AMORTIZED PHYSICS-INFORMED LEARNING VIA GENERATIVE INITIALIZATION OF RADIAL BASIS FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Physics-Informed Neural Networks (PINNs) commonly optimize a new solution model for each PDE instance, making repeated changes in parameters, boundary conditions, or forcing terms costly. GI-RBF addresses this limitation through a decoupled offline–online framework combining an explicit anisotropic Radial Basis Function (RBF) solver with a task-conditioned generative initializer. The initializer amortizes the search for the complete RBF state—including coefficients, centers, and kernel geometry—evaluated once per query; its output is then refined directly under the deployment physics objective. Closed-form derivatives and the localized explicit representation enable inexpensive physics updates, while learned initialization reduces the refinement needed to reach useful accuracy. Experiments on four canonical PDE families and a transient $(2+1)$-dimensional Flow Mixing problem show improved finite-budget adaptation over standard RBF initialization across interpolation and extrapolation regimes. The resulting inference-through-adaptation paradigm reaches an accurate solution through fast, physics-based refinement at deployment.

1 INTRODUCTION

Multi-query PDE problems arise when the same governing equation must be solved repeatedly for varying material properties, boundary conditions, or forcing terms. These settings are common in design optimization, uncertainty quantification, and inverse modeling, where repeated solves must be both accurate and fast (Hesthaven et al., 2015). High-fidelity discretization-based solvers such as the Finite Element Method (FEM) remain accurate, but their per-query cost becomes prohibitive when many parameter settings must be evaluated, since each new condition typically requires a new solve (Quarteroni et al., 2015).

Addressing these computational challenges has motivated the development of various strategies to accelerate the solution process. Reduced-order models and machine learning-based surrogates offer significant speed-ups compared to conventional PDE solvers (Lu et al., 2021; Padula et al., 2024). However, while effective, these methods often require large training datasets to generalize well, and surrogate models may struggle with extrapolation beyond the parameter regimes represented in the training data (Musekamp et al., 2024). As an alternative, PINNs eliminate the need for labeled data by embedding the governing equations directly into the loss function. However, they often suffer from slow convergence and require retraining for each new instance of the PDE, limiting their efficiency in multi-query scenarios (Ren et al., 2025).

A key insight is that parametrized PDE solutions often concentrate near a lower-dimensional solution manifold, enabling information learned from one set of configurations to guide adaptation to new configurations (Fresca & Manzoni, 2022; Benner et al., 2017). We exploit this by decomposing multi-query PDE solving into two components: a generative model that amortizes task-specific initializations across PDE conditions, and a lightweight physics-informed refinement stage that corrects each prediction at deployment. The variational initializer supports both a single posterior-mean prediction and sampling alternative RBF initializations for the same task. Unlike purely zero-shot predictors, this design keeps an online adaptive solver active, so new queries can be refined rather than accepted as fixed surrogate outputs.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The adaptive solver is represented with Radial Basis Functions (RBFs), which provide smooth, localized, explicit, and differentiable solution representations with closed-form derivatives. Their online capacity is directly interpretable through the kernel budget K and can be extended by adding or refining kernels, in contrast to dense black-box weights where local capacity changes are less transparent. GI-RBF predicts task-specific RBF parameters before online PINN refinement; offline training can use physics, data-driven, or hybrid objectives.

The main contributions are:

- We show that a structured anisotropic RBF representation with closed-form derivatives can replace the standard MLP inside a physics-informed objective, yielding an efficient, interpretable adaptive solver whose online capacity is controlled by the kernel budget K.
- We propose GI-RBF, a task-conditioned generative initializer that amortizes the complete RBF solver state and accelerates finite-budget physics-informed adaptation across parametric PDE families.
- We evaluate inference-through-adaptation as a deployment paradigm through matched online refinement curves against supervised zero-shot references (U-Net, FNO) and physics-informed learned baselines (Hyper-LR-PINN, PINO, P^{2}INN-SVD), a time-to-target analysis against non-amortized RBF initialization, explicit offline/online cost accounting, and a transient $(2+1)$-dimensional Flow Mixing problem.

2 Related Work

Reduced-Order and Meta-Learned Parametric Solvers. Classical reduced-order models amortize PDE families through a shared trial basis and solve only for reduced coordinates online (Hesthaven et al., 2015; Quarteroni et al., 2015; Benner et al., 2017; Fresca & Manzoni, 2022). Physics-informed counterparts reuse information across tasks in different ways: HyperPINN generates the weights of a small PINN; Hyper-LR-PINN factorizes a meta-trained backbone into fixed low-rank components and predicts only the coefficients that combine them per task; GPT-PINN combines selected pretrained PINNs as a reduced basis; and P^{2}INN conditions a shared PINN and adapts only a small set of SVD modulation variables over an otherwise fixed backbone (de Avila Belbute-Peres et al., 2021; Cho et al., 2023b; Chen & Koohy, 2024; Cho et al., 2024). These methods establish conditional weight generation, meta-learning, and restricted online adaptation, since task-specific updates are confined to a small coefficient subspace while the backbone itself stays fixed; DATS separately improves how meta-training tasks are sampled (Toloubidokhti et al., 2024). GI-RBF instead initializes the complete, low-dimensional state of an explicit RBF solver, whose coefficients, centers, scales, and orientations all remain freely trainable at deployment, without a fixed backbone or restricted subspace.

Physics-Informed Operator Refinement. PINO learns a neural solution operator with data and/or equation constraints and can fine-tune the learned operator ansatz on an individual test PDE using its equation loss (Li et al., 2024). It is therefore a direct precedent for family-level learning followed by physics-informed test-time adaptation, while PDE-Refiner improves neural PDE rollouts through repeated full-field denoising (Lippe et al., 2023). GI-RBF follows the same broad offline–online pattern, but the online update targets an explicit RBF kernel state rather than the operator network itself: whereas PINO fine-tunes the same network end-to-end, GI-RBF's generator and online solver are architecturally decoupled, so refinement never updates the generator's weights. Accordingly, PINO is evaluated as a physics-refined baseline; test-time correction itself is not claimed as a contribution.

RBF Collocation and Gaussian PDE Representations. RBF collocation has a long history as a meshfree PDE method, including Kansa-type domain collocation, localized variants, anisotropic kernels, and space–time constructions (Chen et al., 2014; Buhmann, 2003; Fasshauer, 2007; Fornberg & Flyer, 2015). Neither RBFs nor their analytic differentiation are new. Physics-Informed Gaussians more recently introduced adaptive Gaussian primitives with a lightweight prediction head for individual PDE instances (Kang et al., 2024). The narrower contribution of GI-RBF is the task-conditioned amortization of the complete anisotropic RBF state across a PDE family, followed by independent physics-based refinement of that explicit state at deployment.

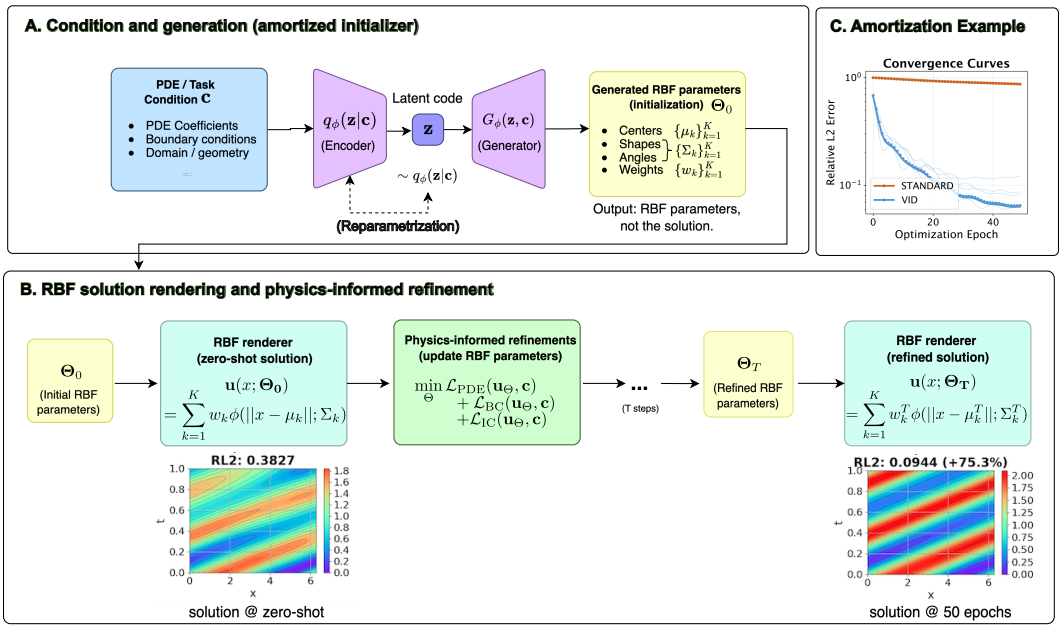

Figure 1: Overview of the proposed amortized physics-informed solver. Given a PDE condition c, a generative initializer produces RBF parameters $\mathbf{\Theta}_0$, which are refined through a small number of physics-informed updates to obtain $\mathbf{\Theta}_T$. Panel C conceptually illustrates improved optimization under a fixed budget.

3 Problem Setting

Consider a family of parametric partial differential equations (PDEs) defined over a spatial domain $\Omega \subset \mathbb{R}^d$ and time interval $\mathcal{T}$, indexed by a condition vector $\boldsymbol{c} \in \mathcal{C}$ that may encode PDE parameters, boundary conditions, initial conditions, or domain geometry[1]:

$$\begin{aligned} \mathcal{N}_{\boldsymbol{c}}[u](x,t) &= 0, \quad (x,t) \in \Omega \times \mathcal{T}, \\ \mathcal{B}_{\boldsymbol{c}}[u](x,t) &= 0, \quad (x,t) \in \partial\Omega \times \mathcal{T}, \\ \mathcal{I}_{\boldsymbol{c}}[u](x,0) &= 0, \quad x \in \Omega. \end{aligned} \tag{1}$$

Here, $\mathcal{N}_{\boldsymbol{c}}[u]$ is the PDE residual (e.g., $\mathcal{N}_{\boldsymbol{c}}[u] = u_t + \nabla\cdot(a_{\boldsymbol{c}}\nabla u) - f_{\boldsymbol{c}}$ for diffusion), while $\mathcal{B}_{\boldsymbol{c}}[u]$ and $\mathcal{I}_{\boldsymbol{c}}[u]$ enforce boundary and initial conditions. In the Physics-Informed Neural Network (PINN) framework, these residuals are minimized without relying on labeled solution data.

Building on this formulation, the problem can be cast in a task-based setting. Each task $i \in \mathcal{S}$ is associated with a condition vector $\boldsymbol{c}_i \in \mathcal{C}$ and a corresponding solution u_i of the PDE. A limited subset $\mathcal{S}_{\text{train}} \subset \mathcal{S}$ is available, with solutions sharing structural similarities. The objective is to train an auxiliary model that generates approximations for unseen tasks $i_{\text{test}} \in \mathcal{S} \setminus \mathcal{S}_{\text{train}}$, specified by their condition vectors $\boldsymbol{c}_{i_{\text{test}}}$, enabling an adaptive model to improve under a small wall-clock refinement budget rather than requiring a full per-instance solve. Offline training can use physics-based, data-driven, or hybrid objectives, compared in Appendix Table 20. Online refinement uses the governing equation and prescribed initial and boundary conditions, without interior solution labels for the new task.

4 Methods

Given a parametric PDE specified by a condition vector $\boldsymbol{c}$, the goal is to produce accurate solutions under physics-informed deployment: at test time, each prediction remains correctable by online PINN refinement rather than being used only as a fixed zero-shot surrogate. As illustrated in Fig. 1,

[1]The experiments reported here use a fixed spatial domain within each PDE family.

GI-RBF combines (i) an RBF-based PINN refined by physics-informed optimization at inference time and (ii) a task-conditioned generative initializer. The implementation maps $\boldsymbol{c}$ to a variational latent distribution over RBF parameters; posterior-mean deployment provides one reproducible starting point, while sampling can generate alternative candidates for the same task.

4.1 Adaptive Model

The solution u is represented using K anisotropic Gaussian RBFs over spatiotemporal coordinates $\boldsymbol{\xi} = [\boldsymbol{x}^\top, t]^\top \in \mathbb{R}^{d+1}$:

$$u(\boldsymbol{\xi}; \boldsymbol{\Theta}) = \sum_{k=1}^{K} w_k \, \phi_k(\boldsymbol{\xi}; \boldsymbol{\mu}_k, \Sigma_k), \tag{2}$$

$$\phi_k(\boldsymbol{\xi}; \boldsymbol{\mu}_k, \Sigma_k) = \exp\left(-\frac{1}{2} (\boldsymbol{\xi} - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\boldsymbol{\xi} - \boldsymbol{\mu}_k) \right), \tag{3}$$

Each basis function is parameterized by a center $\boldsymbol{\mu}_k$ and covariance Σ_k, which together control kernel location and anisotropic shape. With parameters $\boldsymbol{\Theta} = \{(w_k, \boldsymbol{\mu}_k, \Sigma_k)\}_{k=1}^K$, each covariance is parameterized as $\Sigma_k = \boldsymbol{Q}_k \, \mathrm{diag}(\sigma_{k,1}^2, \ldots, \sigma_{k,d+1}^2) \boldsymbol{Q}_k^\top$, where $\boldsymbol{Q}_k$ encodes kernel orientation (e.g., a 2D rotation matrix). This explicit RBF formulation enables analytic computation of derivatives for efficient PINN residual evaluation.

4.2 Adaptive Model Objective

The training objective combines residuals evaluated at collocation points $\mathcal{X}_r$, boundary points $\mathcal{X}_b$, and initial points $\mathcal{X}_0$, together with optional observational data $\mathcal{D}$. The formulation is presented for a time-dependent PDE with spatiotemporal coordinates (x, t), but extends directly to steady-state problems or higher-dimensional settings. This objective is referred to as the *task loss* because, depending on data availability, it can be used in a physics-only (PINN) regime, a data-driven regime, or a hybrid regime combining both.

$$\begin{aligned}
\mathcal{L}_{\text{task}}(\boldsymbol{\Theta}; \boldsymbol{c}) = \alpha_r \, \mathbb{E}_{(x,t)\in\mathcal{X}_r} \Big\| \mathcal{N}_{\boldsymbol{c}}[u(\cdot;\boldsymbol{\Theta})]\,(x,t) \Big\|_2^2 \\
+ \alpha_b \, \mathbb{E}_{(x,t)\in\mathcal{X}_b} \Big\| \mathcal{B}_{\boldsymbol{c}}[u(\cdot;\boldsymbol{\Theta})]\,(x,t) \Big\|_2^2 \\
+ \alpha_0 \, \mathbb{E}_{(x,t)\in\mathcal{X}_0} \Big\| u(x,t;\boldsymbol{\Theta}) - u_0(x) \Big\|_2^2 \\
+ \alpha_y \, \mathbb{E}_{(x,t,y)\in\mathcal{D}} \Big\| u(x,t;\boldsymbol{\Theta}) - y \Big\|_2^2 .
\end{aligned} \tag{4}$$

When no observational data are available ($\alpha_y = 0$), $\mathcal{L}_{\text{task}}$ reduces to a standard data-free PINN objective; when physics terms are omitted, it reduces to the purely supervised setting.

4.3 Generative Initialization

While the RBF model in (3) can fit a single PDE instance, training a large number of instances becomes time-consuming, particularly when solving families of related PDEs under varying conditions $\boldsymbol{c}$. To mitigate this cost, a *generative initializer* G_ϕ is introduced to predict a suitable starting point for each PDE instance conditioned on $\boldsymbol{c}$. This approach amortizes the search for effective kernel locations, scales, and orientations across the training distribution of PDEs. The initializer is cast as a variational latent-variable model rather than a deterministic conditional map so that a single trained network supports posterior-mean deployment, stochastic sampling, and multi-candidate sets, without requiring separate models for each use case.

