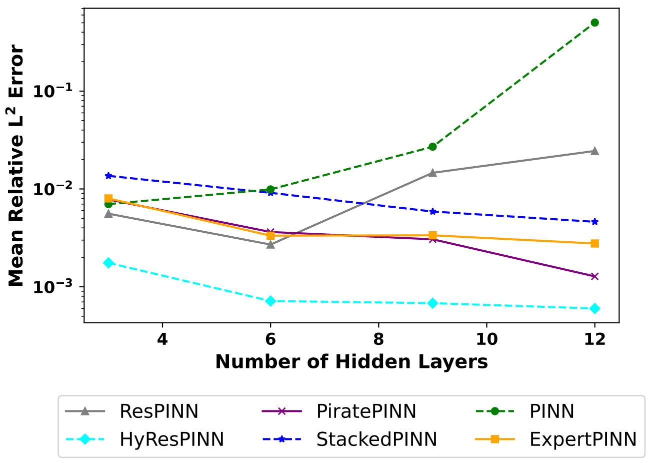


Figure 7: *Allen-Cahn equation:* Comparison of the mean relative L^2 error using various methods as a function of the number of hidden layers.

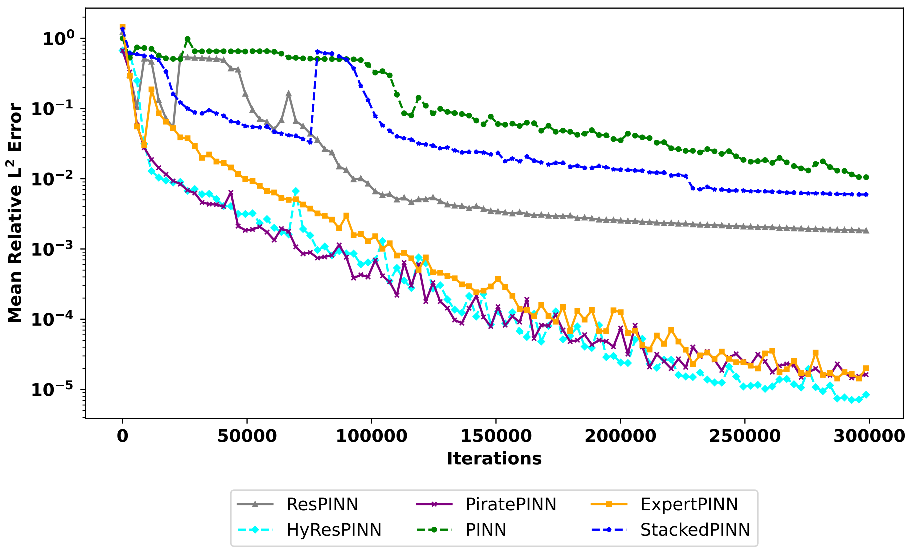


Figure 8: *Allen-Cahn equation:* Comparison of mean relative L^2 error across various methods, plotted against the number of training iterations.

Model	# of Parameters	Computational Cost
PINN	727,170	0.0365 (0.0003)
StackedPINN	795,283	0.0525 (0.0002)
ResPINN	793,218	0.0444 (0.0003)
PiratePINN	727,173	0.0648 (0.0002)
ExpertPINN	727,170	0.0558 (0.0002)
HyResPINN	739,976	0.0769 (0.0001)

Table 4: Comparison of model complexity and efficiency across PINN variants. The table reports the total number of trainable parameters and the average wall-clock time (in seconds) required to perform 100 iterations, with standard deviations in parentheses.

5.3 Darcy Flow with Smooth Coefficients

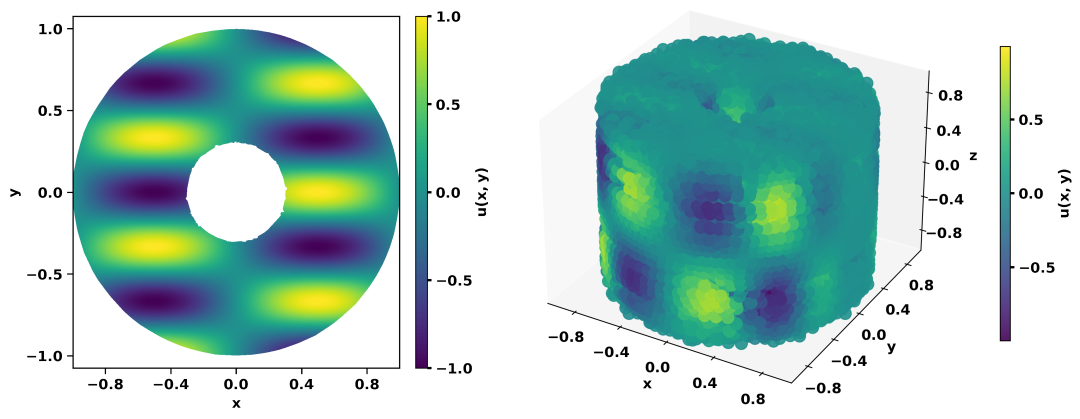


Figure 9: *Smooth Darcy Flow equation:* True solutions of the 2D and 3D problems.

We next test the performance of HyResPINN on the Darcy Flow equation with smooth coefficients. Darcy Flow serves as a fundamental model for various physical processes, including porous media flow, heat transfer, and semiconductor doping. In most applications, the flux $\mathbf{u} = -\mu\nabla\phi$ is the variable of primary interest. The Darcy Flow problem is an elliptic boundary value problem given by:

$$-\nabla \cdot \mu\nabla\phi = f \quad \text{in } \Omega \tag{19}$$

$$\phi = u \quad \text{on } \Gamma_D$$
$$\mathbf{n} \cdot \mu\nabla\phi = g \quad \text{on } \Gamma_N$$

where Γ_D and Γ_N denote Dirichlet and Neumann parts of the boundary Γ, respectively, μ is a symmetric positive definite tensor describing a material property and f, u and g are given data.

