HYRESPINNS: ADAPTIVE HYBRID RESIDUAL NET WORKS FOR LEARNING OPTIMAL COMBINATIONS OF NEURAL AND RBF COMPONENTS FOR PHYSICS INFORMED MODELING

Anonymous authors

Paper under double-blind review

ABSTRACT

Physics-informed neural networks (PINNs) are an increasingly popular class of techniques for the numerical solution of partial differential equations (PDEs), where neural networks are trained using loss functions regularized by relevant PDE terms to enforce physical constraints. We present a new class of PINNs called HyResPINNs, which augment traditional PINNs with adaptive hybrid residual blocks that combine the outputs of a standard neural network and a radial basis function (RBF) network. A key feature of our method is the inclusion of adaptive combination parameters within each residual block, which dynamically learn to weigh the contributions of the neural network and RBF network outputs. Additionally, adaptive connections between residual blocks allow for flexible information flow throughout the network. We show that HyResPINNs are more robust to training point locations and neural network architectures than traditional PINNs. Moreover, HyResPINNs offer orders of magnitude greater accuracy than competing methods on certain problems, with only modest increases in training costs. We demonstrate the strengths of our approach on challenging PDEs, including the Allen-Cahn equation and the Darcy-Flow equation. Our results suggest that HyResPINNs effectively bridge the gap between traditional numerical methods and modern machine learning-based solvers.

031 032

008

009

010 011 012

013

015

016

017

018

019

021

023

025

026

027

028

029

033 034

1 INTRODUCTION

Partial differential equations (PDEs) model a variety of phenomena across science and engineering and are traditionally solved using numerical methods such as finite difference methods (LeVeque, 2007) 037 and finite elements (Strang et al., 1974). Physics-informed neural networks (PINNs) are meshless methods (Raissi et al., 2019; Raissi, 2018) that solve PDEs by training deep feedforward neural networks (DNNs) using PDEs as soft constraints. This traditional PINN training method poses challenges 040 due to complicated loss landscapes arising from the PDE-based soft constraints (Krishnapriyan et al., 041 2021). Recent approaches to ameliorating issues include curriculum learning (Krishnapriyan et al., 042 2021), novel optimization techniques (Cyr et al., 2020), domain decomposition (X-PINNs) (Jagtap & 043 Karniadakis, 2020), gradient-enhanced training (G-PINNs) (Yu et al., 2022a), or discretely-trained 044 PINNs using RBF-FD approximations in place of automatic differentiation (DT-PINNs) (Sharma & Shankar, 2022). Further significant strides have been made in developing new DNN architectures to enhance PINNs' representational capacity, including adaptive activation functions Jagtap et al. 046 (2020b;a), positional embeddings Liu et al. (2020); Wang et al. (2021b), and innovative architec-047 tures Wang et al. (2021a); Sitzmann et al. (2020); Gao et al. (2021); Fathony et al. (2020); Moseley 048 et al. (2023); Kang et al. (2023). 049

In addition to architectural choices, certain training pathologies exist including, spectral bias (Rahaman et al., 2019; Wang et al., 2021b), unbalanced back-propagated gradients (Wang et al., 2021a; 2022b), and causality violations (Wang et al., 2022a; 2024b). Efforts focused on improving PINNs' training performance include loss re-weighting schemes (Wang et al., 2021a; 2022b; McClenny & Braga-Neto, 2020; 2023; Maddu et al., 2022) and adaptive resampling of collocation points, such as

importance sampling (Nabian et al., 2021), evolutionary sampling (Daw et al.), and residual-based adaptive sampling (Wu et al., 2023).

Combining traditional methods with DNNs is a new line of research which focuses on leveraging the 057 advantages of both. For example, incorporating a Fourier feature layer to preprocess DNN inputs enhances their capacity to capture high-frequency functions (Tancik et al., 2020), while also reducing eigenvector bias in PINNs (Wang et al., 2021b; Raynaud et al., 2022). Fourier neural operators (Li 060 et al., 2020) utilize Fourier layers in the network architecture and have become a popular approach for 061 inverse problems. Similarly, physics-informed radial basis networks (PIRBNs) have been proposed 062 as efficient local approximators that integrate domain knowledge during training, performing well in 063 solving PDEs with high-frequency features or ill-posed domains (Bai et al., 2023; Fu et al., 2024). 064 Chrysos et al. (2022; 2020) augment polynomial neural networks with DNNs, with a direct focus on classification and discriminative problems. Further explorations encompass alternative objective 065 functions, such as those employing numerical differentiation techniques Huang & Alkhalifah (2024) 066 and variational formulations inspired by Finite Element Methods (FEM) Kharazmi et al. (2019); 067 Berrone et al. (2022); Patel et al. (2022), along with additional regularization terms to expedite 068 PINNs' convergence Yu et al. (2022b); Son et al. (2021). 069

Recent work has investigated deep architectures for solving PDEs through methods such as stacked
DNNs (Howard et al., 2023) and residual blocks with adaptive gating parameters (Wang et al., 2024a).
Wang et al. (2024a) incorporate residual blocks with adaptive skip connections, which dynamically
balance the input and learned residuals at each layer. Specifically, they incorporate adaptive residual
blocks which dynamically balance the contributions of both the input and residual function at each
layer, while the stacked approach of Howard et al. (2023) utilizes transfer learning to improves the
ability of PINNs to handle complex, multi-scale PDEs.

While these deep residual based approaches show much promise, the increased architectural complexity, leads to higher computational costs both in terms of memory and training time—requiring careful selection of training routines to prevent instabilities or poor convergence. Finally, while Fourier features help in capturing high-frequency components, they can still struggle with discontinuous solutions or sharp interfaces, which are challenging for neural networks to approximate due to the smoothness of typical activation functions.