Formally, a latent-variable model is defined as

$$\boldsymbol{z} \sim q_\phi(\boldsymbol{z} \mid \boldsymbol{c}) = \mathcal{N}\big(\boldsymbol{\mu}_\phi(\boldsymbol{c}), \mathrm{diag}\big(\boldsymbol{\sigma}_\phi^2(\boldsymbol{c})\big)\big), \quad \boldsymbol{\Theta}_{\text{ZS}} = G_\phi(\boldsymbol{z}; \boldsymbol{c}). \tag{5}$$

where the encoder $(\boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi)$ maps the PDE condition $\boldsymbol{c}$ to a compact latent representation $\boldsymbol{z}$, and the decoder G_ϕ predicts a full set of RBF parameters $\boldsymbol{\Theta}_{\text{ZS}} = \{(w_k, \boldsymbol{\mu}_k, \Sigma_k)\}_{k=1}^K$. Each covariance

matrix Σ_k is parameterized via log-scales and an orientation angle. At test time, GI-RBF receives only the PDE condition $\boldsymbol{c}$, not the target solution or observed measurements. Sampling $\boldsymbol{z}$ therefore produces alternative RBF initializations for the same PDE instance; the final solution is obtained by subsequent physics-informed refinement.

The mapping G_ϕ is implemented as a permutation-invariant Set Transformer (Lee et al., 2019) that maps $(\boldsymbol{z}, \boldsymbol{c})$ to an unordered set of kernel parameters. This enables flexible allocation of kernels in space–time according to the PDE structure implied by $\boldsymbol{c}$. The resulting initialization encodes prior information about the expected solution structure, such as the presence of localized features or high-frequency regions, thereby reducing the number of subsequent physics-informed optimization steps required.

4.4 Refinement-Informed Training

Training uses a short physics-refinement trajectory to construct a solver-aligned target for the initializer. For each PDE condition $\boldsymbol{c}$, the inner loop adapts the generated RBF parameters $\boldsymbol{\Theta}$, while the outer update encourages the initializer to produce parameters that have low task loss before refinement and remain close to the short-refined target. The refined parameters are treated as a stop-gradient teacher; gradients are not propagated through the refinement trajectory.

Concretely, for each sampled condition $\boldsymbol{c}$, a latent code $\boldsymbol{z}$ is drawn from the conditional distribution $q_\phi(\boldsymbol{z} \mid \boldsymbol{c})$, and an initialization $\boldsymbol{\Theta}_{\mathrm{ZS}} = G_\phi(\boldsymbol{z}; \boldsymbol{c})$ is produced by the generative initializer. Starting from this initialization, T steps of inner optimization are performed:

$$\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t - \eta \, \nabla_{\boldsymbol{\Theta}} \mathcal{L}_{\text{task}}(\boldsymbol{\Theta}_t; \boldsymbol{c}), \qquad t = 0, \ldots, T-1, \tag{6}$$

yielding adapted parameters $\boldsymbol{\Theta}_{\mathrm{FT}} = \boldsymbol{\Theta}_T$.

The parameters ϕ of the generative initializer are optimized by minimizing

$$\begin{aligned} \mathcal{L}(\phi) = \alpha \, \mathcal{L}_{\text{task}}(\boldsymbol{\Theta}_{\mathrm{ZS}}; \boldsymbol{c}) + \gamma \, D_{\mathrm{KL}}(q_\phi(\boldsymbol{z} \mid \boldsymbol{c}) \,\|\, \mathcal{N}(\mathbf{0}, \mathbf{I})) \\ + \beta_{\text{eff}} \, \mathcal{L}_{\text{recon}}(\boldsymbol{\Theta}_{\mathrm{ZS}}, \boldsymbol{\Theta}_{\mathrm{FT}}) \, , \end{aligned} \tag{7}$$

The first term encourages low task loss before adaptation, the KL term regularizes the latent space, and the reconstruction term distills the short inner-loop refinement into the initializer. Concretely, it matches each generated kernel to its inner-refined counterpart in parameter space,

$$\mathcal{L}_{\mathrm{recon}}(\boldsymbol{\Theta}_{\mathrm{ZS}}, \boldsymbol{\Theta}_{\mathrm{FT}}) = \frac{1}{K} \sum_{k=1}^{K} \| \boldsymbol{\Theta}_{\mathrm{ZS},k} - \mathrm{sg}(\boldsymbol{\Theta}_{\mathrm{FT},k}) \|_2^2 \, , \tag{8}$$

Here, $\boldsymbol{\Theta}_{\cdot,k}$ denotes the decoder-ordered parameters of kernel k, and sg stops gradients through the inner-refined target. Matching is performed in decoder order, without separate permutation matching. During inner optimization, parameters are projected to valid ranges proportional to the PDE domain size. The adaptive schedule for β_{eff} and the full inner/outer-loop procedure are detailed in Algorithms 1–2. Additional regularizers (e.g., kernel repulsion and zero-shot self-bias) and training schedules used in the experiments are summarized in Appendix A.2. Appendix D.1 compares direct outer-task training with zero teacher steps against the 25-step refinement-informed teacher, separating useful conditional initialization from additional alignment with the deployment optimizer.

During inference, the trained generative initializer produces an initialization $\boldsymbol{\Theta}_{\mathrm{ZS}} = G_\phi(\boldsymbol{z}; \boldsymbol{c})$, followed by online refinement on (4). The refinement budget is a deployment choice; our evaluation reports 50, 500, and 1000 RBF optimization steps to measure how quickly useful accuracy is reached. The latent code can be drawn from the encoder distribution $q_\phi(\boldsymbol{z} \mid \boldsymbol{c})$ or from the prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$, depending on whether the encoder is used at deployment. The resulting pipeline performs amortized physics-informed inference by combining a learned initializer with online refinement under the task loss.

5 Experiments

Overview. Experiment I evaluates RBFs as the adaptive solver under physics-informed and supervised objectives. Experiment II tests whether learned initialization reduces online refinement across interpolation and extrapolation tasks, including time-to-target, offline cost, and a transient $(2+1)$-dimensional Flow Mixing extension. Supporting ablations are in Appendix D; PDE definitions are in Appendix E.

Datasets and PDE Tasks. Evaluation uses four PDE families that span nonlinear reaction–diffusion, shock-forming transport, linear advection, and steady-state wave equations: Allen–Cahn, Burgers, Convection, and Helmholtz 2D. All datasets are evaluated on a uniform 64×64 grid. Two dataset modalities are used: a standard random split and a parametric split for few-shot generalization. For the parametric modality, interpolation and extrapolation splits are defined by solution-space distance (PCA-based), yielding a clear difficulty gap of roughly $1.9\times$–$5.6\times$ (Table 24). Test indices are fixed across all models and seeds for fair comparisons.

Baselines. Experiment I compares physics-based solvers trained from scratch: RBF, PINN (MLP) (Raissi et al., 2019), SPINN (Cho et al., 2023a), and PIG (Kang et al., 2024), under identical adaptation budgets. Experiment II groups baselines by deployment: supervised zero-shot references (FNO (Li et al., 2021), U-Net; PDEArena benchmarks), physics-refined solvers from scratch (RBF, PINN), and amortized or physics-informed learned baselines (GI-RBF, Hyper-LR-PINN (Cho et al., 2023b), PINO (Li et al., 2024), and P^{2}INN-SVD (Cho et al., 2024)).

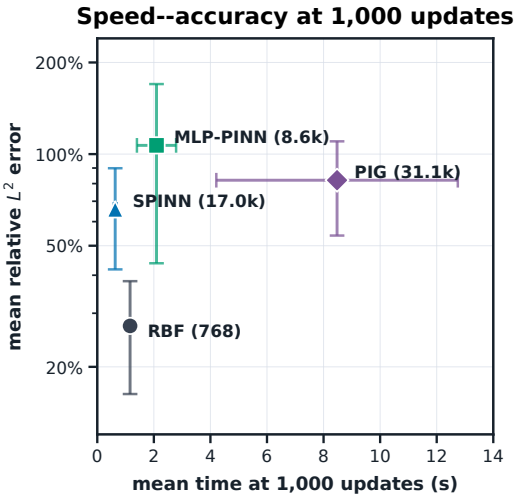

Figure 2: Speed–accuracy after 1,000 physics updates (mean $\pm$ s.d. across four equation-level means; labels show parameter counts). Full results: Table 5.

5.1 Experiment I: Choice of Adaptive Model (Why do we use RBFs?)

We compare RBF, SPINN, PIG, and an MLP PINN under the PINN loss and supervised data fitting, separating physics-objective optimization from representation capacity. Figure 2 shows that RBF achieves the lowest mean PINN error at 1,000 updates with fewer trainable parameters and competitive wall-clock time. Appendix B reports the 50/250/1,000-update results, supervised fits, configurations, initialization, and kernel-capacity audit. The supervised–PINN gap motivates learned initialization as an optimization aid.

5.2 Experiment II: Parametric Setting and Generative Initialization

Experiment II evaluates GI-RBF as an amortized initializer followed by physics-informed RBF refinement, not as a purely zero-shot neural operator. We use the parametric few-shot split ($n_{\text{train}} = 10$), evaluate interpolation/extrapolation regimes, and compare against a non-amortized RBF initialization with the same kernel budget, online optimizer, and refinement steps. Unless stated otherwise, benchmark evaluations use posterior-mean latent decoding for reproducibility. FNO and U-Net are supervised zero-shot reference baselines rather than equal-objective baselines. The reported GI-RBF checkpoints use full-field solution supervision for the offline outer task term and parameter-space targets produced by a short physics-only teacher refinement. At deployment, the target field is unavailable and online adaptation uses only the PDE objective. Appendix D.3 tests whether the stochastic and KL components are necessary for the primary single-start acceleration; neither is required, while KL has a bounded benefit for a four-candidate set on Allen–Cahn.

To evaluate amortization directly, Figure 3 reports online adaptation curves under matched conditions: the same kernel budget K, online optimizer, refinement objective, and adaptation steps, with only the initialization differing between GI-RBF and the non-amortized RBF baseline. It reports zero-shot RL2 and RL2 after 50, 500, and 1000 refinement steps, showing whether the learned initializer reduces the adaptation budget under the same RBF representation and optimizer. Representative 50-step solution and error maps for this comparison are shown in Appendix Figure 9. Because this controlled comparison holds the solver fixed, it isolates the benefit of amortized initialization rather than the lowest error attainable by the framework; a strengthened solver setting is considered separately below.

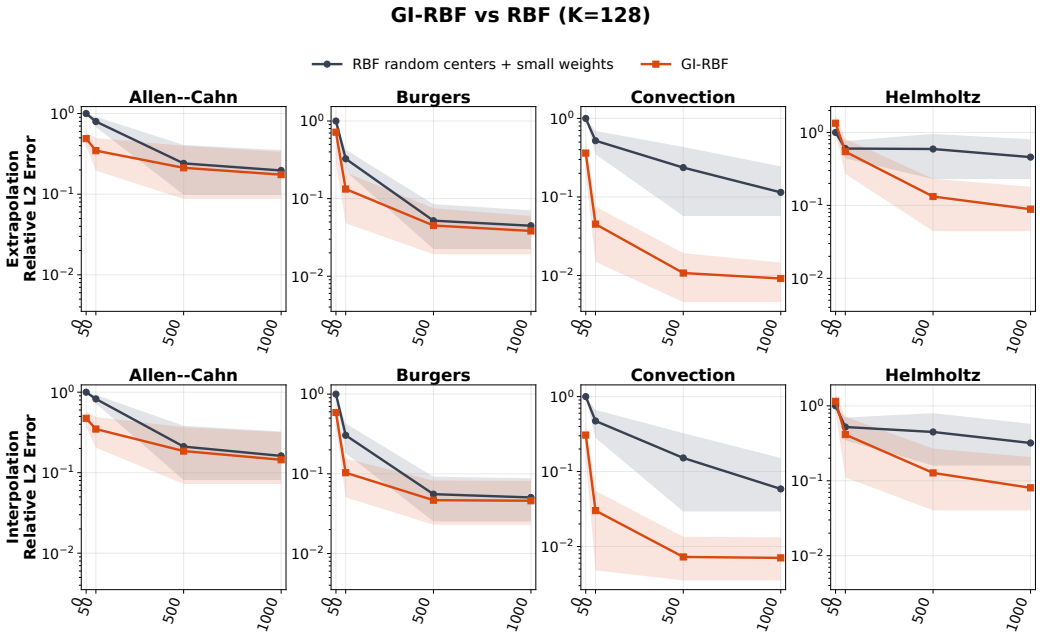

Figure 3: Online physics-informed adaptation of GI-RBF compared with the non-amortized RBF initialization baseline at matched refinement budgets.

To distinguish genuinely faster convergence from methods that merely solve fewer cases, Table 1 complements the refinement curves with a common 5% target, reporting conditional crossing time together with reach.

Table 1: Pooled 5% target attainment within 1,000 updates for the selected $K = 256$ operating points, with conditional median update/time among reached records. $\approx$ denotes a transferred same-equation, same-capacity clock; full timing provenance and split-level results are in Appendix C.2.

Equation	Std-RBF reach	GI reach	Std-RBF updates/time	GI updates/time
Allen–Cahn	100.0%	100.0%	500 / 0.431 s	50 / $\approx$0.073 s
Burgers	81.7%	94.2%	400 / $\approx$0.284 s	150 / $\approx$0.109 s
Convection	70.8%	95.0%	400 / 0.266 s	150 / 0.096 s
Helmholtz	60.8%	85.8%	275 / $\approx$0.205 s	175 / $\approx$0.124 s

Figure 4 compares GI-RBF-128 interpolation trajectories with physics-informed learned baselines and supervised zero-shot references. These results use the original refinement settings; Table 2 separately tests stronger solvers. Tables 11–12 give all interpolation/extrapolation values for $K = 128, 256$, while Appendix C.2 reports offline costs and repeated-query accounting.

To test whether this initialization benefit holds when the underlying solver is strengthened rather than held fixed, Table 2 reports a secondary sensitivity view for Allen–Cahn and Helmholtz, the two cases where a strengthened physics solver lowered the error of both the standard and amortized operating points — since GI-RBF only initializes rather than replaces the online solver, its finite-budget accuracy remains coupled to that solver's capability.

Within both strengthened operating points, GI-RBF preserves the benefit of a learned initialization while inheriting the lower-error online solver. This supports a modular interpretation: improvements to the underlying physics optimizer can improve amortized deployment as well as standard per-instance refinement.

GI-RBF has a one-time pretraining cost, but generator inference is sub-millisecond and negligible compared with online refinement. For the original $K = 128$ configuration, training takes 25.12 ± 0.08

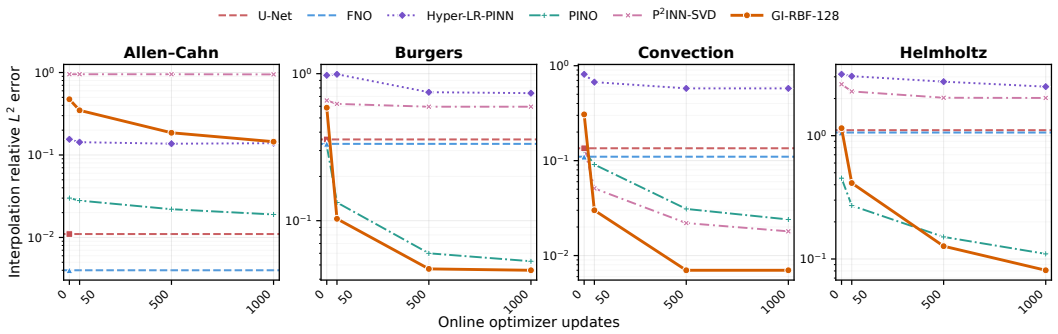

Figure 4: GI-RBF-128 interpolation trajectories vs. physics-informed learned baselines (Hyper-LR-PINN, PINO, P^{2}INN-SVD) and supervised zero-shot references (U-Net, FNO). Variability bands are omitted since aggregation differs across baseline families; full values are in Tables 11–12.