We use manufactured solutions in both 2D and 3D domains, ensuring that the exact solution contains global structure and cross-dimensional correlations. Specifically, the exact solutions are given as:

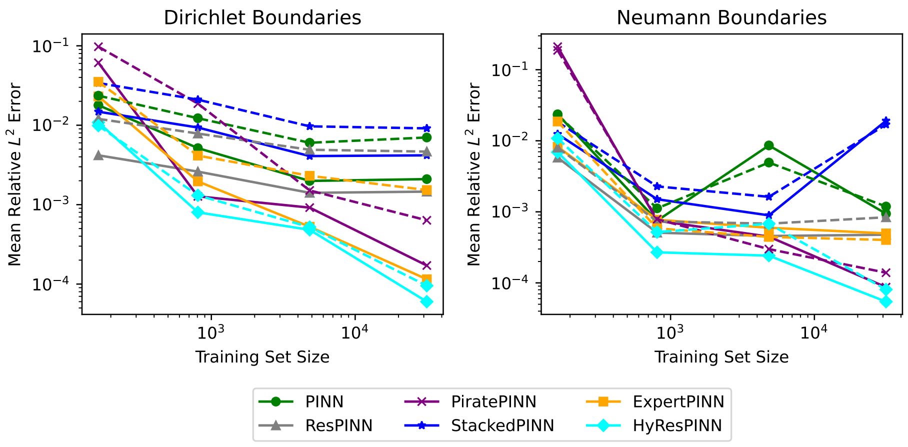
$$u(x, y) = \sin(\pi x)\cos(3\pi y), \tag{20}$$
$$u(x, y, z) = \sin(\pi x)\sin(3\pi y)\sin(\pi z), \tag{21}$$

for the 2D and 3D cases, respectively. Figure 9 displays the target solutions for both cases.

To rigorously evaluate data efficiency, adaptivity, and convergence, we generated static training datasets for all experiments, fixing the set of collocation points within each run. This design enables controlled studies of error reduction as the number of training points increases, while eliminating the variability introduced by stochastic sampling. We systematically varied the number of collocation points to perform convergence studies, and solved each problem under both Dirichlet and Neumann boundary conditions in both 2D and 3D domains. This approach allows for a fair, direct comparison of accuracy, data efficiency, and robustness across problem dimensions and boundary types. All baseline models were matched in parameter count and trained under identical conditions for consistent benchmarking.

Expressivity: Accurate Solution of Smooth Multidimensional PDEs. HyResPINN accurately reconstructs these solutions and achieves low mean relative L^2 errors across all tested training set sizes (see Figure 10, solid lines), indicating its capacity to represent smooth solutions in higher dimensions. Notably, HyResPINN's predictions for both the solution and the physically relevant flux quantity remain stable and accurate even as dimensionality increases.

Adaptivity: Robustness to Boundary Conditions and Domain Complexity. HyResPINN maintains high accuracy across both types of boundary conditions in 2D and 3D, whereas baseline methods often exhibit



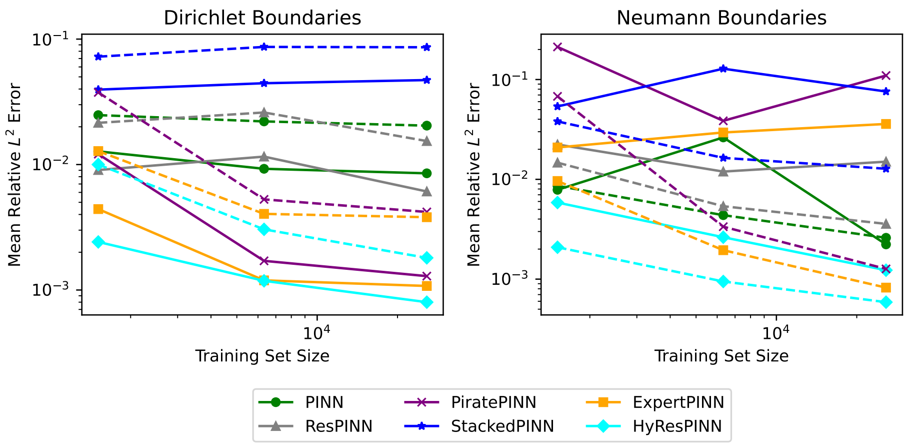


Figure 10: *2D and 3D Smooth Darcy Flow equations:* Comparison of the mean relative L^2 errors across each baseline method for the 2D Darcy Flow problem (top) and the 3D Darcy flow problem (bottom) plotted against the number of training collocation points using both Dirichlet and Neumann boundary conditions. Solid lines show the solution error, while dashed lines show the x-directional flux errors.

significant degradation or instability, particularly for Neumann constraints (see Figure 10, where dashed lines denote flux errors). The model also seamlessly scales from the 2D annulus to the more complex 3D cylindrical domain, demonstrating flexibility in handling increased geometric and dimensional complexity without additional tuning.

Efficiency per Degree of Freedom: Data Efficiency and Convergence with Static Datasets. HyResPINN consistently achieves lower solution and flux errors than competing models, especially in the low-data regime. This data efficiency confirms that HyResPINN can generalize robustly from limited training points, reducing computational cost while maintaining accuracy.

Summary. In summary, HyResPINN demonstrates (i) strong function space expressivity for smooth multi-dimensional PDEs, (ii) robust adaptivity to varying boundary conditions and domain complexity, and (iii) superior efficiency per degree of freedom—achieving state-of-the-art accuracy and convergence with less data. These results further validate the hybrid residual framework as an effective and scalable solver for elliptic PDEs in both 2D and 3D.