To address these challenges, we propose a novel class of network architectures termed HyResPINNs, 083 that combines a standard DNN with a Radial Basis Function (RBF) network within bi-level adaptive 084 residual blocks, such that the relative contribution of both the DNN and RBF networks are adaptively 085 learned during training, along with the full residual block outputs. The proposed architecture leverages the strengths of both smooth and non-smooth function approximators to model complex physical 087 systems. The standard DNN using smooth activation functions effectively captures the continuous, 880 global behaviors of the solution, which are common in many physical phenomena. While the RBF 089 network then captures the localized, discontinuous or sharp features in the target solution. This representational division ensures that the model can accurately capture both smooth and non-smooth 091 components of the solution, forming a robust network capable of modeling many types of physical 092 systems.

Our main contributions are summarized as follows:

095

096

098 099

102

103

- **Propose a novel adaptive residual architecture:** We introduce a new class of physicsinformed neural networks, HyResPINN, that combine standard neural networks with Radial Basis Function (RBF) networks within bi-level adaptive residual blocks.
- **Demonstrate the superiority of the block structure:** We show that our residual block architecture provides significant improvements over standard approaches in capturing both smooth and non-smooth features, leading to more accurate modeling of complex physical systems.
- Highlight the benefits of adaptivity between residual blocks: We demonstrate that the adaptive learning of contributions between the NN and RBF networks, as well as between residual blocks, results in superior performance and stability compared to non-adaptive methods.

• **Thorough empirical evaluation:** Show our method HyResPINN outperforms standard PINNs along with state of the art methods in PINNs on an array of baseline problems, confirming the general applicability of HyResPINNs.

2 BACKGROUND

2.1 Physics-informed neural network

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

$$\mathbf{u}_t + \mathcal{F}[\mathbf{u}] = \mathbf{f} \,, \tag{1}$$

such that $\mathcal{F}[\cdot]$ is a linear or nonlinear differential operator, and $\mathbf{u}(t, \mathbf{x})$ denotes a unknown solution.

The general initial and boundary conditions can be then formulated as:

$$\mathbf{u}(0,\mathbf{x}) = \mathbf{g}(\mathbf{x}), \quad \mathbf{x} \in \Omega,$$
(2)

$$\mathcal{B}[\mathbf{u}] = 0, \quad t \in [0, T], \ \mathbf{x} \in \partial\Omega.$$
(3)

Here, **f** and $\mathbf{g}(\mathbf{x})$ are given functions with certain regularity; $\mathcal{B}[\cdot]$ denotes an abstract boundary operator, representing various boundary conditions such as Dirichlet, Neumann, Robin, and periodic conditions.

We aim at approximating the unknown solution $\mathbf{u}(t, \mathbf{x})$ by a deep neural network $\mathbf{u}_{\theta}(t, \mathbf{x})$, where θ denotes the set of all trainable parameters of the network (e.g., weights and biases). If a smooth activation function is employed, \mathbf{u}_{θ} provides a smooth representation that can be queried for any (t, \mathbf{x}) . The PDE residuals are defined as,

$$\mathcal{R}_{\rm int}[\mathbf{u}_{\theta}](t,\mathbf{x}) = \frac{\partial \mathbf{u}_{\theta}}{\partial t}(t,\mathbf{x}) + \mathcal{F}[\mathbf{u}_{\theta}](t,\mathbf{x}) - \mathbf{f}(\mathbf{x}), \quad (t,\mathbf{x}) \in [0,T] \times \Omega,$$
(4)

and spatial and temporal boundary residuals, respectively, by

$$\mathcal{R}_{\rm bc}[\mathbf{u}_{\theta}](t,\mathbf{x}) = \mathcal{B}[\mathbf{u}_{\theta}](t,\mathbf{x}), \quad (t,\mathbf{x}) \in [0,T] \times \partial\Omega,$$
(5)

and

$$\mathcal{R}_{ic}[\mathbf{u}_{\theta}](\mathbf{x}) = \mathbf{u}_{\theta}(0, \mathbf{x}) - \mathbf{g}(\mathbf{x}), \quad \mathbf{x} \in \Omega.$$
(6)

Then, we train a physics-informed model by minimizing the following composite *empirical loss*:

$$\mathcal{L}(\theta) := \underbrace{\frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} \left| \mathcal{R}_{ic}[\mathbf{u}_{\theta}](\mathbf{x}_{ic}^{i}) \right|^{2}}_{\mathcal{L}_{ic}(\theta)} + \underbrace{\frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} \left| \mathcal{R}_{bc}[\mathbf{u}_{\theta}](t_{bc}^{i}, \mathbf{x}_{bc}^{i}) \right|^{2}}_{\mathcal{L}_{bc}(\theta)} + \underbrace{\frac{1}{N_{r}} \sum_{i=1}^{N_{r}} \left| \mathcal{R}_{int}[\mathbf{u}_{\theta}](t_{r}^{i}, \mathbf{x}_{r}^{i}) \right|^{2}}_{\mathcal{L}_{r}(\theta)},$$
(7)

which aims to enforce the neural network function \mathbf{u}_{θ} to satisfy the PDEs (1) with initial and spatial boundary conditions (2)–(3). 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 the vertices of a fixed mesh or points randomly sampled at each iteration of a gradient descent algorithm.