Table 2: Solver sensitivity at $K = 256$: interpolation/extrapolation mean relative-L^2 error (%, not a decimal fraction) at 1,000 updates for each full-domain solve; the strengthened Allen–Cahn solver uses 1,000 updates per window (3,000 total). The strengthened setting uses windowed hard-constraint refinement (Allen–Cahn) and gap-gated projected refinement (Helmholtz), each with its solver-aligned GI checkpoint. Appendix A.4 describes both settings.

Equation	**Solver setting**	**Standard-RBF I/E RL2 (%)**	**GI-RBF I/E RL2 (%)**
Allen–Cahn	Original physics objective	14.8/14.0	15.2/13.5
	Strengthened solver	0.60/0.99	0.37/0.43
Helmholtz 2D	Original physics objective	21.0/34.0	4.6/9.8
	Strengthened solver	6.00/7.51	2.95/4.94

minutes, one generator pass takes 0.382 ± 0.003 ms, and 1,000 RBF refinement updates take 2.374 ± 0.010 s. Thus, the claim is amortized optimization in multi-query settings rather than a free end-to-end speedup; the capacity breakdown and representative crossovers are reported in Appendix C.2. The crossover examples use a separate reduced-cost $K = 256$ configuration trained for 3,000 epochs with scheduled teacher masking, designed to reduce pretraining cost and the number of queries needed to amortize it. Because the repeated-query crossover point depends on this offline pretraining cost, we tested whether that cost itself can be reduced. Within a later 3,000-outer-update training configuration used for an offline-masking audit, randomly masking half of the decoded kernel rows during the short-refinement teacher reduced median warmed-loop time from 197.74s to 153.60s (22.3%) on Allen–Cahn and from 453.33s to 324.63s (28.4%) on Burgers, reducing the offline contribution to the crossover calculation. Because this configuration differs from the original 30k model and its accuracy effects are equation-dependent, the result is an offline cost–quality trade-off rather than a controlled schedule-only or universally lossless acceleration; Appendix D.2 gives the full qualification.

Extension to the transient $(2 + 1)$-dimensional Flow Mixing problem. To test whether the initialization benefit extends beyond the canonical benchmarks, GI-RBF is additionally evaluated on the transient Flow Mixing problem defined in Appendix E.3. Across five fixed-geometry test cases, Standard-RBF requires a median 250 updates in each of three initialization seeds to reach 10% relative-L^2 error, whereas posterior-mean GI-RBF requires 50 updates for every training seed. At 50 updates, the corresponding aggregate errors are $16.138 \pm 0.107\%$ for Standard-RBF and $7.851 \pm 0.277\%$ for GI-RBF; at 5,000 updates, they decrease to $4.399 \pm 0.003\%$ and $3.245 \pm 0.173\%$, respectively. Full trajectories are reported in Appendix C.1. The scope is limited to five cases with fixed geometry and one varying transport parameter; it does not establish general high-dimensional scaling.

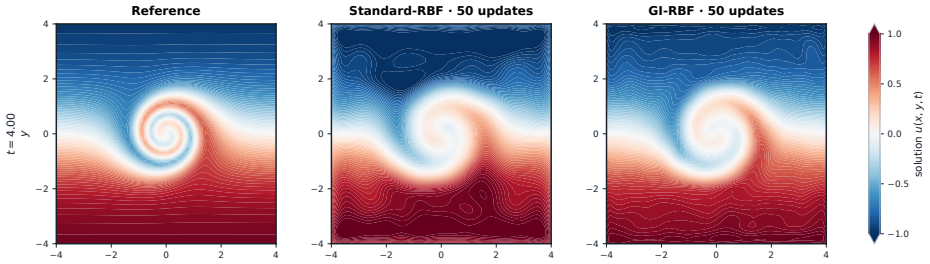

Figure 5: Representative held-out Flow Mixing fields at $t = 4$ after 50 matched updates. The GI-RBF operating point used for this visualization has five-case error numerically similar to the reported posterior-mean deployment ($7.835 \pm 0.318\%$ versus $7.851 \pm 0.277\%$).

6 Discussion

GI-RBF separates task-conditioned initialization from online physics refinement. Its benefits persist under the stronger solver settings in Table 2, where both Standard-RBF and GI-RBF improve, supporting their use as complementary components.

The framework is most useful when many related PDE instances must be solved and each query permits a small refinement budget. Its practical value depends jointly on the accuracy reached within that budget and the number of queries over which offline training is amortized.

Limitations. The observed gains depend on the PDE family, kernel budget, training data, and refinement objective. The reported initializers use offline solution supervision, and their training cost is recovered only after sufficiently many queries. Cross-method timings are approximate and use different timing boundaries. The Flow Mixing extension covers five fixed-geometry cases with one varying parameter and does not establish general high-dimensional or geometric generalization. A fixed kernel budget can limit localized solution features. Reduced training and teacher masking alter several settings and can affect accuracy; their savings are not universally lossless. Variational sampling is not required for the observed single-start gains, and sample dispersion is not calibrated uncertainty or a validated stopping criterion.

Future work. The initialization–refinement design could also be applied to other differentiable bases, such as Fourier bases, by adapting the parameter decoder and residual evaluation. RBF locality offers a route to spatially targeted refinement: future work could selectively update or insert kernels in regions with large physics residuals. Such local solution edits would need to preserve boundary conditions and consistency with the surrounding solution. Separating learned initialization from numerical refinement also offers a route to incorporating established stability and convergence guarantees for the underlying solver, provided their assumptions are satisfied. Establishing such guarantees for the adaptive RBF formulation remains future work.

6.1 Conclusion

GI-RBF combines an explicit anisotropic RBF solver with a learned initializer to accelerate physics-informed adaptation across parametric PDEs. The experiments demonstrate useful accuracy at finite online budgets and compatibility with stronger refinement methods, while separating the benefit of initialization from its offline cost.

AI USE STATEMENT

In this work, LLMs were used to help implement our method, write scripts to visualize experimental results, and enhance the readability of the text. LLMs were not used to develop theoretical models, conceptual frameworks, or mathematical claims; to propose or refine hypotheses; to design or provide feedback on research methodology or experiments; to conduct data analysis; or to interpret results. No synthetic datasets were used (all datasets were obtained by solving the governing PDEs), and no LLM-assisted data cleaning or reformatting was performed. The remaining disclosure tasks of assisting with mathematical proofs and language translation are not applicable to this work. The authors reviewed all LLM-assisted code and text and take full responsibility for the final content of this work.

ETHICS STATEMENT

This paper presents work whose goal is to advance the field of Machine Learning. The experiments use synthetic/numerical PDE benchmarks and do not involve human subjects or private data; potential risks mainly concern misuse of faster PDE surrogates without validation in safety-critical scientific or engineering workflows.

REFERENCES

Peter Benner, Mario Ohlberger, Albert Cohen, and Karen Willcox. *Model reduction and approximation: theory and algorithms*. SIAM, 2017.

Johannes Brandstetter, Rianne van den Berg, Max Welling, and Jayesh K Gupta. Clifford neural layers for pde modeling. *arXiv preprint arXiv:2209.04934*, 2022.

Martin D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, 2003.

Wen Chen, Zhuo-Jia Fu, and Ching-Shyang Chen. *Recent advances in radial basis function collocation methods*, volume 768. Springer, 2014.

Yanlai Chen and Shawn Koohy. GPT-PINN: Generative pre-trained physics-informed neural networks toward non-intrusive meta-learning of parametric PDEs. *Finite Elements in Analysis and Design*, 228:104047, 2024.

Junwoo Cho, Seungtae Nam, Hyunmo Yang, Seok-Bae Yun, Youngjoon Hong, and Eunbyung Park. Separable physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36:23761–23788, 2023a.

Woojin Cho, Kookjin Lee, Donsub Rim, and Noseong Park. Hypernetwork-based meta-learning for low-rank physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36:11219–11231, 2023b.

Woojin Cho, Minju Jo, Haksoo Lim, Kookjin Lee, Dongeun Lee, Sanghyun Hong, and Noseong Park. Parameterized physics-informed neural networks for parameterized pdes, 2024. URL https://arxiv.org/abs/2408.09446.

Filipe de Avila Belbute-Peres, Yi-fan Chen, and Fei Sha. HyperPINN: Learning parameterized differential equations with physics-informed hypernetworks. In *The symbiosis of deep learning and differential equations*, 2021.

Gregory E. Fasshauer. *Meshfree Approximation Methods with MATLAB*. World Scientific, 2007.

Bengt Fornberg and Natasha Flyer. *A primer on radial basis functions with applications to the geosciences*. SIAM, 2015.

Stefania Fresca and Andrea Manzoni. POD-DL-ROM: Enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering*, 388:114181, 2022.

Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

Jan S Hesthaven, Gianluigi Rozza, and Benjamin Stamm. *Certified reduced basis methods for parametrized partial differential equations*. SpringerBriefs in mathematics. Springer International Publishing, Cham, Switzerland, 1 edition, August 2015.

Namgyu Kang, Jaemin Oh, Youngjoon Hong, and Eunbyung Park. PIG: Physics-Informed Gaussians as adaptive parametric mesh representations. *arXiv preprint arXiv:2412.05994*, 2024. URL http://arxiv.org/abs/2412.05994.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning (ICML)*, pp. 3744–3753, 2019.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021. URL https://arxiv.org/abs/2010.08895.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024.

Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. PDE-Refiner: Achieving accurate long rollouts with neural PDE solvers. *Advances in Neural Information Processing Systems*, 36:67398–67433, 2023.

Kuan Lu, Kangyu Zhang, Haopeng Zhang, Xiaohui Gu, Yulin Jin, Shibo Zhao, Chao Fu, and Yongfeng Yang. A review of model order reduction methods for large-scale structure systems. *Shock and Vibration*, 2021(1):6631180, 2021.

Daniel Musekamp, Marimuthu Kalimuthu, David Holzmüller, Makoto Takamoto, and Mathias Niepert. Active learning for neural pde solvers. *arXiv preprint arXiv:2408.01536*, 2024.

Guglielmo Padula, Michele Girfoglio, and Gianlugi Rozza. A brief review of reduced order models using intrusive and non-intrusive techniques. *PAMM*, 24(4):e202400210, 2024.

Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced basis methods for partial differential equations: An introduction*. UNITEXT. Springer International Publishing, Cham, Switzerland, 1 edition, July 2015.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Zhiyuan Ren, Shijie Zhou, Dong Liu, and Qihe Liu. Physics-informed neural networks: A review of methodological evolution, theoretical foundations, and interdisciplinary frontiers toward next-generation scientific computing. *Applied Sciences*, 15(14):8092, 2025.

Maryam Toloubidokhti, Yubo Ye, Ryan Missel, Xiajun Jiang, Nilesh Kumar, Ruby Shrestha, and Linwei Wang. DATS: Difficulty-aware task sampler for meta-learning physics-informed neural networks. In *International Conference on Learning Representations*, volume 2024, pp. 19897–19917, 2024.

A Experimental Details

This section collects implementation details for the RBF adaptive model, GI-RBF initializer, and refinement-informed training. Dataset modalities and PDE definitions are summarized separately in Appendix E.

A.1 Adaptive Model

A.1.1 Parametrization of Gaussian Kernels

This section details the parameterization used in Eq. (3) for the anisotropic covariance matrices Σ_k, providing a practical implementation of the general form $\Sigma_k = \boldsymbol{Q}_k \operatorname{diag}(\sigma_{k,1}^2, \ldots, \sigma_{k,d+1}^2)\, \boldsymbol{Q}_k^\top$ introduced in the main text.

Each Gaussian basis function is parameterized by its center $\boldsymbol{\mu}_k \in \mathbb{R}^2$, log-scaled anisotropic standard deviations $\log \boldsymbol{\sigma}_k \in \mathbb{R}^2$, and rotation angle $\theta_k \in [0, 2\pi)$. The anisotropic standard deviations $\boldsymbol{\sigma}_k$ correspond to the shape parameters collected in $\boldsymbol{\lambda}$. Together with the kernel weights w_k, all trainable quantities are collected in the parameter set $\boldsymbol{\Theta}$.

The covariance matrix of the k-th kernel is expressed as

$$\Sigma_k = \boldsymbol{R}(\theta_k) \begin{bmatrix} \sigma_{k,1}^2 & 0 \\ 0 & \sigma_{k,2}^2 \end{bmatrix} \boldsymbol{R}(\theta_k)^\top,$$

where $\boldsymbol{R}(\theta_k)$ is the two-dimensional rotation matrix

$$\boldsymbol{R}(\theta_k) = \begin{bmatrix} \cos\theta_k & -\sin\theta_k \\ \sin\theta_k & \cos\theta_k \end{bmatrix}.$$

For numerical stability, the inverse covariance Σ_k^{-1} is precomputed and reused in both evaluation and derivative routines. A small diagonal term $\epsilon \boldsymbol{I}$ is added during inversion to ensure robustness even when $\sigma_{k,i}$ approaches zero. The weighted Gaussian kernel then takes the form

$$\phi_k(x) = w_k \exp\!\big(-\tfrac{1}{2}(x - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (x - \boldsymbol{\mu}_k)\big)\,.$$

This parameterization is compact and expressive: $\boldsymbol{\mu}_k$ controls kernel location, $\sigma_{k,1}$ and $\sigma_{k,2}$ define anisotropic scaling, θ_k accounts for orientation, and the log-parameterization guarantees positivity of the widths.

Although presented for the two-dimensional case, the formulation extends naturally to higher dimensions by replacing the 2×2 rotation with an orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{d \times d}$, and expressing the covariance as

$$\Sigma_k = \boldsymbol{Q} \operatorname{diag}(\sigma_{k,1}^2, \ldots, \sigma_{k,d}^2)\, \boldsymbol{Q}^\top.$$

For the $(2+1)$-dimensional Flow Mixing experiment, the decoder outputs four quaternion components per kernel, normalizes them to unit norm, and converts the resulting quaternion (scalar-first convention) to a 3×3 rotation matrix acting on normalized (x, y, t) coordinates; quaternions are renormalized during refinement. This generalization preserves the same decomposition—center, anisotropic scaling, and orientation—while maintaining efficient reuse of precomputed inverse covariance matrices during evaluation and differentiation. Thus, this formulation is fully consistent with the anisotropic RBF representation defined in the main text, differing only in its explicit parameterization for implementation and numerical stability.

A.1.2 Analytical Derivatives

The derivative formulation is illustrated in the two-dimensional setting. Given evaluation points $\boldsymbol{X} \in \mathbb{R}^{N \times 2}$ and a collection of anisotropic Gaussian kernels

$$\phi_k(\boldsymbol{x}) = \exp\!\big(-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)\big)\,, \quad k = 1, \ldots, K,$$

with weights w_k, the function of interest is

$$u(\boldsymbol{x}) = \sum_{k=1}^{K} w_k\, \phi_k(\boldsymbol{x}).$$

The gradients and Hessians admit closed-form expressions reusable across PDE operators:

$$\nabla\phi_k(\boldsymbol{x}) = -\phi_k(\boldsymbol{x})\,\Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k), \qquad \boldsymbol{H}[\phi_k](\boldsymbol{x}) = \phi_k(\boldsymbol{x})\big[\big(\Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\big)\otimes\big(\Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\big) - \Sigma_k^{-1}\big].$$

Accordingly,

$$\nabla u(\boldsymbol{x}) = \sum_{k=1}^{K} w_k\,\nabla\phi_k(\boldsymbol{x}), \qquad \boldsymbol{H}[u](\boldsymbol{x}) = \sum_{k=1}^{K} w_k\,\boldsymbol{H}[\phi_k](\boldsymbol{x}).$$

This structure yields direct access to u, u_x, u_y, u_{xx}, u_{yy}, and u_{xy} with a single batched computation. Higher-order derivatives can be incorporated conditionally, without altering the core formulation. Because the expressions are derived once and implemented in a vectorized form, they become reusable components for arbitrary PDEs requiring differential operators such as Laplacians or advection terms. The modular design therefore separates kernel precomputation, evaluation, and differentiation, ensuring that the same routines can be employed across tasks and domains.