5.4 Darcy Flow with Rough Coefficients

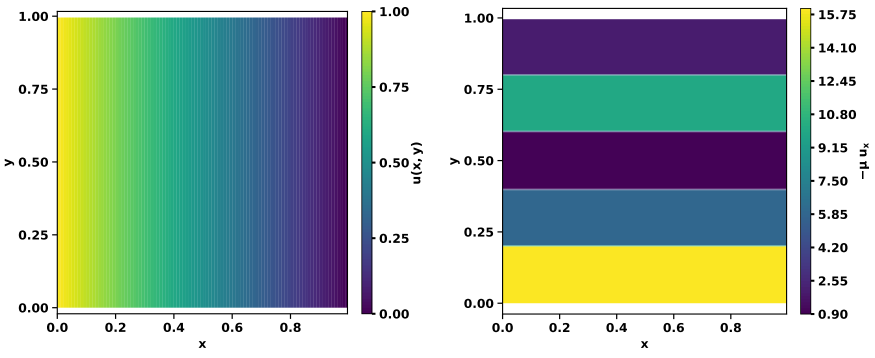


Figure 11: (Left) True solution on the 2D box domain. (Right) True transverse flux.

Accurately solving the Darcy Flow equation with discontinuous coefficients is a fundamental challenge for both numerical and machine learning-based PDE solvers. In heterogeneous materials, the solution and flux field must correctly capture interface behavior: while the normal component of the flux remains continuous, the tangential component may be discontinuous. We focus on the well-studied "five strip problem," well-documented manufactured solution test (Trask et al., 2017; Nakshatrala et al., 2006; Masud & Hughes, 2002), which evaluates the ability of a method to preserve the continuity of normal flux across material interfaces. This problem divides the domain $\Omega = [0, 1]^2$ into five horizontal strips, each with a distinct diffusion coefficient μ_i. Specifically, the prescribed exact solution on domain $\Omega = [0, 1]^2$ under Neumann boundary conditions is given by:

$$\phi_{ex} = 1 - x, \quad \text{and} \quad \Gamma_N = \Gamma, \tag{22}$$

such that Ω is divided in five equal strips,

$$\Omega_i = \{(x, y) \mid 0.2(i-1) \ \leq \ y \ \leq \ 0.2i \ ; \ 0 \leq x \leq 1\}, \quad i = 1, ..., 5 \tag{23}$$

with different μ_i on each Ω_i such that $\mu_1 = 16, \quad \mu_2 = 6, \quad \mu_3 = 1, \quad \mu_4 = 10, \quad \mu_5 = 2$. Figure11 shows the true solution and corresponding transverse flux.

Expressivity: Capturing Discontinuous and Physically Consistent Flux. The rough-coefficient Darcy Flow problem requires a model capable of representing changes in material properties and the resulting discontinuous flux behavior. HyResPINN's compositional block representation enables it to accurately approximate both the solution and its flux, even in the presence of abrupt material transitions.

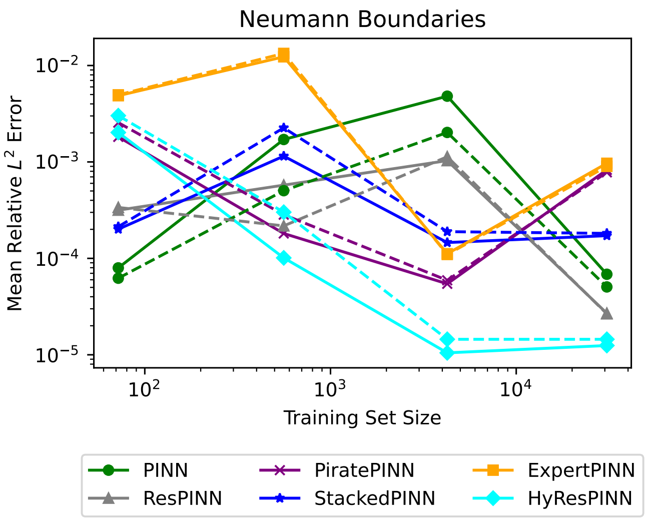


Figure 12: *2D Rough Darcy Flow equation:* Comparison of the mean relative L^2 errors across various methods, plotted against the number of training collocation points. Solid lines show the solution error, while dashed lines show the x-directional flux errors.

Adaptivity: Robustness to Material Interfaces. Robust adaptivity is crucial for eliminating spurious oscillations and enforcing correct flux continuity at interfaces. Unlike classical collocated methods—which frequently exhibit oscillatory flux or fail to preserve normal continuity at discontinuities—HyResPINN adapts its representation to maintain the correct physical behavior of both normal and tangential flux components. The method handles Neumann boundary conditions and variable μ_i, confirming its architectural flexibility for rough, heterogeneous domains.

Efficiency per Degree of Freedom: Convergence with Increasing Data. To quantify data efficiency, we perform convergence studies by increasing the number of collocation points with static datasets and matching parameter counts for all methods. As shown in Figure 12, HyResPINN achieves superior accuracy in both solution and flux as data increases. While it may require larger training sets for full convergence on rough problems, it ultimately surpasses baselines, especially in avoiding non-physical oscillations and preserving flux continuity. This demonstrates high efficiency per DoF, particularly for complex, interface-driven physics.

Summary. In summary, HyResPINN demonstrates (i) strong expressivity for non-smooth PDEs, (ii) robust adaptivity to material interfaces and challenging boundary conditions, and (iii) superior efficiency per DoF for physically accurate solution and flux approximation in rough-coefficient Darcy Flow problems.

5.5 Kuramoto–Sivashinsky Equation

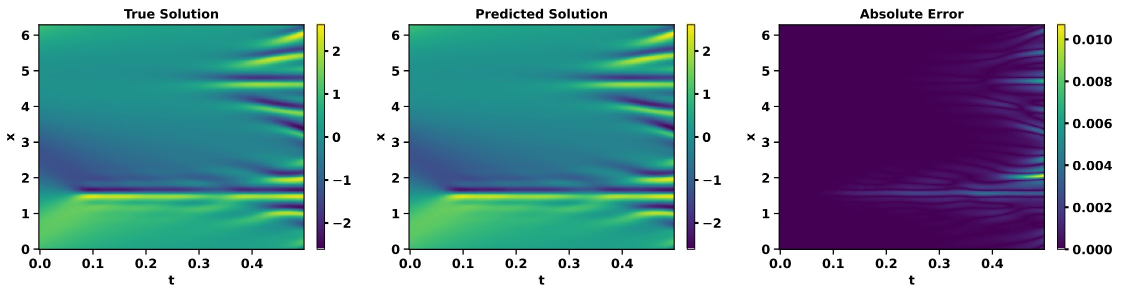


Figure 13: *Kuramoto–Sivashinsky equation:* Comparison of the best prediction against the reference solution.