³ 2.2 RADIAL BASIS FUNCTION NETWORKS

The general form of an RBF network for the training data points $\mathbf{x} = {\mathbf{x}_i}_{i=1}^N$ with corresponding function values y_i is given by,

$$f(\mathbf{x}) = \sum_{j=1}^{N_c} c_j \psi(||\mathbf{x} - \mathbf{x}_j^c||),$$
(8)

where $\mathbf{x}^c = {\{\mathbf{x}_j^c\}}_{j=1}^{N_c}$ are the center points of the radial basis functions (RBFs). Here, ψ represents the radial basis function, which are commonly chosen as the Gaussian, multi-quadric, or inverse

```
153
154
```

multi-quadric functions, each offering different characteristics and benefits. The elements of the kernel matrix are given by:

$$\mathbf{K}_{ij} = \psi(||\mathbf{x}_i - \mathbf{x}_j^c||),\tag{9}$$

where \mathbf{x}_i represent the data points, and (9) evaluates the RBF based on the distance between the points. The kernel matrix captures the pairwise interactions between data points through the chosen RBF, and its structure is typically symmetric and positive-definite, assuming an appropriate choice of ψ . The choice of center points can vary depending on the application; in some cases, the center points may coincide with the data points, but they may also be selected independently of the data points to optimize approximation accuracy or computational efficiency. The RBFs determine how much influence each center has on the input, based on the distance between the input and the center.



Figure 1: Illustration of the RBF+PINN hybrid residual block with trainable strength connections between the RBF and PINN outputs.

Gradient descent is sometimes used to optimize both the centers and the unknown weights (c_j) simultaneously. These approaches are referred to as RBF networks, where the radial basis functions are the hidden layer activations, and the RBF coefficients are the trainable network parameters. The mean-squared error between the linear combination of the hidden layer outputs and the known functions values is commonly minimized.

3 HYBRID RESIDUAL PINNS (HYRESPINNS)

In this section, we describe our proposed architecture—HyResPINNs. HyResPINNs are a novel type of residual network such that each residual block incorporates an RBF kernel along with a standard DNN.

213 3.1 HYBRID RESIDUAL BLOCKS

Figure 1 shows the hybrid residual block specifics. We denote the output of the *l*-th residual block for the input $\mathbf{x}^{(l)}$ as $H^{(l)}(\mathbf{x}^{(l)})$. Formally, the forward pass of each hybrid residual block in the HyResPINN architecture is,

220

221 222

223 224

229 230

231 232

252 253 254

260

261 262 263

266

267 268 269

$$H^{(l)}(\mathbf{x}^{(l)}) = \phi(\alpha^{(l)}) F_R^{(l)}(\mathbf{x}^{(l)}) + (1 - \phi(\alpha^{(l)})) F_N^{(l)}(\mathbf{x}^{(l)}),$$
(10)

such that $\alpha^{(l)} \in \mathbb{R}$ is a trainable parameter, ϕ represents the sigmoid function, $F_R^{(l)}(\mathbf{x}^{(l)})$ is the output of the RBF network and $F_N^{(l)}(\mathbf{x}^{(l)})$ is the output from a DNN or PINN within the *l*-th block. We then pass $\sigma(H^{(l)}(\mathbf{x}^{(l)}))$ as the input to the next block, where σ is some non-linear activation function.



Figure 2: Illustration of the HyResPINN architecture using three blocks.

Adaptive Hybrid Contribution Coefficients The trainable parameter $\alpha^{(l)}$ controls the relative contributions of the DNN and RBF components to the output—where optimizing α finds the best balance between these two elements. Since the sigmoid function ϕ constrains its output to the range [0, 1], the resulting output is a convex combination of the RBF and DNN components. We further incorporate adaptable residual connection parameters between each hybrid block, denoted as $\beta^{(l)}$ —similar to the approaches in (Howard et al., 2023) and (Wang et al., 2024a)

239 However, unlike (Wang et al., 2024a), where their adaptable parameters are initialized to zero to force 240 the network to learn non-linearities from scratch, we initialize each $\phi(\alpha^{(l)}) = 0.5$, ensuring equal 241 contributions from both components at the start of training, and each $\beta^{(l)} = 1$. When $\phi(\alpha) = 1$, 242 only the RBF network contributes; when $\phi(\alpha) = 0$, only the DNN contributes. The optimal choice 243 of α depends on the problem characteristics. For problems with large regions of smoothness, the 244 model might favor a lower α , assigning more weight to the DNN components in each residual block. 245 Conversely, for problems involving sharp transitions or discontinuities, a higher α (favoring the RBF) may be preferable. In hybrid problems with smooth and non-smooth regions, α will likely fall 246 between 0.4 and 0.6, providing a balanced combination of the two networks. 247

To encourage smoother solutions and prevent the network from introducing excessive non-linearity, we add a regularization term to each block's trainable parameter α . This regularization penalizes large values of α , effectively controlling the contribution of non-linearity from the RBF components. Specifically, the total loss function is defined as:

$$\mathcal{L} = \lambda_{ic} \mathcal{L}_{ic}(\boldsymbol{\theta}) + \lambda_{bc} \mathcal{L}_{bc}(\boldsymbol{\theta}) + \lambda_r \mathcal{L}_r(\boldsymbol{\theta}) + \lambda_p \sum_{i=1}^{N_{blocks}} \alpha_i^2,$$
(11)

where \mathcal{L}_{ic} and \mathcal{L}_{bc} represent the loss terms for the initial and boundary conditions, \mathcal{L}_r represents the residual loss, and the final term $\lambda_p \sum_{i=1}^{N_{blocks}} \alpha_i^2$ acts as an L_2 regularization on the α parameters. The regularization helps balance the contributions of the smooth neural network and non-linear RBF components, promoting smoother solutions and more stable training.