A.1.3 Regularization Strategies

Parameter Initialization. The initialization of RBF parameters is critical for stable and efficient training of PINNs. For two-dimensional problems, the RBF centers $\boldsymbol{\mu}$ are sampled from a coarse meshgrid spanning the spatial and temporal domain, providing well-distributed coverage of the input space. The shape parameters $\boldsymbol{\lambda}$, represented by the anisotropic widths $\boldsymbol{\sigma}$, are initialized according to the spacing between neighboring centers to avoid excessive overlap, while the rotation angles θ are sampled randomly to promote diversity in kernel orientations. Kernel weights $\mathbf{w}$ are initialized to small or zero values, allowing the model to learn the optimal contribution of each basis function during training. All initialized quantities are collected in the trainable parameter set $\boldsymbol{\Theta}$.

Parameter Projection. To enhance numerical stability and regularize the training of RBF-based PINNs, kernel parameters are periodically projected onto physically meaningful bounds. For two-dimensional problems, the RBF center locations $\boldsymbol{\mu}$ are clipped to remain within the spatial domain defined by the evaluation points. The kernel shape parameters $\boldsymbol{\lambda}$, represented by the anisotropic standard deviations $\boldsymbol{\sigma}$, are constrained in log-space between a minimum value proportional to the average spacing of evaluation points and a maximum value related to the overall domain size. This ensures that kernels are neither too narrow to overfit local noise nor too wide to miss fine-scale structures. See Table 3 for more details.

Component (regularized)	**Projection**
Centers $\boldsymbol{\mu} = (\mu_x, \mu_y)$	$\mu_x \in [x_{\min}, x_{\max}],\ \mu_y \in [y_{\min}, y_{\max}]$
Log-scales $\log\boldsymbol{\sigma} = (\log\sigma_x, \log\sigma_y)$	$\log\sigma_x \in [\log\sigma_{\min}, \log\sigma_{\max}],\ \log\sigma_y \in [\log\sigma_{\min}, \log\sigma_{\max}]$
Angle θ	$\theta \leftarrow \theta \bmod 2\pi$
Weight w	No projection

Table 3: Projection rules for RBF parameters collected in $\boldsymbol{\Theta}$.

A.2 GI-RBF Architecture and Training Details

Encoder The parametric PDE inputs are denoted by $\boldsymbol{c}$, while $\boldsymbol{\Theta}$ collects all trainable RBF parameters, including weights $\mathbf{w}$, centers $\boldsymbol{\mu}$, and widths $\boldsymbol{\sigma}$. The mean and standard deviation of the latent sampling distribution are denoted by $\boldsymbol{\mu}_\ell$ and $\boldsymbol{\sigma}_\ell$, respectively. A lightweight MLP encoder maps $\boldsymbol{c}$ to a Gaussian latent distribution $q_\phi(z \mid \boldsymbol{c}) = \mathcal{N}(\boldsymbol{\mu}_\ell(\boldsymbol{c}), \boldsymbol{\sigma}_\ell^2(\boldsymbol{c}))$. The active configuration uses Fourier feature conditioning, concatenating Fourier embeddings of $\boldsymbol{c}$ to the encoder input. Dropout provides regularization, and the log-variance is constrained via a $\tanh$ squashing to avoid numerical instability during KL computation. In encoder sampling mode, z is drawn using the reparameterization trick; for stability-focused benchmarking, z may be fixed to the mean $\boldsymbol{\mu}_\ell(\boldsymbol{c})$.

Decoder The decoder is a Set Transformer that generates K RBF kernels conditioned on $(z, \boldsymbol{c})$. A dense layer followed by ReLU maps the concatenated vector $[z, \boldsymbol{c}]$ to Kd_h values, reshaped into K kernel embeddings of width $d_h = 32$. Stacked ISAB attention blocks then model interactions among

kernels Lee et al. (2019). A multi-head output layer produces kernel parameters with domain-aware scaling: centers $\boldsymbol{\mu} \in \mathbb{R}^d$ are mapped to the configured domain bounds per dimension, log-widths are mapped to a domain-adaptive $\boldsymbol{\sigma}$ range, the angle head predicts a normalized two-component direction and converts it through $\mathrm{atan2}$ to a scalar angle in $[0, 2\pi)$, and weights $\mathbf{w}$ are bounded to $[-1, 1]$ using $\tanh$. This bound applies only to the decoder output at initialization; subsequent refinement updates the weights without projecting them to this interval (Table 3). Domain bounds act as normalization limits that restrict the range of kernel parameters such as $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. For 2D domains, this yields $(K, 6)$ parameters $[\mu_x, \mu_y, \log \sigma_x, \log \sigma_y, \theta, w]$ (time t is treated as a second axis when present). Parameter reconstruction compares the stored scalar angles directly using squared differences, so it remains sensitive to angular wrap-around.

Regularization Training uses a VAE-style KL term $D_{\mathrm{KL}}(q_\phi(z \mid \boldsymbol{c}) \,\|\, \mathcal{N}(0, I))$ and a kernel repulsion penalty to discourage center collapse:

$$\mathcal{L}_{\text{rep}} = \lambda_{\text{rep}} \frac{2}{K(K-1)} \sum_{i<j} \frac{1}{1 + \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|_2^2 / s_{\text{rep}}^2}, \tag{9}$$

where s_{rep} is a domain-scaled length (Appendix Table 4). A zero-shot self-bias penalty can also be added to the adaptation loss during GI-RBF training, and optionally during online refinement:

$$\mathcal{L}_{\text{zs}} = \left\| u(X; \boldsymbol{\Theta}_{\text{FT}}) - u(X; \boldsymbol{\Theta}_{\text{ZS}}) \right\|_2^2. \tag{10}$$

Here, X denotes collocation anchor points, and $u(X; \boldsymbol{\Theta}_{\text{ZS}})$ acts as a self-distillation pseudo-observation generated by the initializer itself, not as external supervised data. In the final deployment configuration, this term is used as a training-time shaping regularizer and disabled during online refinement unless explicitly ablated. In the inner-optimization setting, the reconstruction weight is adaptively scaled using an EMA of the relative improvement in task loss, $\beta_{\text{eff}} = \mathrm{clip}(\beta(1 - \bar{r})^p, \beta_{\min}, \beta)$, where $\bar{r}$ is the EMA of the batch mean of the per-instance relative improvements $(\ell_{\text{init}} - \ell_{\text{opt}})/(\ell_{\text{init}} + \varepsilon)$. Each relative improvement is clipped to $[0, 1]$ before averaging, with $\varepsilon = 10^{-8}$. This schedule is a heuristic for balancing parameter matching against the other outer-objective terms: it reduces the reconstruction coefficient when inner-loop refinement substantially lowers the task loss and retains a larger coefficient when improvement is limited.

Hyperparameter Configuration Table 4 summarizes the GI-RBF architecture and core training configuration used in the main experiments. The final reported GI-RBF models use a uniform 30k outer-update budget across equations for consistency; the corresponding offline cost is reported separately in Table 14.

The four canonical PDE benchmarks use the fixed 64×64 dataset grid for residual evaluation. Boundary and initial-condition losses use the corresponding grid edges. These coordinates are reused across optimization steps.

A.3 Refinement-Informed Training Details

This appendix provides implementation details for the refinement-informed objective in Section 4.4 and Eq. (7). It does not introduce a separate training objective; it specifies the optimizers, numerical safeguards, and update schedule used in the experiments.

A.3.1 Inner-Loop Adaptation

The inner loop refines task-specific parameters using a small number of gradient-based optimization steps (Algorithm 1). To ensure numerical stability during adaptation, loss and gradient values are sanitized to prevent the occurrence of NaN or Inf entries, and an optional domain projection of the RBF parameters may be applied. The inner-loop optimizer uses AdamW with the learning rate and weight decay listed in Table 4.

A.3.2 Outer-Loop Hypernetwork Training

The outer loop updates the hypernetwork parameters using a batch of PDE instances, as summarized in Algorithm 2. For each task, a latent variable z_i is sampled to generate zero-shot RBF parameters

Table 4: GI-RBF configuration used in the main experiments.

Hyperparameter	Value	Notes
Encoder		
latent_dim	16	Latent size for $q_\phi(z \mid \boldsymbol{c})$
hidden_dims	[64, 64]	MLP widths
encoder_dropout	0.05	Dropout rate
encoder_sigma_min	0.25	Minimum encoder stddev
conditioning_type	fourier	Conditioning mode
fourier_dim	32	Fourier embedding size
fourier_scale	1.0	Fourier frequency scale
Decoder		
num_kernels	256 (128 ablation)	RBF kernels per sample
set_transformer_dims	[32, 32]	Attention width
num_attention_heads	4	Multi-head attention
num_attention_layers	4	ISAB depth
num_inducing_points	32	Inducing points for ISAB
decoder_dropout	0.01	Dropout rate
domain_x	$[x_{\min}, x_{\max}]$	Normalization bounds for μ_x, σ_x
domain_t (or y)	$[t_{\min}, t_{\max}]$ / $[y_{\min}, y_{\max}]$	Normalization bounds for $\mu_t/\mu_y, \sigma_t/\sigma_y$
Training		
trainer_type	dual_loss_inner_opt	Refinement-informed training with a short teacher adaptation
outer_updates	30000	Uniform GI-RBF training budget
batch_size	10	PDE instances per batch
inner_epochs	25	Inner-loop steps during training
outer_optimizer	Muon	$\eta_{\text{outer}} = 3 \times 10^{-4}$, wd=$10^{-2}$, clip=10
inner_optimizer	AdamW	$\eta_{\text{inner}} = 10^{-2}$, wd=$10^{-4}$
Regularization		
α	0.05	Weight on the task loss in the outer objective
γ	1.0	Weight on the KL term
β	10.0	Maximum reconstruction weight
adaptive_β	on	$\tau = 0.5$, $p = 2$, $\beta_{\min} = 5 \times 10^{-4}$
repulsion	on	$\lambda_{\text{rep}} = 0.75$, $s_{\text{rep}} = 0.25$ of domain size
zero-shot self-bias	train on / online off	$\lambda_{\text{zs}} = 1.0$

Algorithm 1 Inner-Loop Adaptation with Numerical Stabilization

Input: zero-shot RBF parameters $\boldsymbol{\Theta}_{\text{ZS}}$, PDE parameters $\boldsymbol{c}$, collocation points $\boldsymbol{X}$
Initialize optimizer state $s_0 \leftarrow \text{opt.init}(\boldsymbol{\Theta}_{\text{ZS}})$
Set $\boldsymbol{\Theta}_0 \leftarrow \boldsymbol{\Theta}_{\text{ZS}}$
for $t = 1$ to N_{inner} **do**
  Compute task loss: $\ell_{\text{task}} \leftarrow \mathcal{L}_{\text{task}}(\boldsymbol{\Theta}_{t-1}; \boldsymbol{c}, \boldsymbol{X})$
  Compute gradients $g \leftarrow \nabla_{\boldsymbol{\Theta}} \ell_{\text{task}}$, sanitized for NaN/Inf
  Update step and optimizer state: $\Delta, s_t \leftarrow \text{opt.update}(g, s_{t-1}, \boldsymbol{\Theta}_{t-1})$
  Apply update: $\boldsymbol{\Theta}_t \leftarrow \boldsymbol{\Theta}_{t-1} + \Delta$
  if enabled: project parameters $\boldsymbol{\Theta}_t \leftarrow \text{project}(\boldsymbol{\Theta}_t)$
end for
Return: fine-tuned parameters $\boldsymbol{\Theta}_{\text{FT}} = \boldsymbol{\Theta}_{N_{\text{inner}}}$

$\boldsymbol{\Theta}_{\text{ZS},i}$, and the corresponding task loss is evaluated. These parameters are then refined through inner-loop adaptation to obtain fine-tuned parameters $\boldsymbol{\Theta}_{\text{FT},i}$, which are used to compute the reconstruction term in Eq. (7). Gradients $\nabla_\phi \mathcal{L}$ are computed with respect to the hypernetwork parameters and updated using Muon with the learning rate, weight decay, and gradient-clipping settings listed in Table 4.

Algorithm 2 Outer-Loop Hypernetwork Training

Input: batch of PDE parameters $\{\boldsymbol{c}_i\}_{i=1}^{B}$, hypernetwork parameters ϕ
for each $\boldsymbol{c}_i$ in parallel **do**
  Sample latent variable: $\boldsymbol{z}_i \sim \mathcal{N}\Big(\boldsymbol{\mu}_\phi(\boldsymbol{c}_i), \mathrm{diag}\Big(\boldsymbol{\sigma}_\phi^2(\boldsymbol{c}_i)\Big)\Big)$
  Generate zero-shot RBF parameters: $\boldsymbol{\Theta}_{\mathrm{ZS},i} = g_\phi(\boldsymbol{z}_i, \boldsymbol{c}_i)$
  Evaluate initial task loss: $\ell_{\mathrm{init},i} = \mathcal{L}_{\mathrm{task}}(\boldsymbol{\Theta}_{\mathrm{ZS},i}; \boldsymbol{c}_i)$
  Run inner-loop adaptation: $\boldsymbol{\Theta}_{\mathrm{FT},i} = \textsc{InnerOpt}(\boldsymbol{\Theta}_{\mathrm{ZS},i}, \boldsymbol{c}_i)$
  Evaluate optimized task loss: $\ell_{\mathrm{opt},i} = \mathcal{L}_{\mathrm{task}}(\boldsymbol{\Theta}_{\mathrm{FT},i}; \boldsymbol{c}_i)$
  Compute reconstruction loss: $\ell_{\mathrm{recon},i} = \mathcal{L}_{\mathrm{recon}}(\boldsymbol{\Theta}_{\mathrm{ZS},i}, \boldsymbol{\Theta}_{\mathrm{FT},i})$
end for
Aggregate batch losses and compute adaptive weighting β_{eff}
Compute total outer-loop loss: $\mathcal{L} = \alpha\bar{\ell}_{\mathrm{init}} + \gamma\bar{D}_{\mathrm{KL}} + \beta_{\mathrm{eff}}\bar{\ell}_{\mathrm{recon}}$
Compute gradients $\nabla_\phi\mathcal{L}$ and update hypernetwork parameters using a momentum-based optimizer (Muon) with learning rate η_{outer}, weight decay 10^{-2}, and gradient clipping at norm 10

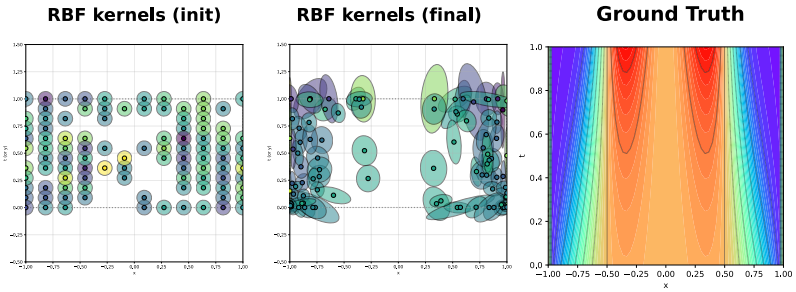

Figure 6: Standard RBF initialization (left) vs. optimized kernel configuration (middle) for a representative Allen–Cahn case (reduced number of kernels shown for clarity), alongside the ground-truth solution (right).