Model	Final Mean Relative L^2 Errors
PINN	0.142 (0.003)
StackedPINN	0.186 (0.021)
ResPINN	1.29×10^{-2} (0.001)
PiratePINN	2.0×10^{-3} (0.0013)
ExpertPINN	1.57×10^{-3} (0.002)
HyResPINN	$\mathbf{8.12 \times 10^{-4}}$ (0.0075)

Table 5: *Kuramoto–Sivashinsky equation:* Final mean relative L^2 error results for all baseline models. Values in parentheses indicate standard deviations across runs. The boldfaced entry highlights the best-performing method (lowest mean error).

In this section, we use the Kuramoto–Sivashinsky equation as a benchmark problem under periodic boundary conditions. The Kuramoto–Sivashinsky equation is a fourth-order nonlinear partial differential equation widely used to model spatiotemporal patterns in fluid dynamics, combustion, and other systems exhibiting chaotic behavior. We consider the 1D Kuramoto–Sivashinsky equation of the form:

$$\frac{\partial u}{\partial t} + \alpha u \frac{\partial u}{\partial x} + \beta \frac{\partial^2 u}{\partial x^2} + \gamma \frac{\partial^4 u}{\partial x^4} = 0, \tag{24}$$

subject to periodic boundary conditions and an initial condition:

$$u(0, x) = u_0(x) = \cos(x)(1 + \sin(x)). \tag{25}$$

The parameters α, β, and γ control the dynamical behavior of the equation. We use the configurations: $\alpha = 100/16$, $\beta = 100/16^2$, and $\gamma = 100/16^4$ for chaotic behaviors. Solving this equation allows us to assess HyResPINN's ability to approximate highly complex and chaotic systems. Following (Wang et al., 2023a), we employ a time-marching training routine to facilitate stable optimization. HyResPINN and all baseline models are trained using the same time-stepping scheme and matched parameter budgets. Full experimental details are provided in Appendix C.

HyResPINN demonstrates strong expressivity by accurately recovering the intricate, chaotic solution trajectories generated by the Kuramoto–Sivashinsky equation, closely matching the reference simulation as illustrated in Figure 13. Throughout both regular and highly variable temporal regions, the model maintains high accuracy and adaptively resolves evolving spatial features and sharp transitions, even as chaotic patterns emerge and develop over time. In terms of efficiency per DoF, HyResPINN achieves the lowest mean relative L^2 error (8.12×10^{-4}) among all tested methods, as summarized in Table 5, outperforming state-of-the-art PINN architectures without requiring an increase in parameter count.

5.6 Korteweg-De Vries Equation

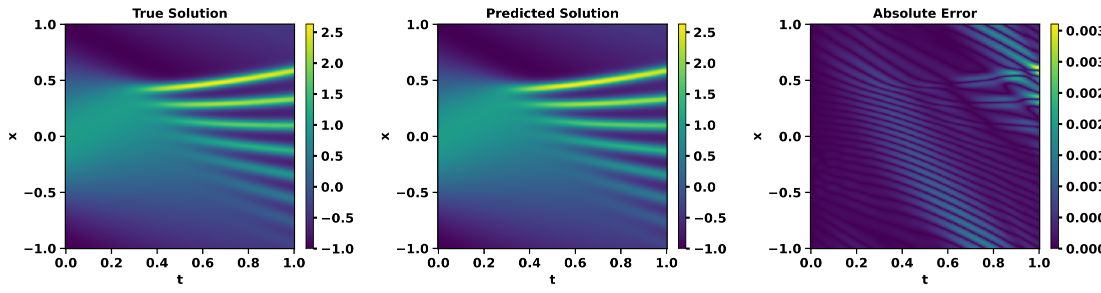


Figure 14: *Korteweg-De Vries equation:* Comparison of the best prediction against the reference solution.

Model	**Final Mean Relative L^2 Errors**
PINN	0.817 (0.003)
StackedPINN	2.98×10^{-2} (0.021)
ResPINN	0.572 (0.001)
PiratePINN	5.61×10^{-4} (0.0013)
ExpertPINN	6.86×10^{-4} (0.0021)
HyResPINN	$\mathbf{4.09 \times 10^{-4}}$ (0.0052)

Table 6: *Korteweg-De Vries equation:* Final mean relative L^2 error results for all baseline models. Values in parentheses indicate standard deviations across runs. The boldfaced entry highlights the best-performing method (lowest mean error).

Next, we explore the one-dimensional Korteweg–De Vries (KdV) equation, a fundamental model used to describe the dynamics of solitary waves, or solitons. The KdV equation is expressed as follows:

$$
\begin{aligned}
&u_t + \eta u u_x + \mu^2 u_{xxx} = 0\,, \quad t \in (0,1), \quad x \in (-1,1)\,,\\
&u(x,0) = \cos(\pi x)\,,\\
&u(t,-1) = u(t,1)\,,
\end{aligned}
$$

where η governs the strength of the nonlinearity and μ controls the dispersion level. Under the KdV dynamics, this initial wave evolves into a series of solitary-type waves. We adopt the classical parameters of the KdV equation, setting $\eta = 1$ and $\mu = 0.022$ (Zabusky & Kruskal, 1965). Full experimental details are provided in Appendix C.

HyResPINN accurately reconstructs the evolution of solitary waves as shown in Figure 14. The model adaptively resolves the highly localized soliton structures and evolving nonlinear features across the domain, demonstrating its ability to allocate expressive capacity where needed as the solution evolves. In terms of efficiency per DoF, HyResPINN achieves the lowest relative L^2 error (4.09×10^{-4}) of all tested models, as reported in Table 6, outperforming baselines with matched parameter budgets and confirming the architecture's capacity for efficient, high-accuracy approximation of dispersive, nonlinear PDEs.