Adaptive RBF Block Kernel The output of the RBF kernel $F_{R}^{(l)}(\mathbf{x}^{(l)})$ can be described as,

$$F_R^{(l)}(\mathbf{x}^{(l)}) = K(\mathbf{x})W,\tag{12}$$

such that the kernel matrix K is defined as (9) where the kernel matrix K is based on the Wendland C^4 kernel in 2D. In this work, we focus on isotropic version of the Wendland kernel.

The isotropic Wendland C^4 kernel for input x and the *i*-th center is defined as:

$$\psi_i(\mathbf{x}) = \left(1 - \frac{\|\mathbf{x} - \mathbf{x}_i^c\|}{\tau_i}\right)_+^6 \left(35 \left(\frac{\|\mathbf{x} - \mathbf{x}_i^c\|}{\tau_i}\right)^2 + 18 \frac{\|\mathbf{x} - \mathbf{x}_i^c\|}{\tau_i} + 3\right),\tag{13}$$



Figure 3: 2D Allen-Cahn equation: Comparison of (a subset of) the learned RBF kernels for a standard RBF network (top row) and the proposed hybrid RBF+NN residual block approach (bottom row). Each subplot shows the RBF value corresponding to different learned RBF centers in the input domain, marked by red crosses.

where \mathbf{x}_{i}^{c} is the center of the *i*-th RBF, τ is a scaling parameter, and $\|\mathbf{x} - \mathbf{x}_{i}^{c}\|$ represents the Euclidean distance between x and x_i^c . The Wendland kernel is compactly supported, meaning $\psi_i(\mathbf{x}) = 0$ for $\|\mathbf{x} - \mathbf{x}_i^c\| \ge \tau_i$, which leads to sparse kernel matrices and computational efficiency. Each τ_i is a trainable parameter in the network and optimize through gradient descent along with all other network parameters. Figure 3 demonstrates learned scale parameters within an optimized RBF kernel.

Block Neural Network Following the convention of Cyr et al. (2020), we represent the output of the DNN $F_N^{(l)}(\mathbf{x}^{(l)})$ $F_N^{(l)} \in \mathbb{R}^d \to \mathbb{R}$ of width w, as a linear combination of adaptive basis functions given by

$$F_N^{(l)}(\mathbf{x}; a, \theta^h) = \sum_{i=1}^w a_i \sigma_i(\mathbf{x}; \theta^h), \tag{14}$$

301 where each a_j for j = 1, ..., w and θ^h constitute the weights and biases in the last layer and hidden 302 layers respectively, forming the set of all network parameters θ . Then, each σ_i are non-linear smooth 303 activation functions such as Tanh acting on the outputs of the hidden layers. We choose to use a standard DNN architecture, but any DNN architecture such as ResNets, would work. The parameters 304 θ are computed through some iterative optimization technique. In this work, we use variants of 305 gradient descent methods such as ADAM Kingma & Ba (2015) and L-BFGS Liu & Nocedal (1989). 306

Figure 2 shows a visualization of our full model architecture with adaptive residual block skip 308 connections, along with the input block structure used to lift the inputs to the desired higher dimension, 309 and the output neural network block used to project each block's output down to the output dimension.

311	Problem	Domain	Boundary Cond.	PINN	ResPINN	Expert	Stacked	PirateNet	HyResPINN
312	Allen-Cahn	1D Space/Time	Periodic	0.526	0.0027	0.00386	0.00587	0.00022	$9.62\times\mathbf{10^{-5}}$
313	DarcyFlow	2D Annulus	Neumann	0.00075	0.00046	0.0005	0.0009	8.71×10^{-5}	$5.44 imes10^{-5}$
314	(smooth coefficients)		Dirichlet	0.0020	0.0014	0.00012	0.0041	0.00017	$6.0 imes 10^{-5}$
		3D Annulus	Neumann	0.0022	0.012	0.021	0.054	0.039	0.0012
315			Dirichlet	0.0085	0.0061	0.0011	0.039	0.0013	0.0011
316	(rough coefficients)	2D Box	Neumann	6.85×10^{-5}	2.69×10^{-5}	0.00011	0.00015	5.44×10^{-5}	$1.05 imes 10^{-5}$

Table 1: Relative L^2 test error results for various PDE problems and baseline methods.

319 320 321

317

318

284

285

287 288

289

290

291

292

293

295

296

307

310

322 323

In this section, we demonstrate the effectiveness of the proposed HyResPINNs architecture across a

EXPERIMENTAL RESULTS AND DISCUSSION

diverse collection of benchmark problems compared to the leading baseline methods. We show that

HyResPINNs consistently produce lower relative L^2 errors between the predicted and ground truth solutions for the same training set size when compared to the applicable baselines trained under the same experimental procedures and hyperparameters. We compare each method on the 1D non-linear hyperbolic Allen-Cahn equation and the Darcy Flow equation in two and three dimensions with both Dirichlet and Neumann boundary conditions. Our main results are summarized in Table 1.