A.3.3 Summary

The inner-loop focuses on rapid, task-specific refinement of RBF parameters, while the outer-loop trains the hypernetwork to produce strong initializations that minimize inner-loop adaptation. Gradients are typically stopped through the inner loop to improve stability and reduce computation. Figure 6 illustrates this refinement for a representative Allen–Cahn case, comparing the initial kernel configuration (left) against the optimized configuration after training (middle), alongside the ground-truth solution (right).

A.4 Strengthened refinement settings

Table 2 compares Standard-RBF and GI-RBF within each refinement setting. The strengthened settings use equation-specific objectives and solver-aligned initializers; the difference from the original setting is not a single-factor ablation.

Allen–Cahn. We solve sequentially on $[0, 1/3]$, $[1/3, 2/3]$, and $[2/3, 1]$, using $K = 256$ kernels per window. For a window $[t_j, t_{j+1}]$, define $\xi = (x + 1)/2$ and $\tau = (t - t_j)/(t_{j+1} - t_j)$. The solution takes the form

$$u_j(x, t) = G_j(x, t) + 4\tau\xi(1 - \xi)v_{\boldsymbol{\Theta}_j}(x, t),$$

where $v_{\boldsymbol{\Theta}_j}$ is the RBF field and G_j blends the prescribed left/right boundary traces linearly in ξ, adding $(1-\tau)$ times the difference between the window's initial profile and that boundary blend at t_j. This enforces the initial and boundary values exactly, while refinement minimizes the PDE residual. At deployment, each window receives the preceding window's terminal prediction, with no reference profile supplied after the first window. The shared initializer is conditioned on the PDE parameters, window identity, and available initial profile. Table 2 uses 1,000 updates per window, or 3,000 total.

Helmholtz. We enforce the prescribed boundary trace g by the additive correction $u_\Theta = v_\Theta + \mathcal{C}[g] - \mathcal{C}[v_\Theta|_{\partial\Omega}]$, where $\mathcal{C}$ interpolates the four edge traces and subtracts their bilinear corner interpolant (a Coons patch). On the native 64×64 grid, the interior PDE residual is expanded in an orthonormal discrete sine basis (DST-I). With $\lambda_{mn} = (\pi/2)^2(m^2 + n^2)$ for $m, n = 1, \ldots, 62$, the modal weights are

$$w_{mn} = 1 + \min\left\{ W_{\max} - 1, \frac{\lambda_{11}^2}{(k^2 - \lambda_{mn})^2 + (\delta\lambda_{11})^2} \right\}.$$

We use $(\delta, W_{\max}) = (0.01, 10^4)$ when $\min_{m,n} |k^2 - \lambda_{mn}| < 0.1\lambda_{11}$, and $(0.1, 100)$ otherwise. The objective adds $62^{-2} \sum_{m,n}(w_{mn} - 1)|\widehat{r}_{mn}|^2$ to the full-grid squared-residual average, where $\widehat{r}_{mn}$ are the interior residual coefficients. The stronger branch uses normalized trapezoidal weights for the full-grid average; the other branch uses a uniform average. The gate depends only on k and the domain spectrum. Both initializations use the same projected objective and 1,000 refinement updates in Table 2.

B Experiment I Support: Adaptive Model Selection

This section supports Experiment I by comparing RBF, SPINN, PIG, and a standard MLP PINN on the same 40 standard-split test samples per equation (seed 0). Each model is trained for 50, 250, and 1000 epochs under both the deployed PINN loss and a supervised data-fitting loss, separating optimization behavior from representational capacity. AdamW learning rates and model settings are listed in Table 7; timings are train-only and exclude data loading/evaluation overhead. For each model, the learning rate was tuned for convergence speed, selecting the best-performing rate among those tested. The comparison uses these model-specific learning rates under the same reported optimization budgets.

Figure 7 audits the RBF capacity by varying the kernel count K. In the reported two-coordinate formulation, each kernel contributes six trainable parameters, so the online parameter budget scales linearly with K. Data-driven error remains low as K increases, while physics-informed error generally improves until diminishing returns or optimization variability appear. This supports the interpretation that the physics-informed objective, rather than representation capacity alone, is an important bottleneck on these cases. The corresponding 1,000-update online cost remains approximately one second for $K \leq 512$ under this single-seed diagnostic.

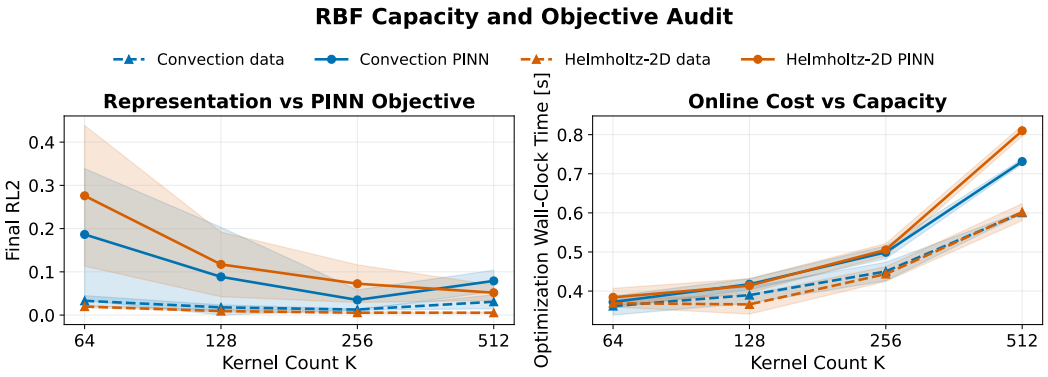

Figure 7: RBF capacity scaling under a 1,000-update budget. Left: relative-L^2 error as a function of kernel count under supervised and physics-informed objectives. Right: corresponding wall-clock time. This single-seed diagnostic motivates the selected capacity; it is not a universal solver comparison.

B.1 Interpretation

Tables 5–6 support the choice of RBF as the adaptive base model. Under the PINN loss, RBF provides a strong accuracy–efficiency tradeoff across equations, while under supervised fitting it

Table 5: Comparison of models across PDEs with parameter counts, error, and training time under different training budgets. Each model is evaluated on 40 standard-split test parameters (seed 0) with a physics-informed (PINN) training objective. Values are mean over test cases; std is omitted for space. Train-only time reported.

Equation	Model	Params	50 epochs		250 epochs		1000 epochs	
			Error	Time	Error	Time	Error	Time
Allen–Cahn (1+1)D	RBF	768	0.79	0.060s	0.31	0.299s	0.22	1.195s
	SPINN	17,026	0.98	0.033s	0.99	0.163s	0.99	0.652s
	PIG	31,105	0.77	0.547s	0.85	2.733s	0.55	10.931s
	STANDARD PINN	8,577	0.98	0.132s	0.98	0.661s	1.01	2.644s
Burgers (1+1)D	RBF	768	0.53	0.063s	0.14	0.315s	0.14	1.259s
	SPINN	17,026	0.94	0.033s	0.78	0.165s	0.64	0.659s
	PIG	31,105	0.71	0.541s	1.09	2.703s	1.11	10.812s
	STANDARD PINN	8,577	0.96	0.128s	0.93	0.639s	0.70	2.556s
Convection (1+1)D	RBF	768	0.95	0.055s	0.58	0.274s	0.36	1.095s
	SPINN	17,026	0.77	0.028s	0.59	0.139s	0.59	0.557s
	PIG	31,105	0.58	0.105s	0.58	0.525s	0.61	2.100s
	STANDARD PINN	8,577	0.91	0.057s	0.67	0.284s	0.58	1.135s
Helmholtz 2D (2D)	RBF	768	0.44	0.054s	0.46	0.272s	0.36	1.088s
	SPINN	17,026	1.40	0.033s	1.58	0.166s	0.41	0.664s
	PIG	31,105	1.00	0.503s	1.00	2.516s	1.01	10.066s
	STANDARD PINN	8,577	1.00	0.102s	1.04	0.510s	1.97	2.042s

Table 6: Comparison of models across PDEs with parameter counts, error, and training time under different training budgets. Each model is evaluated on the same 40 standard-split test parameters (seed 0) with a data-driven supervised loss training objective. Values are mean over test cases; std is omitted for space. Train-only time reported.

Equation	Model	Params	50 epochs		250 epochs		1000 epochs	
			Error	Time	Error	Time	Error	Time
Allen–Cahn (1+1)D	RBF	768	0.19	0.046s	0.07	0.229s	0.01	0.917s
	SPINN	17,026	0.87	0.027s	0.42	0.135s	0.31	0.540s
	PIG	31,105	0.59	0.041s	0.22	0.203s	0.18	0.812s
	STANDARD PINN	8,577	0.92	0.030s	0.82	0.151s	0.53	0.603s
Burgers (1+1)D	RBF	768	0.09	0.045s	0.03	0.227s	0.02	0.910s
	SPINN	17,026	0.73	0.027s	0.54	0.135s	0.34	0.540s
	PIG	31,105	0.52	0.041s	0.26	0.206s	0.13	0.823s
	STANDARD PINN	8,577	0.85	0.030s	0.79	0.151s	0.55	0.605s
Convection (1+1)D	RBF	768	0.48	0.046s	0.17	0.230s	0.02	0.918s
	SPINN	17,026	0.58	0.024s	0.57	0.122s	0.57	0.487s
	PIG	31,105	0.50	0.040s	0.39	0.200s	0.17	0.802s
	STANDARD PINN	8,577	0.62	0.030s	0.58	0.148s	0.56	0.593s
Helmholtz 2D (2D)	RBF	768	0.14	0.045s	0.03	0.225s	0.01	0.902s
	SPINN	17,026	0.99	0.027s	0.94	0.135s	0.64	0.541s
	PIG	31,105	0.82	0.041s	0.58	0.204s	0.43	0.817s
	STANDARD PINN	8,577	1.00	0.030s	1.00	0.150s	0.95	0.599s

Table 7: Model configurations used for the adaptive-model benchmarks (standard modality). Learning rates were tuned separately for each model for convergence speed.

Model	Hidden dim	Layers	Learning rate	Other settings
RBF	–	–	1e-2	num_kernels=128 (6 params per kernel)
STANDARD PINN	64	3	1e-4	activation=tanh, pos_enc=0
SPINN	64	3	1e-2	rank=auto (features[-1]=1), pos_enc=0
PIG	64	3	1e-3	num_gaussians=800, pig_mlp_dim=4, activation=tanh

reaches very low errors, indicating that the main limitation is often optimization of the physics-informed objective rather than RBF expressivity. Based on these results, RBF is selected for GI-RBF because it balances fast physics-informed refinement, high supervised expressivity, and a compact, interpretable kernel parameterization that is well suited for hypernetwork prediction and targeted inference-time adaptation.

Figure 8 provides an additional qualitative capacity check: increasing the number of RBF kernels sharpens the supervised reconstruction of a Gray–Scott target, with $K = 256$ already capturing its dominant structures.

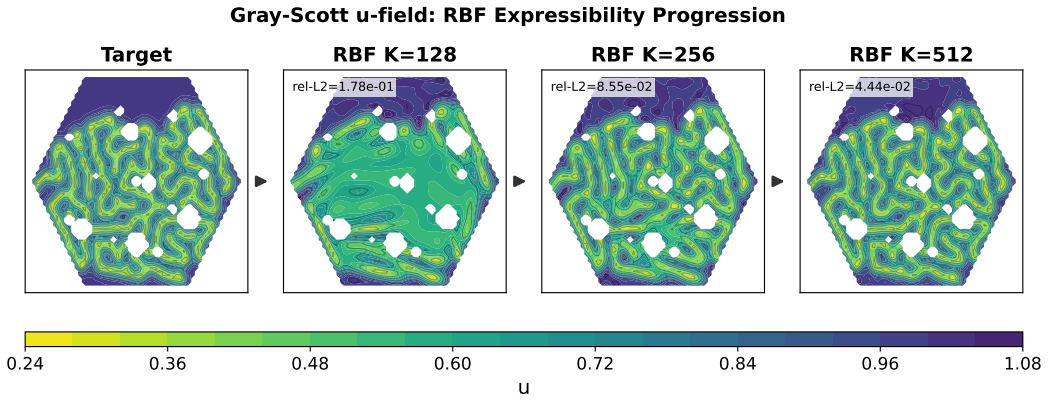

Figure 8: Data-driven RBF reconstruction of a Gray–Scott target as the kernel count increases. This is a representation-capacity diagnostic rather than a physics-informed GI-RBF result.

C Experiment II Support: Parametric Generalization

This section supports Experiment II. All methods use the same few-shot parametric split with $n_{\text{train}} = 10$ training parameter settings and 20 interpolation plus 20 extrapolation test cases per equation. Tables 11–12 report the full benchmark values.

The RBF and GI-RBF rows use the matched online-refinement settings from Figure 3: the same kernel budget, same online PINN optimizer, and different initialization only. FNO and U-Net are supervised zero-shot references trained with PDEArena (Gupta & Brandstetter, 2022; Brandstetter et al., 2022). Hyper-LR-PINN follows the default two-phase generative-initialization setup from Cho et al. (2023b); Table 9 lists the settings used here. We additionally compare with PINO (Li et al., 2024) and P^{2}INN-SVD (Cho et al., 2024): PINO uses a physics-informed neural-operator backbone followed by online physics refinement, while P^{2}INN-SVD uses the official conditional architecture and its singular-value-restricted adaptation, evaluated with the benchmark-specific physics residuals. Both baselines use the same ten training instances and the same 20 interpolation and 20 extrapolation cases per equation; their entries aggregate two independently trained seeds.

Table 8 reports refreshed GI–RBF deltas (GI-RBF minus RBF) under the same matched $K = 128$ settings as Figure 3.

Table 8: Delta in relative-L^2 decimal fractions (GI-RBF-128 minus RBF-128) for interpolation and extrapolation benchmarks under the refreshed matched adaptation settings. Negative values indicate improvement.

Setting	Model	Zero-shot	Epoch 50	Epoch 500	Epoch 1000
Allen–Cahn					
Extrapolation	Δ RBF-128	-0.511	-0.449	-0.029	-0.022
Interpolation	Δ RBF-128	-0.526	-0.476	-0.025	-0.017
Burgers					
Extrapolation	Δ RBF-128	-0.282	-0.193	-0.007	-0.007
Interpolation	Δ RBF-128	-0.413	-0.201	-0.009	-0.005
Convection					
Extrapolation	Δ RBF-128	-0.638	-0.475	-0.226	-0.105
Interpolation	Δ RBF-128	-0.693	-0.443	-0.145	-0.052
Helmholtz					
Extrapolation	Δ RBF-128	$+0.339$	-0.056	-0.461	-0.370
Interpolation	Δ RBF-128	$+0.150$	-0.109	-0.321	-0.238

Figure 9 provides a qualitative view of the same matched online-adaptation settings used in Figure 3. For each equation, we select one extrapolation sample by the median GI-RBF relative L^2 error after 50 online refinement steps, then plot the ground truth, GI-RBF prediction, non-amortized RBF prediction, and absolute-error maps on that same sample. This selection rule is representative rather than best-case; the Helmholtz row also illustrates that early-budget gains are PDE-dependent.

Table 9: Hyper-LR-PINN configuration used for the parametric few-shot benchmark. Settings follow the default two-phase training procedure from the public implementation.