5.7 Grey-Scott equation

Model	**Final Mean Relative L^2 Errors**
PINN	0.754 (0.003) / 0.956 (0.003)
StackedPINN	0.00121 (0.009) / 0.069 (0.021)
ResPINN	0.00987 (0.002) / 0.011 (0.001)
PiratePINN	$\mathbf{3.61 \times 10^{-3}}$ (0.0025)/$\mathbf{9.39 \times 10^{-3}}$ (0.0013)
ExpertPINN	4.79×10^{-3} (0.021) / 8.98×10^{-3} (0.009)
HyResPINN	6.43×10^{-3} (0.0041) /1.31×10^{-2} (0.0052)

Table 7: *Grey Scott equation:* Final mean relative L^2 error results for all baseline models. Values in parentheses indicate standard deviations across runs. The boldfaced entry highlights the best-performing method (lowest mean error).

In this example, we solve the 2D Grey-Scott equation, a reaction-diffusion system that describes the interaction of two chemical species. The form of this PDE is given as follows

$$
\begin{aligned}
u_t &= \epsilon_1 \Delta u + b_1(1-u) - c_1 u v^2, \quad t \in (0,2)\,,\ (x,y) \in (-1,1)^2\,,\\
v_t &= \epsilon_2 \Delta v - b_2 v + c_2 u v^2\,, \quad t \in (0,2)\,,\ (x,y) \in (-1,1)^2\,,
\end{aligned}
$$

subject to the periodic boundary conditions and the initial conditions

$$
\begin{aligned}
u_0(x,y) &= 1 - \exp(-10((x+0.05)^2 + (y+0.02)^2))\,,\\
v_0(x,y) &= 1 - \exp(-10((x-0.05)^2 + (y-0.02)^2))\,.
\end{aligned}
$$

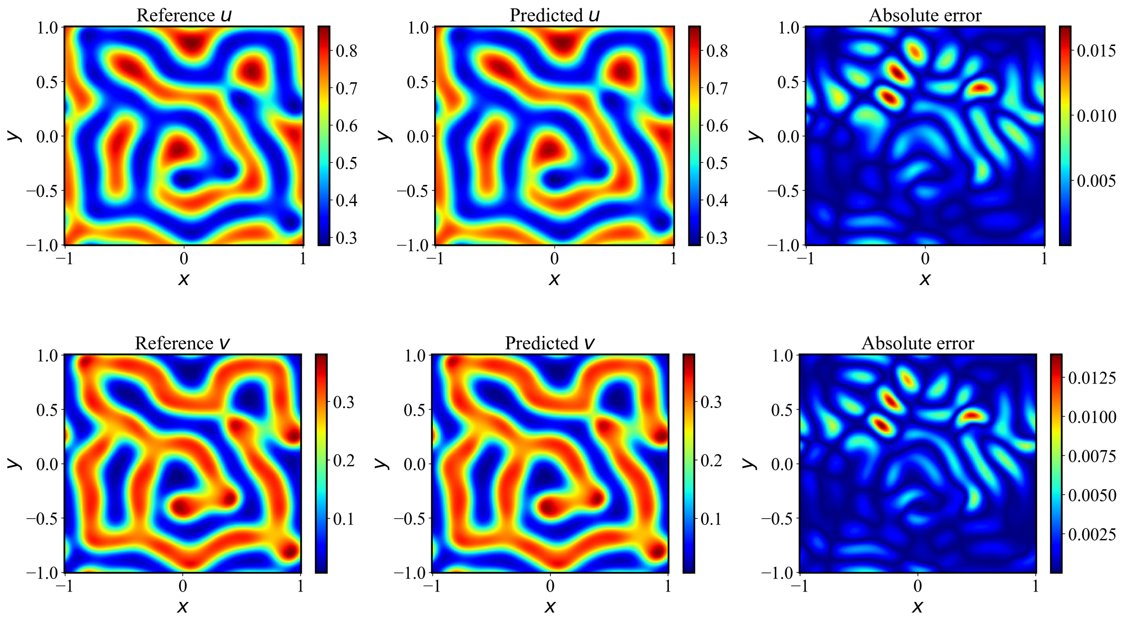


Figure 15: *Grey-Scott equation:* Comparisons between the solutions predicted by a trained HyResPINN and the reference solutions.

Here u and v represent the concentrations of the two species. The system can generate a wide range of patterns, including spots, stripes, and more complex forms, depending on the parameters chosen. For this example, we set $\epsilon_1 = 0.2, \epsilon_2 = 0.1, b_1 = 40, b_2 = 100, c_1 = c_2 = 1,000$. We employ a standard time-marching strategy to train our PINN models, as outlined in various studies (Wight & Zhao, 2020; Wang et al., 2023a). Full experimental details are provided in Appendix E.

For the Grey-Scott reaction–diffusion system, HyResPINN accurately captures the complex spatiotemporal patterns characteristic of multicomponent reaction–diffusion systems. The model's predictions align closely with the ground truth across the domain, as visualized in Figure 15. In terms of efficiency per DoF, HyResPINN achieves competitive error rates compared to state-of-the-art PINN models, with relative L^2 errors of 6.43×10^{-3} for u and 1.31×10^{-2} for v (see Table 7).

6 Conclusion

In this work, we introduced HyResPINNs, a novel class of physics-informed neural networks that integrate adaptive hybrid residual blocks, combining the complementary strengths of DNNs and RBFNNs. The use of adaptive gating parameters allows the model to dynamically balance the contributions of DNN and RBFNN components throughout training, resulting in enhanced expressivity and adaptivity. Through our experiments on a diverse set of benchmark PDEs, we demonstrated that HyResPINNs consistently outperform traditional PINNs and state-of-the-art neural PDE solvers in accuracy while maintaining computational efficiency. Our method excels particularly in problems featuring mixed smooth and non-smooth solution features, where existing approaches often struggle. Future directions include extending this hybrid residual architecture to more complex, high-dimensional problems, as well as further investigating its interpretability and applicability in real-world engineering and scientific domains.