Baseline Methods. We compare the proposed 330 HyResPINN against a suite of baseline methods 331 under exactly the same hyper-parameter settings 332 in each example. Baselines include the standard 333 PINN as originally formulated in (Raissi et al., 334 2019) (PINN), a PINN based on the architectural guidelines proposed in Wang et al. (2023) (Ex-335 pertPINNs), PINNs with residual connections 336 (ResPINNs), PirateNets Wang et al. (2024a), 337 and the stacked PINN approach of (Howard 338 et al., 2023) (StackedPINNs). To ensure a fair 339 comparison, we implemented the approaches de-340 tailed in (Raissi et al., 2019; Wang et al., 2024a; 341 2023; Howard et al., 2023) following the archi-342 tectural details provided in each. 343

Experimental Setup. We follow similar exper-344 imental design procedures as those described 345 in Wang et al. (2022a; 2023; 2024a). A full de-346 scription of the experimental setups are included 347 in Appendix A, and exact hyper-parameter set-348 ting are detailed in Table 2. We train all models 349 on Nvidia A100 GPU running Centos 7.2. Code 350 for our methods and all compared baseline ap-351 proaches is written in libtorch 1.11 (the C++ version of PyTorch) and will be made publicly 352 353 available upon publication.

354 355

4.1 1D Allen-Cahn

356 We first focus on the Allen-Cahn equation, a 357 challenging benchmark for conventional PINN 358 models that has been extensively studied in re-359 cent literature (Wight & Zhao, 2020; Wang 360 et al., 2022a; Daw et al.). For simplicity, we 361 consider the one-dimensional case with a peri-362 odic boundary condition with $t \in [0, 1]$, and 363 $x \in [-1, 1]$:

$$u_t - 0.0001u_{xx} + 5u^3 - 5u = 0,$$

$$u(0, x) = x^2 \cos(\pi x),$$

$$u(t, -1) = u(t, 1), \ u_x(t, -1) = u_x(t, 1).$$



Figure 4: 2D Allen-Cahn equation: Comparison of the predicted solutions for the Allen-Cahn equation using HyResPINN, standard PINN, and RBF network models. The top row shows the predicted solutions at for HyResPINN (left), standard PINN (center), and RBF network (right). The second row illustrates the absolute error between the predicted solutions and the exact solution. The bottom row shows the predicted solutions for three time steps comparing the exact solution (green), HyResPINN (orange), and standard PINN (dashed blue). The HyResPINN model effectively captures both the smooth and sharp features of the solution, while the standard PINN struggles with sharp transitions, leading to larger errors in these regions.

The Allen-Cahn PDE is an interesting benchmark for PINNs as it introduces periodic boundary conditions, and because it is a "stiff" PDE that challenges PINNs to approximate solutions with sharp space and time transitions.

371

364

366

367

Hybrid residual blocks improve prediction accuracy. We first demonstrate that the hybrid residual block structure introduced in our method enhances the ability to capture both smooth and sharp features in the 2D Allen-Cahn solution. As shown in Figure 4, the proposed approach outperforms competing baselines, which struggle to accurately represent the sharp transitions in the solution. This difficulty arises from the smoothness constraints of standard neural network architectures. In contrast, our method effectively captures these sharp transitions by leveraging smooth neural networks and adaptive non-smooth RBF kernels, allowing the model to balance global smooth behavior and

378 localized discontinuities. Figure 3 shows a random subset of the learned RBF kernels captured by 379 the proposed hybrid approach. Specifically, it captures similar RBF structures (to a standard RBF 380 network). However, it benefits from the additional flexibility the NN component provides, allowing for more nuanced function approximation in regions requiring a balance of smoothness and sharp 382 transitions. This is likely apparent in the smaller kernels learned by the hybrid block.



381

383

384

385

386

387

388

389

390

391

392

394

395

396

397

398

399

400

405

406

428

Figure 5: Allen-Cahn equation: Comparison of the mean relative L^2 error using various methods as a function of the number of hidden layers.

Stacking structure of residual blocks improve prediction ac**curacy.** We next show that the stacking structure employed by the HyResPINN in each hybrid residual block structure introduced here further helps to capture the smooth and sharp features in the Allen-Cahn solution. Figure 5 shows the predicted solutions between the proposed method and the competing baselines. The competing methods need help accurately capture the sharp transitions in the solutions. In contrast, our approach more accurately captures these sharp transitions due to the conjunction of the smooth neural network architecture and the learned nonsmooth RBF kernels.

Next, we evaluate the effect of network depth on predictive accuracy for different architectures. Figure 5 HyResPINN consistently achieves the lowest error across different network depths. The performance of other methods shows varying sensitivity to the number of hidden layers, with HyResPINN offering the most robust performance. Further, the left plot in Figure 6 shows that HyResPINN and PiratePINN achieve the lowest error over iter-

401 ations, with HyResPINN consistently reducing the error more effectively than the other methods. The right plot demonstrates that-although HyResPINN has a slightly higher training cost-it signif-402 icantly outperforms other methods in accuracy, particularly for long training times. These results 403 highlight the robustness and efficiency of HyResPINN in solving complex PDEs. 404

Remark While

attempting to replicate 407 the results of (Wang 408 et al., 2024a), us-409 the reported ing 410 hyperparameters 411 and implementation 412 details, we observed a slight performance 413 between the 414 gap results obtained in 415 implementation our 416 and those listed in 417 their paper. The 418 observed discrepancy 419 in the final model's 420 performance could



Figure 6: Allen-Cahn equation: Comparison of mean relative L^2 error across various methods, plotted against the number of training iterations (left) and the mean training time (right). Vertical bars in right plot indicate the time for which each method achieved an error of 10^{-2} .

421 stem from several factors such as subtle variations in the implementation environment—such as 422 hardware differences, software library versions, or random seed initialization. Despite the differences 423 between our replication results and those reported in (Wang et al., 2024a), the comparison between 424 our architecture and the replicated version of their approach remains fair, as both were tested under identical conditions using the same dataset, hyperparameters, and training setup (including hardware 425 and software environment). Our findings indicate that our architecture outperforms the replicated 426 version of their model in many cases, along with all other replicated baseline methods. 427

4.2 DARCY FLOW 429

430 In this section, we describe the Darcy Flow problem, an elliptic boundary value problem given by, 431

$$-\nabla \cdot \mu \nabla \phi = f \quad \text{in } \Omega \tag{15}$$

433 $\phi = u \quad \text{on } \Gamma_D$ 434 $\mathbf{n} \cdot \mu \nabla \phi = g \quad \text{on } \Gamma_N$

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.