Component	Setting
Backbone width	Hidden dim $= 50$ (tanh)
Phase-1 training	20,000 epochs, AdamW, lr 2.5×10^{-4}, full batch
Phase-2 training	10,000 epochs, AdamW, lr 2.5×10^{-4}, full batch
Few-shot budget	$n_{\text{train}} = 10$ parameter settings
Test split sizes	$n_{\text{interp}} = 20$, $n_{\text{extrap}} = 20$
Loss	PDE residual + IC/BC terms

Table 10 reports the resulting parameter counts and memory footprint across all compared models. GI-RBF separates the frozen generative initializer from the adapted RBF state. The initializer is evaluated once per query to produce the initial RBF parameters; subsequent physics-based adaptation updates only the $6K$ RBF parameters. No gradients or optimizer states for the initializer are needed during adaptation. We therefore report its parameter storage separately from the online solver, while accounting for both components of the model.

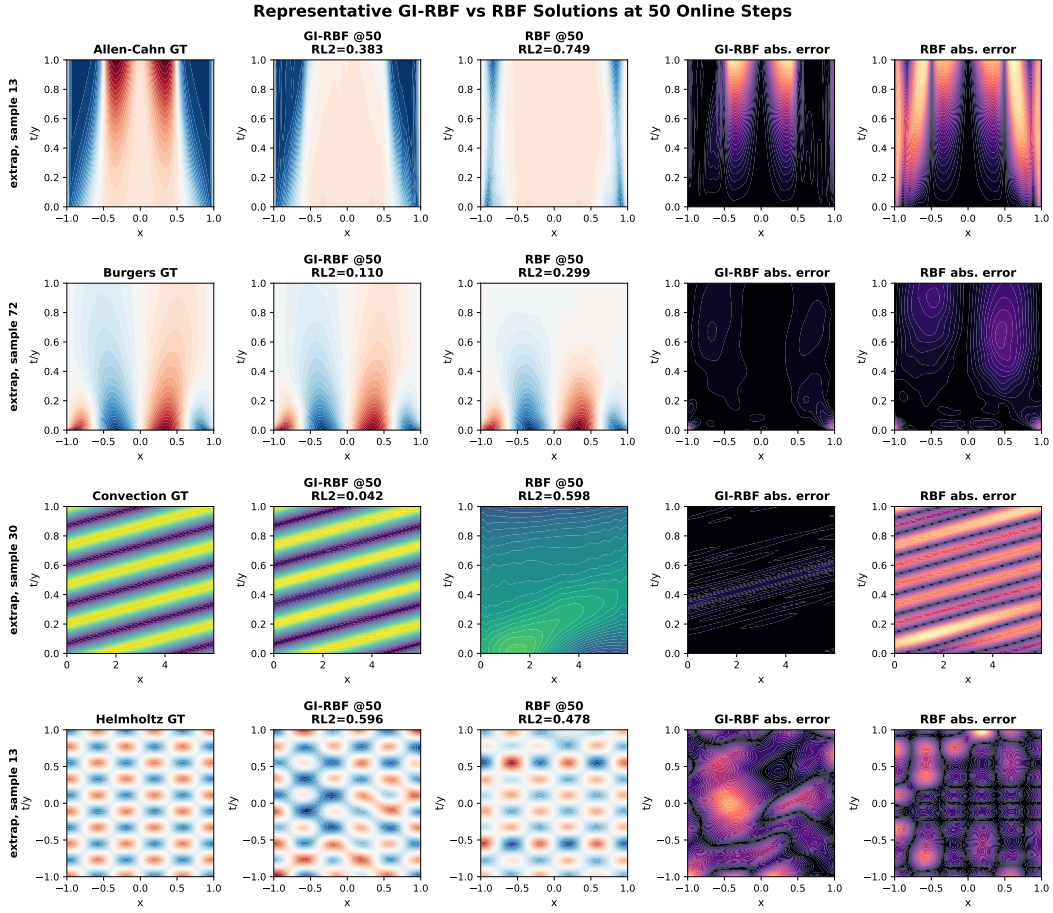

Figure 9: Representative extrapolation-split solution fields at 50 online PINN refinement steps. Rows correspond to one selected sample per equation; columns show ground truth, GI-RBF-128 prediction, non-amortized RBF-128 prediction, GI-RBF absolute error, and RBF absolute error. Samples are selected by median GI-RBF @50 relative L^2 error within each equation, and the same sample is used for all columns in a row. Both methods use the same RBF budget and online optimizer; only the initialization differs.

Table 10: Memory footprint comparison across surrogate and physics-informed models. GI-RBF is listed by component: frozen initializer and adapted RBF state. Initializer counts vary across the four PDE families; their storage counts FP32 parameters only, excluding activations and optimizer states. For Hyper-LR-PINN, peak GPU memory consumption during fine-tuning is also reported. Peak GPU memory for U-Net and FNO is not included as these models are used only for inference, making direct comparison with training memory inappropriate. PINO adapts all parameters online; its small parameter-count range reflects equation-dependent input channels. P^{2}INN-SVD has 76,851 total parameters but adapts only 550 singular-value and modulation parameters. Its online peak memory was not recorded and is therefore left unreported. RBF peak-memory entries refer to the solver alone; full GI-RBF deployment peak memory is not reported here.

Model	**# Parameters**	**Model Size (MiB)**	**Peak GPU Memory (MB)**
U-Net	146.7M	559.5	—
FNO	36.0M	137.4	—
Hyper-LR-PINN	15,351	0.1	18.68
PINO	339,385–339,481	1.30	40.34–40.42
P^{2}INN-SVD	76,851 / 550 adapted	0.29	—
RBF-128	768	0.003	1.01
RBF-256	1,536	0.006	1.01
Frozen GI-RBF initializers (not adapted online)			
$K = 128$	137,847–146,199	0.53–0.56	—
$K = 256$	211,575–228,119	0.81–0.87	—

Table 11: Interpolation performance comparison across surrogate and physics-informed models on parametric PDE benchmarks. RL2 values are decimal fractions and include zero-shot error and error after online refinement at different budgets (mean $\pm$ std), together with approximate computational costs. U-Net/FNO times are seconds per case, obtained by dividing the combined 40-case evaluation time by 40 (batch size 8); they include evaluation overhead and are shared across the two splits. Hyper-LR-PINN times are approximate 1,000-update benchmark costs. RBF/GI-RBF times are per-case refinement wall times. Timing boundaries differ across methods, so these values provide rough cost estimates. The RBF and GI-RBF rows use the refreshed matched comparison from Figure 3. For PINO and P^{2}INN-SVD, the reported standard deviation is the population standard deviation of the two seed-level test-case means, and wall time is the mean per-case cumulative online adaptation through 1,000 updates on A100 GPUs.

Model	Zero-shot	Epoch 50	Epoch 500	Epoch 1000	Approx. time (s)
Allen–Cahn					
U-Net	0.011 ± 0.003	N/A	N/A	N/A	0.0114 ± 0.0031
FNO	0.004 ± 0.003	N/A	N/A	N/A	0.0073 ± 0.0017
Hyper-LR-PINN	0.155 ± 0.012	0.143 ± 0.009	0.137 ± 0.008	0.139 ± 0.009	29.2 ± 1.6
PINO	0.030 ± 0.003	0.028 ± 0.003	0.022 ± 0.003	0.019 ± 0.002	12.87 ± 0.66
P^{2}INN-SVD	0.953 ± 0.004	0.953 ± 0.004	0.953 ± 0.004	0.949 ± 0.008	14.83 ± 0.69
RBF-128	1.000 ± 0.000	0.824 ± 0.101	0.211 ± 0.172	0.162 ± 0.163	2.5 ± 0.4
RBF-256	1.000 ± 0.000	0.817 ± 0.110	0.201 ± 0.175	0.148 ± 0.161	4.6 ± 0.4
GI-RBF-128	0.474 ± 0.104	0.348 ± 0.144	0.186 ± 0.182	0.145 ± 0.173	2.1 ± 0.4
GI-RBF-256	0.507 ± 0.152	0.355 ± 0.206	0.191 ± 0.192	0.152 ± 0.182	2.2 ± 0.5
Burgers					
U-Net	0.357 ± 0.075	N/A	N/A	N/A	0.0096 ± 0.0006
FNO	0.333 ± 0.052	N/A	N/A	N/A	0.0073 ± 0.0018
Hyper-LR-PINN	0.975 ± 0.041	0.991 ± 0.144	0.748 ± 0.036	0.737 ± 0.035	25.2 ± 1.4
PINO	0.324 ± 0.036	0.133 ± 0.005	0.060 ± 0.0005	0.053 ± 0.003	10.17 ± 0.02
P^{2}INN-SVD	0.657 ± 0.077	0.623 ± 0.040	0.596 ± 0.029	0.596 ± 0.026	15.00 ± 0.82
RBF-128	1.000 ± 0.000	0.304 ± 0.123	0.056 ± 0.037	0.051 ± 0.037	2.3 ± 0.0
RBF-256	1.000 ± 0.000	0.398 ± 0.159	0.052 ± 0.041	0.048 ± 0.037	4.5 ± 0.1
GI-RBF-128	0.587 ± 0.213	0.103 ± 0.052	0.047 ± 0.036	0.046 ± 0.036	2.1 ± 0.4
GI-RBF-256	0.634 ± 0.232	0.100 ± 0.056	0.047 ± 0.040	0.045 ± 0.038	2.0 ± 0.5
Convection					
U-Net	0.135 ± 0.028	N/A	N/A	N/A	0.0111 ± 0.0026
FNO	0.110 ± 0.025	N/A	N/A	N/A	0.0070 ± 0.0020
Hyper-LR-PINN	0.813 ± 0.091	0.672 ± 0.025	0.577 ± 0.001	0.577 ± 0.001	28.5 ± 1.6
PINO	0.126 ± 0.010	0.091 ± 0.005	0.031 ± 0.009	0.024 ± 0.007	11.97 ± 0.71
P^{2}INN-SVD	0.137 ± 0.011	0.051 ± 0.005	0.022 ± 0.002	0.018 ± 0.001	14.30 ± 0.07
RBF-128	1.000 ± 0.000	0.473 ± 0.192	0.152 ± 0.173	0.059 ± 0.093	2.1 ± 0.0
RBF-256	1.000 ± 0.000	0.475 ± 0.228	0.157 ± 0.180	0.064 ± 0.104	3.7 ± 0.1
GI-RBF-128	0.307 ± 0.162	0.030 ± 0.025	0.007 ± 0.006	0.007 ± 0.006	1.9 ± 0.0
GI-RBF-256	0.313 ± 0.156	0.019 ± 0.014	0.006 ± 0.004	0.005 ± 0.003	2.1 ± 0.0
Helmholtz					
U-Net	1.108 ± 0.016	N/A	N/A	N/A	0.0113 ± 0.0029
FNO	1.060 ± 0.033	N/A	N/A	N/A	0.0070 ± 0.0019
Hyper-LR-PINN	3.148 ± 3.038	3.037 ± 2.880	2.741 ± 2.452	2.497 ± 1.985	34.8 ± 1.5
PINO	0.452 ± 0.017	0.271 ± 0.001	0.151 ± 0.004	0.110 ± 0.008	13.32 ± 0.16
P^{2}INN-SVD	2.610 ± 1.607	2.284 ± 1.281	2.030 ± 1.027	2.022 ± 1.019	19.47 ± 0.92
RBF-128	1.000 ± 0.000	0.522 ± 0.176	0.448 ± 0.351	0.319 ± 0.257	2.3 ± 0.1
RBF-256	1.000 ± 0.000	0.431 ± 0.181	0.290 ± 0.251	0.210 ± 0.213	4.3 ± 0.1
GI-RBF-128	1.150 ± 0.322	0.413 ± 0.302	0.127 ± 0.142	0.081 ± 0.127	1.9 ± 0.5
GI-RBF-256	1.092 ± 0.296	0.345 ± 0.200	0.071 ± 0.106	0.046 ± 0.092	2.2 ± 0.5

Table 12: Extrapolation performance comparison across surrogate and physics-informed models on parametric PDE benchmarks. RL2 values are decimal fractions and include zero-shot error and error after online refinement at different budgets (mean ± std), together with approximate computational costs. U-Net/FNO times are seconds per case, obtained by dividing the combined 40-case evaluation time by 40 (batch size 8); they include evaluation overhead and are shared across the two splits. Hyper-LR-PINN times are approximate 1,000-update benchmark costs. RBF/GI-RBF times are per-case refinement wall times. Timing boundaries differ across methods, so these values provide rough cost estimates. The RBF and GI-RBF rows use the refreshed matched comparison from Figure 3. For PINO and P^{2}INN-SVD, the reported standard deviation is the population standard deviation of the two seed-level test-case means, and wall time is the mean per-case cumulative online adaptation through 1,000 updates on A100 GPUs.

Model	Zero-shot	Epoch 50	Epoch 500	Epoch 1000	Approx. time (s)
Allen–Cahn					
U-Net	0.026 ± 0.007	N/A	N/A	N/A	0.0114 ± 0.0031
FNO	0.022 ± 0.011	N/A	N/A	N/A	0.0073 ± 0.0017
Hyper-LR-PINN	0.228 ± 0.025	0.249 ± 0.039	0.278 ± 0.030	0.291 ± 0.041	29.2 ± 1.6
PINO	0.092 ± 0.007	0.094 ± 0.006	0.086 ± 0.009	0.078 ± 0.009	13.10 ± 0.32
P^{2}INN-SVD	0.961 ± 0.003	0.961 ± 0.003	0.953 ± 0.012	0.951 ± 0.014	14.74 ± 0.75
RBF-128	1.000 ± 0.000	0.796 ± 0.119	0.241 ± 0.167	0.197 ± 0.156	2.4 ± 0.0
RBF-256	1.000 ± 0.000	0.814 ± 0.109	0.212 ± 0.171	0.140 ± 0.127	4.5 ± 0.1
GI-RBF-128	0.489 ± 0.096	0.347 ± 0.151	0.212 ± 0.186	0.175 ± 0.160	2.1 ± 0.4
GI-RBF-256	0.512 ± 0.142	0.363 ± 0.196	0.187 ± 0.180	0.135 ± 0.145	2.0 ± 0.5
Burgers					
U-Net	0.396 ± 0.010	N/A	N/A	N/A	0.0096 ± 0.0006
FNO	0.389 ± 0.049	N/A	N/A	N/A	0.0073 ± 0.0018
Hyper-LR-PINN	0.976 ± 0.039	0.901 ± 0.015	0.889 ± 0.023	0.910 ± 0.056	25.2 ± 1.4
PINO	0.482 ± 0.051	0.211 ± 0.012	0.098 ± 0.005	0.088 ± 0.007	10.06 ± 0.02
P^{2}INN-SVD	0.593 ± 0.008	0.566 ± 0.035	0.571 ± 0.056	0.580 ± 0.054	14.85 ± 0.92
RBF-128	1.000 ± 0.000	0.326 ± 0.101	0.052 ± 0.033	0.045 ± 0.026	2.3 ± 0.0
RBF-256	1.000 ± 0.000	0.343 ± 0.156	0.051 ± 0.033	0.045 ± 0.031	4.5 ± 0.1
GI-RBF-128	0.718 ± 0.440	0.133 ± 0.085	0.045 ± 0.030	0.038 ± 0.021	2.2 ± 0.4
GI-RBF-256	0.696 ± 0.296	0.114 ± 0.080	0.045 ± 0.032	0.042 ± 0.030	2.2 ± 0.5
Convection					
U-Net	0.494 ± 0.130	N/A	N/A	N/A	0.0111 ± 0.0026
FNO	0.428 ± 0.086	N/A	N/A	N/A	0.0070 ± 0.0020
Hyper-LR-PINN	0.511 ± 0.081	0.634 ± 0.109	0.577 ± 0.002	0.576 ± 0.002	28.5 ± 1.6
PINO	0.269 ± 0.019	0.174 ± 0.010	0.044 ± 0.010	0.030 ± 0.006	11.36 ± 0.19
P^{2}INN-SVD	0.277 ± 0.008	0.127 ± 0.008	0.065 ± 0.0004	0.050 ± 0.002	13.99 ± 0.25
RBF-128	1.000 ± 0.000	0.520 ± 0.171	0.236 ± 0.197	0.114 ± 0.131	2.0 ± 0.0
RBF-256	1.000 ± 0.000	0.448 ± 0.249	0.133 ± 0.160	0.048 ± 0.084	3.7 ± 0.1
GI-RBF-128	0.362 ± 0.171	0.045 ± 0.030	0.011 ± 0.009	0.009 ± 0.005	2.0 ± 0.0
GI-RBF-256	0.339 ± 0.160	0.024 ± 0.022	0.006 ± 0.006	0.005 ± 0.004	2.1 ± 0.0
Helmholtz					
U-Net	1.148 ± 0.026	N/A	N/A	N/A	0.0113 ± 0.0029
FNO	1.084 ± 0.040	N/A	N/A	N/A	0.0070 ± 0.0019
Hyper-LR-PINN	5.355 ± 6.159	5.257 ± 6.018	4.397 ± 4.815	3.527 ± 3.578	34.8 ± 1.5
PINO	0.408 ± 0.006	0.248 ± 0.006	0.147 ± 0.008	0.126 ± 0.009	13.24 ± 0.18
P^{2}INN-SVD	2.398 ± 1.395	2.234 ± 1.231	2.204 ± 1.201	2.135 ± 1.132	19.41 ± 0.89
RBF-128	1.000 ± 0.000	0.601 ± 0.161	0.593 ± 0.364	0.459 ± 0.346	2.3 ± 0.0
RBF-256	1.000 ± 0.000	0.469 ± 0.142	0.402 ± 0.279	0.340 ± 0.286	4.3 ± 0.1
GI-RBF-128	1.339 ± 0.404	0.545 ± 0.275	0.132 ± 0.100	0.089 ± 0.091	2.0 ± 0.4
GI-RBF-256	1.209 ± 0.223	0.446 ± 0.237	0.120 ± 0.162	0.098 ± 0.159	2.2 ± 0.5