References

Ben Adcock and Nick Dexter. The gap between theory and practice in function approximation with deep neural networks, 2021. URL https://arxiv.org/abs/2001.07523.

Mark Ainsworth and J Tinsley Oden. A posteriori error estimation in finite element analysis. *Computer methods in applied mechanics and engineering*, 142(1-2):1–88, 1997.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in neural information processing systems*, 32, 2019.

Ivo Babuvska and Manil Suri. The p and hp versions of the finite element method, basic principles and properties. *SIAM Review*, 36(4):578–632, 1994.

Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 2002.

Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer, 2008.

Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta numerica*, 13:147–269, 2004.

Zhang Chen, Zhong Li, Liangchen Song, Lele Chen, Jingyi Yu, Junsong Yuan, and Yi Xu. Neurbf: A neural fields representation with adaptive radial basis functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4182–4194, 2023.

Madison Cooley, Varun Shankar, Mike Kirby, and Shandian Zhe. Fourier PINNs: From strong boundary conditions to adaptive fourier bases. *Transactions on Machine Learning Research*, 2025a. ISSN 2835-8856. URL `https://openreview.net/forum?id=KqRnsEMYLx`.

Madison Cooley, Shandian Zhe, Robert M. Kirby, and Varun Shankar. Polynomial-augmented neural networks (panns) with weak orthogonality constraints for enhanced function and pde approximation. *arXiv preprint 2406.02336*, 2025b.

Steven M Cox and Paul C Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Arka Daw, Jie Bu, Sifan Wang, Paris Perdikaris, and Anuj Karpatne. Rethinking the importance of sampling in physics-informed neural networks. *arXiv preprint arXiv:2207.02338*, 2022.

Leszek Demkowicz, John Kurtz, David Pardo, Maciej Paszynski, Waldemar Rachowicz, and Adam Zdunek. Fully automatic hp-adaptive 2d fem algorithms. *Computer Methods in Applied Mechanics and Engineering*, 191(47–48):4717–4757, 2002.

Ronald DeVore, Boris Hanin, and Guergana Petrova. Neural network approximation, 2020. URL `https://arxiv.org/abs/2012.14501`.

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.

Tobin A Driscoll, Nicholas Hale, and Lloyd N Trefethen. Chebfun guide, 2014.

Lawrence C. Evans. *Partial Differential Equations*. American Mathematical Society, 2nd edition, 2010.

Gregory E Fasshauer. *Meshfree approximation methods with Matlab (With Cd-rom)*, volume 6. World Scientific Publishing Company, 2007.

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*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Michael Hauser. On residual networks learning a perturbation from identity, 2019. URL `https://arxiv.org/abs/1902.04106`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. doi: 10.1016/0893-6080(89)90020-8.

Amanda A Howard, Sarah H Murphy, Shady E Ahmed, and Panos Stinis. Stacked networks improve physics-informed training: Applications to neural networks and deep operator networks. *Foundations of Data Science*, 7(1):134–162, 2025.

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):2002–2041, 2020.

Ameya D. Jagtap, Ehsan Kharazmi, and George Em 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. doi: 10.1016/j.cma.2020.113028.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

E. Kharazmi, Z. Zhang, and G. E. Karniadakis. Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems.* SIAM, 2007.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

Dawei Luo, Soo-Ho Jo, and Taejin Kim. Progressive domain decomposition for efficient training of physics-informed neural network. *Mathematics*, 13(9), 2025. doi: 10.3390/math13091515.

Tanya Marwah, Zachary C. Lipton, and Andrej Risteski. Parametric complexity bounds for approximating pdes with neural networks, 2021. URL `https://arxiv.org/abs/2103.02138`.

Arif Masud and Thomas JR Hughes. A stabilized mixed finite element method for darcy flow. *Computer methods in applied mechanics and engineering*, 191(39-40):4341–4370, 2002.

Levi McClenny and Ulisses Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*, 2020.

Jens Markus Melenk. On the robust exponential convergence of hp finite element methods for problems with boundary layers. *IMA Journal of Numerical Analysis*, 17(4):577–601, 1997.

Xuhui Meng, Zhen Li, Dongkun Zhang, and George Em Karniadakis. Ppinn: Parareal physics-informed neural network for time-dependent pdes. *Computer Methods in Applied Mechanics and Engineering*, 370: 113250, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.113250. URL `https://www.sciencedirect.com/science/article/pii/S0045782520304357`.

KB Nakshatrala, DZ Turner, KD Hjelmstad, and Arif Masud. A stabilized mixed finite element method for darcy flow based on a multiscale decomposition of the solution. *Computer Methods in Applied Mechanics and Engineering*, 195(33-36):4036–4049, 2006.

A. Noorizadegan, R. Cavoretto, D. L. Young, and C. S. Chen. Stable weight updating: A key to reliable pde solutions using deep learning, 2024a. URL `https://arxiv.org/abs/2407.07375`.

A Noorizadegan, R Cavoretto, Der-Liang Young, and CHUIN-SHAN Chen. Stable weight updating: A key to reliable pde solutions using deep learning. *Engineering Analysis with Boundary Elements*, 168:105933, 2024b.

Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks, 2018.

Guofei Pang, Marta D'Elia, Michael Parks, and George E. Karniadakis. npinns: nonlocal physics-informed neural networks for a parametrized nonlocal universal laplacian operator. algorithms and applications, 2020.

Nicolas Papernot, Abhradeep Thakurta, Shuang Song, Steve Chien, and Úlfar Erlingsson. Tempered sigmoid activations for deep learning with differential privacy, 2020. URL `https://arxiv.org/abs/2007.14191`.

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*, 378:686–707, 2019a. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045.

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, 2019b.

Robert Schaback and Holger Wendland. Adaptive greedy techniques for approximate solution of large rbf systems. *Numerical Algorithms*, 24(3):239–254, 2000.