In important applications such as porous media flow, heat transfer, and semiconductor devices, the flux $\mathbf{u} = -\mu \nabla \phi$ is the variable of primary interest. Specifically, during the fabrication of semiconductor devices, impurities are introduced into the silicon substrate to alter its electrical properties—a process known as doping. The diffusion of dopants can be described by equations that are similar in form to Darcy's law, where the dopant concentration gradient drives the diffusion process. In semiconductor doping, impurities are introduced into the substrate to change its electrical properties. This involves diffusion, where the concentration gradient drives the movement of dopant atoms into the semiconductor material. While Darcy's law governs fluid flow through porous media based on pressure gradients, the diffusion of dopants in semiconductors is guided by concentration gradients, analogous to how fluids move through porous materials in Darcy's law.



Figure 7: 2D and 3D Smooth Darcy Flow equation: Comparison of the mean relative L^2 errors across each baseline method for the 2D Darcy Flow problem (left) and the 3D Darcy flow problem (right) plotted against the number of training collocation points. Solid lines show the solution error, while dashed lines show the x-directional flux errors.

4.2.1 ANNULUS DOMAIN WITH SMOOTH COEFFICIENTS

We initially showcase the convergence rate of our method by employing smooth manufactured solutions. We consider two scenarios: one within an annular 2D domain, and another within a 3D domain formed by extruding the 2D geometry in the z-direction, resulting in a cylinder with a height of two. We present a convergence study analyzing the error rate as the number of points increases. The exact solutions are set as,

$$u(x,y) = \sin(x)\sin(y), \quad u(x,y,z) = \sin(x)\sin(y)\sin(z),$$
 (16)

for the 2D and 3D cases, respectively. By substituting these exact solutions into the model problem
we define the source term and the boundary data. We solve the discrete problem with either Dirichlet
or Neumann boundary conditions. Convergence results are shown in Figure 7 for the two and three
dimensional problems and demonstrate that HyResPINNs remain robust to boundary condition type,
size of training set (achieves lowest errors compared to the baselines for fewer training points), and
problem dimension.

477 478

432

438

439

440

441

442

443

444

445

446

447 448

449

450

451

452 453

454

455

456 457

458

459

460

461 462

463

469 470

4.2.2 2D BOX DOMAIN WITH ROUGH COEFFICIENTS

A defining feature of methods for solving the Darcy Flow equation 4.2, lies in their proficiency in accurately depicting the flux variable in scenarios where the coefficients μ are discontinuous. In these instances, the flux's normal component maintains continuity across material interfaces, whereas the tangential component may exhibit discontinuities. In contrast, collocated methods like Galerkin, stabilized Galerkin, and least-squares finite elements often fail to replicate this physical behavior, typically resulting in oscillations at the interface. In this section, we demonstrate that HyResPINNs successfully provides physically accurate flux approximations for problems characterized by discontinuous coefficients.

486 The initial example presented is the well-documented "five strip problem" Nakshatrala et al. (2006); 487 Masud & Hughes (2002), which serves as a conventional manufactured solution test to evaluate 488 a method's capability to preserve the continuity of normal flux. The prescribed exact solution on 489 domain $\Omega = [0, 1]^2$ is given by,

$$\phi_{ex} = 1 - x, \quad \text{and} \quad \Gamma_N = \Gamma, \tag{17}$$

such that Ω is divided in five equal strips,

$$\Omega_i = \{(x, y) \mid 0.2(i-1) \le y \le 0.2i; 0 \le x \le 1\}, \quad i = 1, ..., 5$$
(18)

with different μ_i on each Ω_i such that $\mu_1 = 16$, $\mu_2 = 6$, $\mu_3 = 1$, $\mu_4 = 10$, $\mu_5 = 2$. We solve the discrete problem with Neumann boundary conditions. Convergence results are shown in Figure 8 and show that HyResPINNs provide accurate solutions, albeit with larger training sizes.

498 499 500

523 524

527

531

532

533 534

537

490 491 492

493 494 495

496

497

5 CONCLUSION

501 In this work, we introduced HyResPINNs, a 502 novel class of physics-informed neural networks that incorporate adaptive hybrid residual blocks 504 combining the strengths of standard neural net-505 works and radial basis function (RBF) networks. 506 Our architecture effectively captures continuous and discontinuous features by leveraging 507 smooth and non-smooth function approximators. 508 The adaptive combination parameters within 509 each block allow the model to balance the con-510 tributions of neural and RBF components dur-511 ing training. Furthermore, including Wendland 512 kernels enhances the model's ability to handle 513 sharp transitions while maintaining computa-514 tional efficiency. Our experiments demonstrate 515 that HyResPINNs outperform traditional PINNs 516 and state-of-the-art methods in accuracy and ro-517 bustness, particularly for problems involving 518 mixed smooth and non-smooth regions. This work increases the flexibility and generalizabil-519 ity of PINNs by bridging the gap between clas-520 sical numerical methods and DNN-based ap-521 proaches for solving PDEs. 522



Figure 8: 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.