C.1 TRANSIENT FLOW MIXING EXTENSION

The governing equation, domain, initial and boundary conditions, and evaluation reference are specified in Appendix E.3.

The Flow Mixing experiment uses a global anisotropic $K = 4096$ RBF solver and five fixed-geometry test cases. The scalar rotational transport rate is the varying task condition. Standard-RBF uses a regular grid of centers, fixed initial scales, identity rotations, and Gaussian initial weights with standard deviation 0.01. We repeat the weight initialization with seeds 68, 69, and 70 while holding the collocation seed fixed at 68. GI-RBF uses three independently trained initializers (seeds 84–86). For each method, we first average the relative-L^2 error over the same five cases within each seed, then report the mean and sample standard deviation across seeds. GI-RBF uses one posterior-mean initialization rather than a sampled multi-candidate set, matching the variational formulation used in the main method.

Table 13: Flow Mixing mean relative-L^2 error (%, lower is better) over online refinement. The target column reports the number of cases reaching 10% error and the median update of first recorded attainment. Both rows report mean ± sample standard deviation across three seeds, after averaging the five cases within each seed. Standard-RBF varies its initial weights; GI-RBF varies its initializer-training seed. Target attainment is reported per seed.

Initializer	Init.	50	250	1,000	5,000	10% attainment
Standard-RBF	99.974 ± 0.188	16.138 ± 0.107	9.217 ± 0.019	5.933 ± 0.003	4.399 ± 0.003	5/5; median 250
GI-RBF (posterior mean)	16.201 ± 4.594	7.851 ± 0.277	6.027 ± 0.222	4.429 ± 0.229	3.245 ± 0.173	5/5; median 50

Table 13 shows that, under posterior-mean deployment, GI-RBF reaches the 10% target at the first reported 50-update checkpoint and improves the endpoint relative-L^2 error over Standard-RBF. This result supports task-conditioned initialization of the explicit RBF solver; it does not by itself establish an advantage from stochastic sampling.

C.2 TIME-TO-TARGET AND AMORTIZATION ACCOUNTING

Table 15 expands the pooled main-paper summary into interpolation and extrapolation splits. All entries use the selected $K = 256$ online settings and posterior-mean GI-RBF deployment. Reach is evaluated within 1,000 refinement updates, and time is summarized only over records that attain 5%. Standard-RBF trajectories are repeated across initializer seeds only to preserve pairing.

Table 14: Offline training and inference cost breakdown for GI-RBF. Training time is for 30k GI-RBF epochs. GI-RBF inference is one generator pass per sample. Online refinement is 1,000 physics-informed RBF adaptation updates from the GI-RBF initialization. Parameter counts refer to the Allen–Cahn configuration used in this scaling study.

	Parameters		Cost		
K	RBF	GI-RBF	Train (min)	GI-RBF infer. (ms)	RBF refine (s)
64	384	103,143	15.65 ± 0.04	0.372 ± 0.001	2.275 ± 0.027
128	768	142,055	25.12 ± 0.08	0.382 ± 0.003	2.374 ± 0.010
256	1,536	219,879	40.94 ± 0.09	0.367 ± 0.026	2.425 ± 0.026
512	3,072	375,527	73.44 ± 0.02	0.419 ± 0.000	2.714 ± 0.030

Table 15: Split-level 5% target attainment. Each time is the conditional median wall clock among reached seed–case records, and the paired number is the median first recorded update. Unmarked clocks are directly measured; ≈ marks values transferred from timing anchors with the same equation, kernel capacity, and refinement path.

Equation	Split	Std-RBF reach	GI reach	Std-RBF updates/time	GI updates/time
Allen–Cahn	Interp.	100.0%	100.0%	500 / 0.431 s	50 / ≈0.073 s
	Extrap.	100.0%	100.0%	500 / 0.430 s	50 / ≈0.073 s
Burgers	Interp.	70.0%	90.0%	500 / ≈0.347 s	175 / ≈0.124 s
	Extrap.	93.3%	98.3%	300 / ≈0.220 s	150 / ≈0.109 s
Convection	Interp.	78.3%	93.3%	375 / 0.254 s	125 / 0.086 s
	Extrap.	63.3%	96.7%	450 / 0.288 s	150 / 0.099 s
Helmholtz	Interp.	65.0%	88.3%	225 / ≈0.173 s	125 / ≈0.092 s
	Extrap.	56.7%	83.3%	387.5 / ≈0.276 s	225 / ≈0.156 s

For a fixed target and fixed deployment settings, let T_{offline} be the one-time initializer-training cost and let t_{Standard} and t_{GI} be conditional per-query times to that target. The corresponding repeated-query crossover is

$$N_{\text{break-even}} = \frac{T_{\text{offline}}}{t_{\text{Standard}} - t_{\text{GI}}}. \tag{11}$$

Table 16 evaluates a reduced-cost pretraining approach: a deterministic $K = 256$ initializer is trained for 3,000 epochs, using 25-step physics-refinement teachers. Teacher updates use all 256 kernel rows for the first 300 epochs, 128 uniformly sampled rows for the middle 2,400 epochs, and all 256 rows for the final 300 epochs; the represented field retains all kernels throughout. The objective of this approach is to lower the offline cost while retaining useful online time-to-target performance.

The offline costs, 362.4 s for Allen–Cahn and 287.2 s for Burgers, are median recorded complete training times, including compilation and warmup, across seeds 44–46. The online accuracy and reach come from these 3,000-epoch checkpoints, using strengthened windowed refinement for Allen–Cahn and normalized-residual/trace refinement for Burgers. Allen–Cahn query times are measured directly; Burgers query times are estimated by interpolating same-equation, same-capacity timing anchors at the observed target-reaching update counts. These costs belong to the scheduled-mask runs, separately from both the 30,000-epoch cost audit in Table 14 and the full-versus-masked teacher comparison in Table 18.

Table 16: Representative 5% repeated-query crossovers for the 3,000-epoch, scheduled-mask $K = 256$ configuration. Offline costs are medians across seeds 44–46 and include compilation and warmup; crossover counts use unrounded costs and times. Times are conditional on target attainment, so reach is reported alongside them. The estimates depend on the target, training and refinement settings, and hardware; the Burgers clocks are transferred estimates.

Equation	Std-RBF/GI reach	Std-RBF/GI time	Offline cost	Crossover
Allen–Cahn	100.0% / 100.0%	0.431 / 0.073 s	362.4 s	≈1,014 queries
Burgers	81.7% / 95.0%	≈0.284 / ≈0.093 s	287.2 s	≈1,505 queries

The crossover counts compare conditional successful-query times and must therefore be interpreted together with reach. In particular, the Burgers initializer both reaches the target on more records and lowers the conditional time among reached records. Queries that never attain the target within the evaluation budget are not assigned an artificial crossing time.

D Ablations and Sensitivity

This section reports compact supporting ablations used to validate the deployment settings. These are focused diagnostics rather than additional headline benchmarks: they test how the GI-RBF initializer

depends on the refinement-informed teacher, the online objective, the offline objective, the number of meta-training PDEs, and the non-amortized RBF initialization used for Figure 3.

D.1 Role of refinement-informed teacher steps

This ablation asks which outer-loss supervision term should train the generative initializer: full-field ground-truth solutions, a physics-only PINN residual, or the reported refinement-informed teacher target. Table 17 compares two zero-teacher controls against the reported 25-step refinement-informed training procedure. The first learns directly from full solution fields; the second minimizes the generated state's PINN residual and receives no full-field gradient, although reference window-start profiles remain part of this sequential training procedure. All three use the same deterministic conditional initializer, $K = 256$ RBF solver, ten training tasks, three sequential windows, optimizer, and three training seeds.

The Allen–Cahn ablations in Tables 17 and 19 use the strengthened windowed hard-constraint solver introduced in Table 2. Their endpoint comprises 970 refinement updates in each of three sequential windows (2,910 updates in total). The solver configuration and update budget therefore differ from the original 1,000-update benchmark in Tables 11–12; the ablation effects are assessed within each matched solver configuration.

Table 17: Allen–Cahn refinement-teacher ablation on an opened confirmation panel using the strengthened windowed hard-constraint solver. Values are median [minimum, maximum] across seeds 44–46. Relative-L^2 errors and 1% reach are percentages on the extrapolation split; the endpoint is 970 updates per sequential window. Offline time includes the complete training run.

Outer signal	Teacher steps	Zero-shot RL2	Endpoint RL2	1% reach	Offline time (s)
Full-field solution	0	35.47 [32.64, 41.67]	0.574 [0.572, 0.719]	85 [75, 85]	101.10 [99.61, 101.73]
PINN residual	0	33.02 [29.94, 34.58]	0.579 [0.566, 0.586]	90 [80, 95]	77.93 [77.22, 103.42]
Full field + refined target	25	**29.55 [22.87, 35.34]**	**0.387 [0.370, 0.415]**	**100 [100, 100]**	235.63 [212.23, 248.27]

Both zero-teacher controls learn useful conditional starts, showing that inner refinement is not required for conditional generation. The 25-step teacher nevertheless improves extrapolation endpoint error, 1% reach, and tail robustness, at higher offline cost. On a separate prospectively sealed 20-case extrapolation panel, direct conditional training achieved 75–95% 1% reach across seeds, whereas refinement-informed training achieved 90–100%; endpoint medians were not uniformly improved. We therefore interpret the teacher as offline alignment with the behavior of the deployment optimizer, not as a mathematically necessary component or a controlled proof that two equally accurate zero-shot fields must follow different trajectories.

D.2 Optional offline teacher masking

The short-refinement teacher normally updates all 256 decoded kernel rows. The optional masked variant retains the full RBF field but updates 128 uniformly sampled rows during each middle-phase teacher step. Across three training seeds, this reduces both warmed-loop and complete offline time, as shown in Table 18. This audit uses a later 3,000-update configuration. The original 30k and later 3k pipelines also differ in solver, conditioning, teacher schedule, collocation, and implementation, so their difference cannot be attributed to epoch count alone. The masking timing gain does not imply accuracy equivalence: Allen–Cahn retains 1% reach on all nine interpolation and nine extrapolation seed–case records, but its endpoints are not identical; on Burgers, interpolation remains broadly comparable while the extrapolation median is modestly worse. Residual-guided row selection does not consistently improve on the simpler random mask.

Table 18: Offline time for full and random-half short-refinement teachers. Values are median seconds across training seeds 44–46. Warmed time excludes compilation; total time includes compilation and warmup. Savings are percentages relative to the corresponding full-row teacher.

Equation	Full warmed/total (s)	Random-half warmed/total (s)	Saving (%)
Allen–Cahn	197.74 / 230.52	153.60 / 213.80	22.3 / 7.3
Burgers	453.33 / 476.72	324.63 / 370.39	28.4 / 22.3

Through Eq. (11), reducing offline time can lower the repeated-query crossover. This interpretation applies only when the masked initializer preserves the relevant online reach and time-to-target behavior; the accuracy changes above prevent treating masking as a universally lossless reduction.

D.3 Role of the variational component

We isolate latent sampling from KL regularization without introducing separate named methods. All three controls use the same decoder, RBF solver, training tasks, refinement-informed targets, optimizer, and three training seeds. The deterministic-versus-stochastic contrast tests reparameterized sampling, while the stochastic-versus-variational contrast tests the KL contribution.

Table 19: Matched Allen–Cahn ablation of the variational component using the strengthened windowed hard-constraint solver. Sampling refers to stochastic reparameterization during training; all single-start evaluations decode the conditional mean. Endpoint ranges are relative-L^2 percentages across seeds 44–46 after 970 refinement updates per sequential window.

Ablation	Sampling	KL	Purpose	Interp. (%)	Extrap. (%)
Deterministic control	No	0	Is stochasticity required?	**0.242–0.335**	**0.370–0.415**
Stochastic control	Yes	0	Does sampling alone help?	0.341–0.375	0.526–0.884
Variational model	Yes	> 0	Does KL add value?	0.350–0.436	0.541–0.798

Table 19 shows that the deterministic control is the strongest single-start configuration in this matched experiment, so variational training is not required for the primary acceleration. Its supported role is narrower: it provides a distribution of candidate initializations and an associated prediction-dispersion diagnostic. On a previously opened six-case Allen–Cahn extrapolation panel, four-candidate KL-regularized sets improved over stochastic training without KL in all three seeds: the endpoint median decreased from 0.507% to 0.471%, 0.442% to 0.197%, and 0.225% to 0.144%, respectively, while preserving 1% reach and reducing the reach-matched conditional median update count by 50% in each seed. This comparison isolates a bounded KL benefit relative to the stochastic control.

A separate eight-sample oracle audit found a better refined candidate than the posterior mean for every seed and split on the same opened Allen–Cahn panel, demonstrating multi-candidate headroom but not a deployable selection rule because the oracle uses the reference solution. With a target-free residual selector, a fully parallel four-candidate set improved the equal-work endpoint in all three seeds, but used four times the optimizer work, approximately $3.11\times$ the compiled temporary memory, and was 70.8–76.4% slower to the 1% target. It is therefore a bounded compute–accuracy option rather than a speed advantage. Across-sample field dispersion was spatially associated with local error on Allen–Cahn and Burgers, which makes it a useful ensemble-disagreement diagnostic; however, it was not calibrated as task-level uncertainty and did not improve active kernel selection over random masking. Consequently, neither sampling nor KL is claimed to provide a universal single-start, wall-clock, or uncertainty advantage, and sampling benefits remain equation-dependent.