Christoph Schwab. *p- and hp-Finite Element Methods: Theory and Applications in Solid and Fluid Mechanics*. Oxford University Press, 1998.

Ayush Sekhari, Karthik Sridharan, and Satyen Kale. Sgd: The role of implicit regularization, batch-size and multiple-epochs. *Advances In Neural Information Processing Systems*, 34:27422–27433, 2021.

Varun Shankar. The overlapped radial basis function-finite difference (RBF-FD) method: A generalization of RBF-FD. *J. Comput. Phys.*, 342:211–228, 2017.

Ramansh Sharma and Varun Shankar. Accelerated training of physics-informed neural networks (PINNs) using meshless discretizations. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=NYpU9BRODos`.

Jie Shen, Tao Tang, and Li-Lian Wang. *Spectral methods: algorithms, analysis and applications*, volume 41. Springer Science & Business Media, 2011.

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

Gilbert Strang, George J Fix, and DS Griffin. An analysis of the finite-element method. *Journal of Applied Mechanics*, 41(1):62, 1974.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

Nathaniel Trask, Mauro Perego, and Pavel Bochev. A high-order staggered meshless method for elliptic problems. *SIAM Journal on Scientific Computing*, 39(2):A479–A502, 2017. doi: 10.1137/16M1055992.

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, 2021a.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021b.

Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*, 2022a.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022b.

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

Sifan Wang, Bowen Li, Yuhan Chen, and Paris Perdikaris. Piratenets: Physics-informed deep learning with residual adaptive networks. *arXiv preprint arXiv:2402.00326*, 2024.

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

Bernard Widrow and Michael A Lehr. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 2002.

Colby L Wight and Jia Zhao. Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542*, 2020.

Yue Wu, Hui Wang, Biaobiao Zhang, and K-L Du. Using radial basis function networks for function approximation and classification. *International Scholarly Research Notices*, 2012(1):324194, 2012.

Fabian Wurzberger and Friedhelm Schwenker. Learning in deep radial basis function networks. *Entropy*, 26 (5):368, 2024.

X. I. A. Yang, S. Zafar, J.-X. Wang, and H. Xiao. Predictive large-eddy-simulation wall modeling via physics-informed neural networks. *Phys. Rev. Fluids*, 4:034602, Mar 2019. doi: 10.1103/PhysRevFluids.4.034602. URL `https://link.aps.org/doi/10.1103/PhysRevFluids.4.034602`.

Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.

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.

Shaojie Zeng, Zong Zhang, and Qingsong Zou. Adaptive deep neural networks methods for high-dimensional partial differential equations. *Journal of Computational Physics*, 463:111232, 2022.

Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2019.07.048. URL `https://www.sciencedirect.com/science/article/pii/S0021999119305340`.

A Allen-Cahn with Dirichlet Boundary Conditions

A.1 Reference Solution Generation

We solve the equation numerically using a semi-implicit finite difference method. The spatial domain is discretized uniformly with 1042 grid points, and the temporal domain is discretized into 600 steps. The nonlinear reaction term is treated explicitly, while the linear diffusion term is handled implicitly, forming an IMEX (implicit-explicit) Euler scheme. Homogeneous Dirichlet boundary conditions are imposed at each time step by modifying the linear system accordingly. Solutions are recorded by saving the solution $u(x, t)$ at all time steps along with the initial condition and spatial-temporal grids.

A.2 Ablation Study: Impact of RBF Kernels and Center Density

To evaluate the impact of different radial basis function (RBF) kernels on solution quality, we conduct an ablation study comparing several anisotropic RBF formulations within the hybrid PINN framework. The goal is to determine how the kernel's support, smoothness, and decay behavior affect the model's ability to capture localized features, represent sharp interfaces, and maintain global solution coherence. The architecture, initialization, and training procedure are held constant across trials to isolate the influence of the kernel's functional form. Table 1 summarizes the five kernels studied.

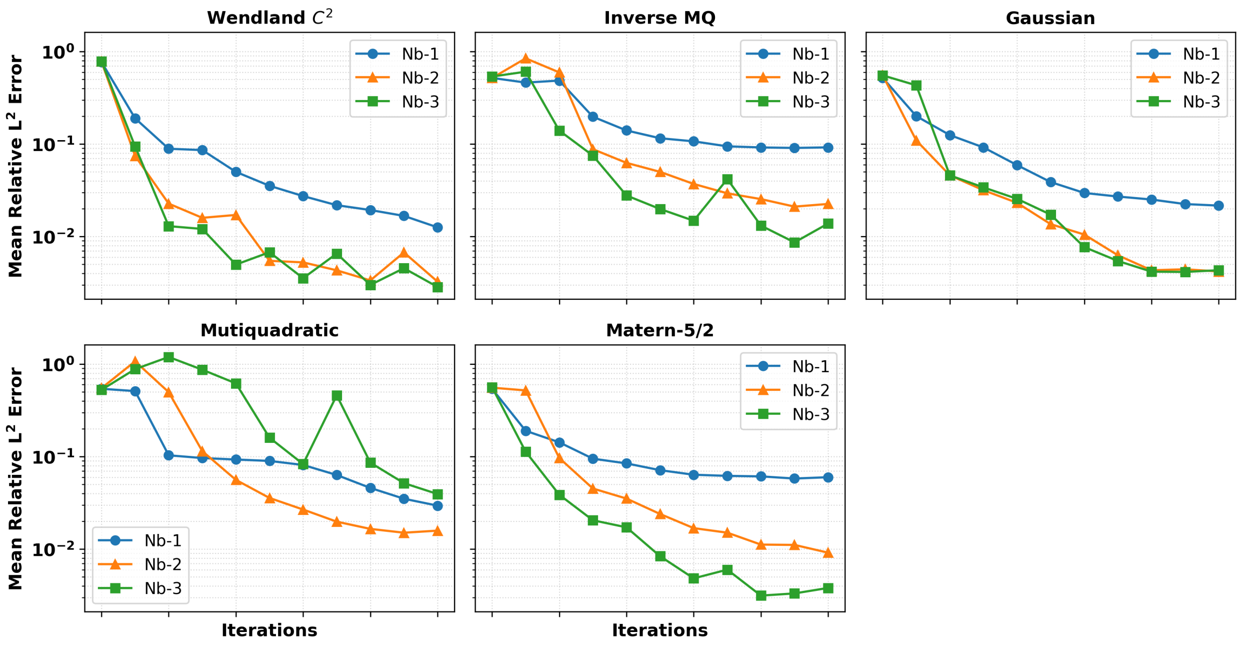


Figure 16: *Allen–Cahn equation:* Prediction errors for various RBF kernels within the HyResPINN framework using 1, 2, and 3 blocks.