REFERENCES

- Jinshuai Bai, Gui-Rong Liu, Ashish Gupta, Laith Alzubaidi, Xi-Qiao Feng, and YuanTong Gu. 526 Physics-informed radial basis network (pirbn): A local approximating neural network for solving nonlinear partial differential equations. Computer Methods in Applied Mechanics 528 and Engineering, 415:116290, 2023. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma. 529 2023.116290. URL https://www.sciencedirect.com/science/article/pii/ 530 S0045782523004140.
 - Stefano Berrone, Claudio Canuto, and Moreno Pintore. Variational physics informed neural networks: the role of quadratures and test functions. Journal of Scientific Computing, 92(3):100, 2022.
 - Grigorios G Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Yannis Panagakis, Jiankang Deng, and Stefanos Zafeiriou. P-nets: Deep polynomial neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7325–7335, 2020.
- Grigorios G Chrysos, Markos Georgopoulos, Jiankang Deng, Jean Kossaifi, Yannis Panagakis, and 538 Anima Anandkumar. Augmenting deep classifiers with polynomial neural networks. In European Conference on Computer Vision, pp. 692-716. Springer, 2022.

540 541 542	Eric C Cyr, Mamikon A Gulian, Ravi G Patel, Mauro Perego, and Nathaniel A Trask. Robust training and initialization of deep neural networks: An adaptive basis viewpoint. In <i>Mathematical and Scientific Machine Learning</i> , pp. 512–536. PMLR, 2020.
543 544 545	Arka Daw, Jie Bu, Sifan Wang, Paris Perdikaris, and Anuj Karpatne. Rethinking the importance of sampling in physics-informed neural networks.
546 547 548 549	Suchuan Dong and Naxian Ni. A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. <i>Journal of Computational Physics</i> , 435: 110242, 2021.
550 551	Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative filter networks. In International Conference on Learning Representations, 2020.
552 553 554	Zhuojia Fu, Wenzhi Xu, and Shuainan Liu. Physics-informed kernel function neural networks for solving partial differential equations. <i>Neural Networks</i> , 172:106098, 2024.
555 556 557	Han Gao, Luning Sun, and Jian-Xun Wang. Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. <i>Journal of Computational Physics</i> , 428:110079, 2021.
558 559 560 561	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. <i>arXiv</i> preprint arXiv:2311.06483, 2023.
562 563	Xinquan Huang and Tariq Alkhalifah. Efficient physics-informed neural networks using hash encoding. <i>Journal of Computational Physics</i> , 501:112760, 2024.
565 566 567	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. <i>Communications in Computational Physics</i> , 28(5):2002–2041, 2020.
568 569 570	Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. <i>Proceedings of the Royal Society A</i> , 476(2239):20200334, 2020a.
572 573 574	Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. <i>Journal of Computational Physics</i> , 404:109136, 2020b.
575 576 577	Namgyu Kang, Byeonghyeon Lee, Youngjoon Hong, Seok-Bae Yun, and Eunbyung Park. Pixel: Physics-informed cell representations for fast and accurate pde solvers. In <i>Proceedings of the AAAI Conference on Artificial Intelligence</i> , volume 37, pp. 8186–8194, 2023.
578 579 580	Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. Variational physics-informed neural networks for solving partial differential equations. <i>arXiv preprint arXiv:1912.00873</i> , 2019.
581 582 583 584 585	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.
586 587 588	Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Char- acterizing possible failure modes in physics-informed neural networks. <i>Advances in Neural</i> <i>Information Processing Systems</i> , 34:26548–26560, 2021.
589 590 591	Randall J LeVeque. Finite difference methods for ordinary and partial differential equations: steady- state and time-dependent problems. SIAM, 2007.
592 593	Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. <i>arXiv preprint arXiv:2010.08895</i> , 2020.

594 595	Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. <i>Mathematical programming</i> , 45(1-3):503–528, 1989.	
597 598	Ziqi Liu, Wei Cai, and Zhi-Qin John Xu. Multi-scale deep neural network (mscalednn) for solving poisson-boltzmann equation in complex domains. <i>arXiv preprint arXiv:2007.11207</i> , 2020.	
599 600 601 602	Suryanarayana Maddu, Dominik Sturm, Christian L Müller, and Ivo F Sbalzarini. Inverse dirichlet weighting enables reliable training of physics informed neural networks. <i>Machine Learning: Science and Technology</i> , 3(1):015026, 2022.	
603 604	Arif Masud and Thomas JR Hughes. A stabilized mixed finite element method for darcy flow. <i>Computer methods in applied mechanics and engineering</i> , 191(39-40):4341–4370, 2002.	
605 606 607	Levi McClenny and Ulisses Braga-Neto. Self-adaptive physics-informed neural networks using soft attention mechanism. arXiv preprint arXiv:2009.04544, 2020.	
608 609	Levi D McClenny and Ulisses M Braga-Neto. Self-adaptive physics-informed neural networks. <i>Journal of Computational Physics</i> , 474:111722, 2023.	
610 611 612 613	Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Finite basis physics-informed neural networks (fbpinns): a scalable domain decomposition approach for solving differential equations. <i>Advances in Computational Mathematics</i> , 49(4):62, 2023.	
614 615 616	Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics- informed neural networks via importance sampling. <i>Computer-Aided Civil and Infrastructure</i> <i>Engineering</i> , 36(8):962–977, 2021.	
617 618 619 620	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. <i>Computer Methods in Applied Mechanics and Engineering</i> , 195(33-36):4036–4049, 2006.	
621 622 623	Ravi G Patel, Indu Manickam, Nathaniel A Trask, Mitchell A Wood, Myoungkyu Lee, Ignacio Tomas, and Eric C Cyr. Thermodynamically consistent physics-informed neural networks for hyperbolic systems. <i>Journal of Computational Physics</i> , 449:110754, 2022.	
624 625 626	Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In <i>International Conference on Machine Learning</i> , pp. 5301–5310. PMLR, 2019.	
627 628 629	Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. <i>The Journal of Machine Learning Research</i> , 19(1):932–955, 2018.	
630 631 632	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. <i>Journal of Computational physics</i> , 378:686–707, 2019.	
634 635 636 637 638	Gaétan Raynaud, Sébastien Houde, and Frédérick P. Gosselin. Modalpinn: An extension of physics- informed neural networks with enforced truncated fourier decomposition for periodic flow recon- struction using a limited number of imperfect sensors. <i>Journal of Computational Physics</i> , 464: 111271, 2022. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2022.111271. URL https: //www.sciencedirect.com/science/article/pii/S0021999122003333.	
639 640 641 642	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.), <i>Advances in Neural Information Processing Systems</i> , 2022. URL https://openreview.net/forum?id=NYpU9BRODos.	
643 644 645	Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. <i>Advances in neural information processing systems</i> , 33:7462–7473, 2020.	
646 647	Hwijae Son, Jin Woo Jang, Woo Jin Han, and Hyung Ju Hwang. Sobolev training for physics informed neural networks. <i>arXiv preprint arXiv:2101.08932</i> , 2021.	