Results demonstrate that kernel choice substantially influences the model's capacity to resolve steep gradients and preserve global coherence. The best overall performance is achieved by the Wendland C^2 kernel, which we adopt for all subsequent experiments, as it offers a slightly better trade-off between accuracy, computational efficiency, compact support, and C^2 smoothness which make it well-suited for the localized dynamics. In addition to kernel type, we also investigate how the number of RBF centers affects predictive performance, using the Wendland C^2 kernel. Results are shown in Table 9 which show that a balance between resolution and overfitting emerges, with best performance at 64 centers.

Name	**Computational Cost**
Gaussian(Nb-1)	0.020064 (0.00067)
Gaussian(Nb-2)	0.040506 (0.00044)
Gaussian(Nb-3)	0.062309 (0.00045)
Inverse MQ(Nb-1)	0.020070 (0.00057)
Inverse MQ(Nb-2)	0.040792 (0.00024)
Inverse MQ(Nb-3)	0.062468 (0.00075)
Matérn-5/2(Nb-1)	0.020033 (0.00076)
Matérn-5/2(Nb-2)	0.040725 (0.00087)
Matérn-5/2(Nb-3)	0.064149 (0.00043)
Multiquadric(Nb-1)	0.019847 (0.00080)
Multiquadric(Nb-2)	0.040620 (0.00077)
Multiquadric(Nb-3)	0.062131 (0.00044)
Wendland C^2(Nb-1)	0.021837 (0.00071)
Wendland C^2(Nb-2)	0.043131 (0.00043)
Wendland C^2(Nb-3)	0.065628 (0.00097)

Table 8: Computational cost measured as the average time in seconds to run 100 iterations for each RBF using 1,2, and 3 blocks. Standard deviations are in parenthesis.

Number of Centers	**Error (Std)**	**Computational Cost**
8	0.028290 (0.042173)	0.017529 (2.634898)
16	0.007372 (0.002045)	0.017578 (3.782532)
32	0.003666 (0.001104)	0.017849 (6.860452)
64	0.002462 (0.000063)	0.018479 (2.414555)
128	0.003265 (0.001240)	0.019347 (7.184156)
256	0.002938 (0.000411)	0.021444 (6.183969)

Table 9: Effect of increasing the number of RBF centers on HyResPINN mean relative L^2 error and computational cost measured as the average wall-clock time (in seconds) required to perform 100 iterations. Standard deviations in parentheses.

B Allen-Cahn with Periodic Boundary Conditions

B.1 Reference Solution Generation

Following the approach in (Wang et al., 2024), we employ standard spectral methods to solve the Allen–Cahn equation under periodic boundary conditions. Starting from the initial state $u_0(x) = x^2 \cos(\pi x)$, we integrate the equation up to a final time of $T = 1$. For generating synthetic validation data, we use the Chebfun (Driscoll et al., 2014) package to perform a Fourier spectral discretization with 512 modes. Time integration is carried out using the fourth-order exponential time-differencing Runge–Kutta method (ETDRK4) (Cox & Matthews, 2002) with a time step of 10^{-5}. Solutions are sampled every $\Delta t = 0.005$, resulting in a high-resolution validation dataset of size 200×512.

C Kuramoto-Sivashinsky Equation

C.1 Reference Solution Generation

We employ standard spectral methods to solve the Kuramoto-Sivashinsky equation under periodic boundary conditions. For generating synthetic validation data, we use the Chebfun (Driscoll et al., 2014) package to perform a Fourier spectral discretization with 512 modes. Time integration is carried out using the fourth-order exponential time-differencing Runge–Kutta method (ETDRK4) (Cox & Matthews, 2002) with a time step of 10^{-5}. Solutions are sampled every $\Delta t = 0.004$, resulting in a high-resolution validation dataset of size 250×512.

D Korteweg-De Vries Equation

D.1 Reference Solution Generation

Similar to (Wang et al., 2024), we generate synthetic validation data for the Korteweg–De Vries equation, we use standard spectral methods under periodic boundary conditions. The simulation begins with the initial condition set to $u_0(x) = \cos(\pi x)$ and is integrated forward in time up to $T = 1$. The Chebfun package (Driscoll et al., 2014) is utilized for a Fourier spectral discretization employing 512 modes. Time integration is performed with the fourth-order exponential time-differencing Runge–Kutta (ETDRK4) method (Cox & Matthews, 2002), using a step size of 10^{-5}. The solution is sampled every $\Delta t = 0.005$, resulting in a validation dataset of size 200×512 covering the temporal and spatial domains.

E Grey-Scott Equation

E.1 Reference Solution Generation

For data generation, we solve the Grey-Scott equation using standard spectral techniques under periodic boundary conditions following the details provided in (Wang et al., 2024). The system is initialized with the prescribed initial condition and integrated up to a final time of $T = 2$. Synthetic validation data are produced with the Chebfun package (Driscoll et al., 2014), employing a 2D Fourier spectral discretization with 200×200 modes. Temporal integration is performed using the fourth-order exponential time-differencing Runge–Kutta (ETDRK4) method (Cox & Matthews, 2002) with a time step of 10^{-3}. Solutions are saved every $\Delta t = 0.02$, resulting in a validation dataset of dimensions $100 \times 200 \times 200$ spanning both the temporal and spatial domains.