648 649	Gilbert Strang, George J Fix, and DS Griffin. An analysis of the finite-element method. 1974.	
650	Matthew Tancik Pratul Srinivasan Ben Mildenhall Sara Fridovich-Keil Nithin Raghayan Utkarsh	
651	Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn	
652	high frequency functions in low dimensional domains. Advances in Neural Information Processing	
653	Systems, 33:7537–7547, 2020.	
654	Sifan Wang, Vujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow nathologies	
655	in physics-informed neural networks. SIAM Journal on Scientific Computing 43(5):A3055–A3081	
656	2021a.	
657		
658	Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks:	
659	<i>Methods in Applied Mechanics and Engineering</i> 384:113038, 2021b	
660	memous in reprice meeting and Engineering, 50 (115550, 20210.	
661	Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training	
662	physics-informed neural networks. <i>arXiv preprint arXiv:2203.07404</i> , 2022a. Sifan Wang, Xinling Xu, and Paris Perdikaris. When and why pinns fail to train: A poural tangant	
664	Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent	
665	kernel perspective. Journal of Computational Physics, 449:110768, 2022b.	
666	Sifan Wang, Shyam Sankaran, Hanwan Wang, and Daris Dardikaris. An avpart's guide to training	
667	physics-informed neural networks arXiv preprint arXiv:2308.08468, 2023	
668		
669	Sifan Wang, Bowen Li, Yuhan Chen, and Paris Perdikaris. Piratenets: Physics-informed deep learning	
670	with residual adaptive networks. arXiv preprint arXiv:2402.00326, 2024a.	
671	Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality for training physics-	
672	informed neural networks. Computer Methods in Applied Mechanics and Engineering, 421:	
673	116813, 2024b.	
674	Colby L Wight and Jia Zhao. Solving allen-cahn and cahn-hilliard equations using the adaptive	
675	physics informed neural networks. arXiv preprint arXiv:2007.04542, 2020.	
677	Chenyi Wu Min Zhu Oinyang Tan Yadhu Kartha and Lu Lu. A comprehensive study of non-	
678	adaptive and residual-based adaptive sampling for physics-informed neural networks. <i>Computer</i>	
679	Methods in Applied Mechanics and Engineering, 403:115671, 2023.	
680	Langua Va. La La Valui Mana and Casara Far Variadabia. Candient schemend abarias informed	
681	neural networks for forward and inverse nde problems. Computer Methods in Applied Mechanics	
682	and Engineering, 393:114823, 2022a.	
683		
684	Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed	
685	and Engineering 393:114823 2022b	
686		
682		
680		
690		
691		
692		
693		
694		
695		
696		
697		
698		
699		
700		
101		

702 A EXPERIMENTAL DESIGN

We use mini-batch gradient descent, where collocation points are randomly sampled during training iteration, for all Allen-Cahn experiments, and we use full-batch gradient descent (but varying training set size) for all other experiments. We use the Adam optimizer Kingma & Ba (2015), and follow the learning rate schedule of Wang et al. (2023) 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. Following the best practices described in Wang et al. (2023), we also employ a learning rate annealing algorithm Wang et al. (2023) to balance losses and causal training Wang et al. (2022a; 2023) to mitigate causality violation in solving time-dependent PDEs and apply exact periodic boundary conditions Dong & Ni (2021) when applicable. We use the hyperbolic tangent activation functions and initialize each network's parameters using the Glorot normal scheme, unless otherwise specified. We ran five random trials for each test, and report the mean values achieved in each plot and table.

716		
717	Parameter	Value
718	Architecture	
719	Number of layers	9
720	Number of channels	128
721	Activation	Tanh
722	Equition footune coole	2.0
723	Fourier leature scale	2.0
724	Random weight factorization	$\mu = 1.0, \sigma = 0.1$
725	Learning rate schedule	
726	Initial learning rate	10^{-3}
727	Decay rate	0.9
728	Decay steps	5×10^3
729	Wormann stons	$E \propto 10^3$
730	warmup steps	$3 \times 10^{\circ}$
731	Training	
732	Training steps	$3 imes 10^5$
733	6 1	
734	Weighting	
735	Weighting scheme	Gradient Norm Wang et al. (2022b; 2023)
736	Causal tolerance	1.0
737	Number of chunks	32
738		

Table 2: Hyper-parameter configurations for experiments.