D.4 Deployment Objective and Zero-Shot Self-Bias

Table 20 summarizes the two most relevant Allen–Cahn deployment/objective ablations on the extrapolation split. Here, the zero-shot self-bias term refers to the auxiliary consistency penalty $\|u(X;\Theta_{\mathrm{FT}}) - u(X;\Theta_{\mathrm{ZS}})\|_2^2$ on collocation anchor points, not to the outer task-loss term in Eq. (7). These anchors behave like pseudo-observations from the GI-RBF zero-shot prediction and can be used during GI-RBF training or optionally during online refinement.

The ablation shows that using this self-bias during online refinement can trap the optimizer near the initial prediction, whereas training with the shaping term and disabling it during online deployment gives the best refinement behavior. The offline-objective ablation shows that all three outer signals learn useful initializations. In this focused experiment, however, the pure-PINN and hybrid objectives outperform the fully data-driven objective both initially and after refinement, indicating that outer-objective alignment can matter in addition to access to solution data.

Table 20: Allen–Cahn GI-RBF deployment/objective ablations on extrapolation. Values are mean RL2 decimal fractions; lower is better. The zero-shot self-bias and offline-objective ablations use their own focused experiments, so comparisons should be read within each ablation group.

Ablation	Variant	Init	@50	@500	@1000	Best
ZS self-bias	Standard RBF, no self-bias	1.0009	0.8628	0.2245	0.0962	0.0944
ZS self-bias	GI-RBF, no self-bias term	0.5049	0.4362	0.1936	0.1152	0.1016
ZS self-bias	GI-RBF, online self-bias on	0.4769	0.4404	0.4279	0.4293	0.4266
ZS self-bias	GI-RBF, train self-bias on / online off	0.4770	0.4329	0.1587	0.0514	0.0494
Objective	GI-RBF, data-driven outer objective	0.5435	0.3907	0.1802	0.1160	0.1097
Objective	GI-RBF, hybrid 50/50 objective	0.4484	0.3712	0.1334	0.1013	0.0907
Objective	GI-RBF, pure PINN objective	0.4357	0.3603	0.1375	0.0986	0.0863

D.5 Meta-Training Size Sensitivity

Table 21 reports meta-training-size sensitivity. Increasing n_{train} generally improves the GI-RBF initialization, especially on extrapolation, where the zero-shot RL2 decreases monotonically from 0.3758 to 0.3331. However, later online-refined errors are not strictly monotone: after initialization, performance also depends on whether the generated kernels enter a favorable PINN refinement basin under the fixed online optimizer. This suggests that larger meta-training sets help the amortized prior, but final refinement quality is also controlled by objective/basin alignment. Improving that alignment may require stronger generator architectures or training strategies that reduce reliance on expensive inner-loop target search.

Table 21: Allen–Cahn meta-training-size sensitivity under the current deployment rule. Smaller training sets are nested subsets of the $n_{\text{train}} = 50$ reference pool. Values are mean RL2 decimal fractions over 40 test evaluations per split.

Split / n_{train}	Init	@50	@500	@1000	Best
Extrap / 5	0.3758	0.2801	0.1876	0.1621	0.1222
Extrap / 10	0.3707	0.2826	0.1936	0.1633	0.1251
Extrap / 25	0.3533	0.2717	0.2054	0.1694	0.1322
Extrap / 50	0.3331	0.2745	0.2150	0.1760	0.1354
Interp / 5	0.4400	0.3293	0.0966	0.0661	0.0595
Interp / 10	0.4172	0.3326	0.0992	0.0657	0.0594
Interp / 25	0.3986	0.3217	0.0988	0.0654	0.0609
Interp / 50	0.4077	0.3351	0.1246	0.0845	0.0807

D.6 Kernel Budget, RBF Initialization, and Data Augmentation Sensitivity

The kernel/runtime scaling audit is summarized numerically in Table 14: GI-RBF training time increases with K, but generator inference remains sub-millisecond and the dominant online cost is the shared RBF refinement. The RBF-initialization sensitivity audit makes the non-amortized baseline in Figure 3 explicit: the RBF baseline uses the same $K = 128$ representation and the same online PINN optimizer as GI-RBF, but starts from a non-amortized standard RBF initialization rather than a learned generator. Finally, the data/physics augmentation ablation provides a secondary flexible-supervision check: adding PINN-only PDE samples to a scarce-data objective was modestly helpful in some cases but did not dominate the physics-only reference.

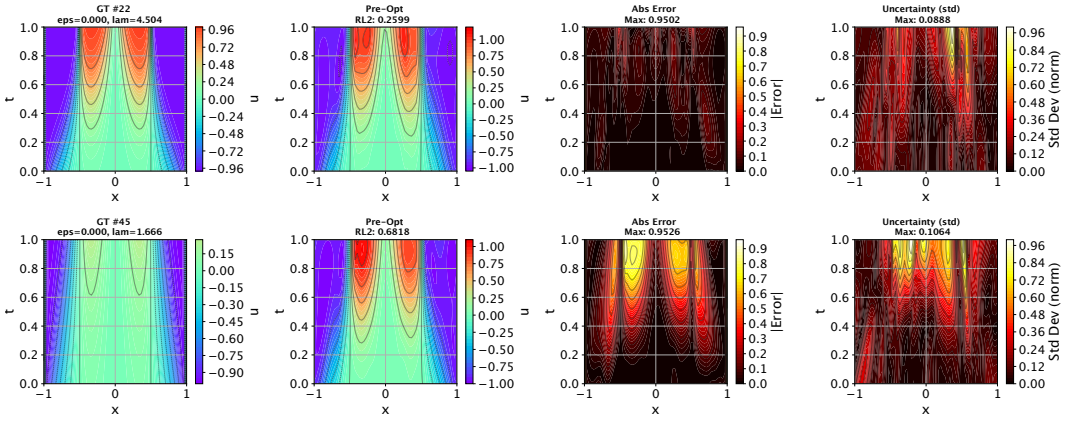

Figure 10: Prediction-level dispersion (standard deviation across multiple GI-RBF samples) vs. absolute error for representative parametric test cases from Allen–Cahn and Burgers. The examples show qualitative overlap in some regions, but do not establish calibrated predictive uncertainty. PDE residual maps (reported in the quantitative table) provide an additional physics-only diagnostic.

Table 22: Correlation between pointwise absolute error and (i) prediction-level sample dispersion and (ii) PDE residual. Values are the mean $\pm$ population standard deviation of $N = 6$ case-level correlations per row: three cases from each of two parametric split seeds (0 and 1), selected by their saved split ordering. Higher ρ indicates stronger positive correlation.

Equation	Split	Error–Dispersion (ρ)	Error–Residual (ρ)	N
Allen–Cahn	Interp	0.321 ± 0.126	0.560 ± 0.056	6
Allen–Cahn	Extrap	0.309 ± 0.177	0.635 ± 0.118	6
Burgers	Interp	0.419 ± 0.124	0.430 ± 0.156	6
Burgers	Extrap	0.498 ± 0.084	0.295 ± 0.204	6

D.7 Exploratory Dispersion Diagnostic

This separate exploratory experiment evaluates whether dispersion across variational samples is associated with spatial error in parametric PINNs. Its evaluation is separate from the main accuracy and adaptation benchmarks. For Allen–Cahn and Burgers, we use two parametric split seeds (0 and 1), with the experiment seed fixed at 44. For each equation and interpolation/extrapolation regime, we evaluate the first three cases in each saved split ordering, yielding six distinct PDE cases and six case-level correlations ($N = 6$). This is an exploratory diagnostic rather than a calibration study. Given a trained GI-RBF hypernetwork, four RBF parameter sets are sampled at inference time from the latent prior, the corresponding solutions are instantiated, and a prediction-dispersion map is computed as the pointwise standard deviation across samples. This map is compared against the pointwise absolute error (used only for evaluation) and the PDE residual, which is available without access to ground truth.

Alignment is quantified by computing per-sample correlations between: (i) absolute error and prediction-level sample dispersion, and (ii) absolute error and PDE residual. For visualization, dispersion and residual heatmaps use robust normalization (percentile clipping followed by [0,1] scaling), while correlations use raw (or log-scaled) values to avoid bias from outliers. This analysis tests whether dispersion induced by the generative initialization highlights the same regions of high error, and whether PDE residuals provide a complementary signal. Figure 10 and Table 22 report the resulting qualitative and quantitative comparisons.

A subsequent three-seed, 40-case-per-equation diagnostic found that this spatial dispersion was informative on Allen–Cahn and Burgers but was not calibrated as a measure of whole-task difficulty. Selecting 128 kernels by dispersion/support overlap also underperformed random selection at matched capacity. We therefore make no uncertainty-guided adaptation claim; the supported variational claim is limited to candidate generation and the bounded multi-candidate comparison in Appendix D.3.

E Dataset Definitions and PDE Tasks

All datasets and split definitions used in this work are publicly available through an external repository. The repository includes download instructions, dataset descriptions, and programmatic usage examples. Details are omitted for anonymity and will be provided in the camera-ready version. We summarize the split modalities and PDE tasks used throughout this work below.

E.1 Dataset Modalities

Standard In the standard modality, samples are split randomly into train and test sets without using parameter-based separation. We follow a fixed budget of $n_{\text{train}} = 25$ training samples and $n_{\text{test}} = 40$ test samples per equation with seed 0, providing a consistent baseline across datasets.

Parametric In the parametric modality, each dataset is split by parameter values rather than by individual space–time samples. A few-shot training set contains $n_{\text{train}} = 10$ parameter configurations; remaining candidates are embedded in solution space using a 5-component PCA representation and split by nearest-neighbor distance to the training set. Samples below the median distance form the interpolation set, while samples above it form the extrapolation set. For each equation we target $n_{\text{interp}} = 20$ and $n_{\text{extrap}} = 20$ samples, with balanced sampling and seeds $0, 1, 2$ for reproducibility. The configuration and resulting difficulty gap are summarized in Tables 23–24.

Table 23: Parametric split configuration.

Setting	Value	Notes
n_{train}	10	few-shot training samples
n_{each}	20	target samples per split (interp/extrap)
balance	true	balanced interp/extrap splits
method	solution_percentile	split by distance in solution space
percentile	50.0	median distance threshold
n_components	5	PCA components for solution embedding
seeds	0, 1, 2	three splits for reproducibility

Table 24: Distances are normalized by the mean leave-one-out nearest-neighbor distance within the training split, yielding a dimensionless measure of how far each regime lies relative to the intrinsic training-set density. The separation ratio (Extrap/Interp) quantifies the difficulty gap between interpolation and extrapolation regimes.

Equation	Interp	Extrap	Ratio
Allen–Cahn	0.35	2.04	5.58×
Burgers	0.59	1.40	2.40×
Convection	0.35	1.14	3.65×
Helmholtz 2D	0.68	1.30	1.91×

E.2 Equations

Allen-Cahn (1 + 1) D

The Allen–Cahn equation is considered with parameters $\varepsilon \in [1.1 \times 10^{-4}, 9.8 \times 10^{-4}]$ and $\lambda \in [1.03, 4.97]$ as:

$$\frac{\partial u}{\partial t} = \varepsilon^2 \frac{\partial^2 u}{\partial x^2} - \lambda(u^3 - u), \quad x \in [-1, 1],\ t \in [0, 1] \tag{12}$$

where $u(x, t)$ represents the state variable evolving over space and time. The parameters ε and λ control diffusion and reaction strength, respectively. Reference solutions are computed up to time $t = 1$ on a finer solver grid before mapping to the common training and evaluation grid.

The problem is equipped with homogeneous Dirichlet boundary conditions:

$$u(-1, t) = 0, \quad u(1, t) = 0, \quad t \in [0, 1], \tag{13}$$

and the initial condition:

$$u(x, 0) = \sin(\pi x), \quad x \in [-1, 1], \tag{14}$$

which specifies the initial phase profile from which the system evolves.

This equation is widely used to model phase separation and interface dynamics in materials science. Its parameterized form challenges PINNs to capture complex nonlinear patterns and sharp gradients that depend sensitively on ε and λ, making it a valuable benchmark for parametric PDE learning.

For each (ε, λ) pair, the reference solution is first computed on a finer grid with 512 spatial points and 512 temporal output points, as recorded in the dataset metadata. The resulting field is then interpolated onto the uniform 64×64 space–time grid used for training and evaluation throughout this work. Thus, 64×64 denotes the stored benchmark resolution, not the resolution of the reference solve.

Burgers (1 + 1) D

In this work, the 1D viscous Burgers' equation parameterized by amplitude A, wavenumber k, and viscosity ν is considered:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \quad (x, t) \in [-1, 1] \times [0, 1], \tag{15}$$

with the initial condition

$$u(x, 0) = A\sin(k\pi x), \tag{16}$$

where

$$A \in [0.51, 2.20], \quad k \in [0.55, 2.50], \quad \nu \in [0.014, 0.299].$$

The wavenumber k is sampled continuously. Consequently, the sinusoidal initial condition generally has unequal endpoint values and is incompatible with pointwise periodic matching at $t = 0$. In the reported refinement setup, we prescribe the boundary traces of the reference trajectory,

$$u(-1, t) = g_-(t), \qquad u(1, t) = g_+(t), \qquad g_\pm(t) = u_{\mathrm{ref}}(\pm 1, t).$$

Boundary and initial-condition losses use the corresponding stored traces; periodic endpoint equality is not imposed. These boundary and initial traces are supplied as problem data during online refinement; interior reference values are used only for evaluation.

The parameters A and k control the amplitude and spatial frequency of the initial wave, respectively, while ν governs the strength of viscous dissipation. This setting is often used in parameterized PDE benchmarks because the nonlinear convection term $u\,u_x$ interacts with viscosity to produce rich dynamics ranging from steep gradients and shock-like structures (low ν) to smooth diffusive behavior (high ν).

Convection-only (1 + 1) D

As a second benchmark, a parameterized one-dimensional convection-only equation is considered:

$$\frac{\partial u}{\partial t} + \beta\frac{\partial u}{\partial x} = 0, \quad x \in [0, L],\ t \in [0, 1], \tag{17}$$

subject to periodic boundary conditions

$$u(x = 0, t) = u(x = L, t), \quad u(x, 0) = 1 + \sin\left(\frac{2\pi x}{L}\right). \tag{18}$$

The state variable u now evolves solely under the effect of convection. The parameter β controls the transport speed, with $\beta \in [0.92, 29.8]$ and $L = 2\pi$.

Helmholtz (2D)

The Helmholtz equation is considered in two spatial dimensions with parameterized source terms and sinusoidal exact solutions. In general form,

$$\nabla^2 u + k^2 u = q(\mathbf{x}), \quad \mathbf{x} \in [-1, 1]^2, \tag{19}$$

with generally nonhomogeneous Dirichlet boundary conditions prescribed by the manufactured exact solution (20):

$$u(\mathbf{x}) = g_{a_1,a_2}(\mathbf{x}) = \sin(a_1 \pi x_1) \sin(a_2 \pi x_2), \qquad \mathbf{x} \in \partial\Omega.$$

The exact solution is given by a separable sinusoidal function,

$$u(\mathbf{x}) = \sin(a_1 \pi x_1) \sin(a_2 \pi x_2), \tag{20}$$

and the corresponding source term is

$$q(\mathbf{x}) = \left(-(a_1\pi)^2 - (a_2\pi)^2 + k^2\right) \sin(a_1 \pi x_1) \sin(a_2 \pi x_2). \tag{21}$$

The parameter set is (a_1, a_2, k) with $a_1 \in [0.53, 4.97]$, $a_2 \in [0.56, 5.00]$, and $k \in [0.52, 4.99]$. The frequency parameters a_1 and a_2 are sampled continuously, without restriction to integers. Within these positive parameter ranges, the boundary condition is homogeneous only when both a_1 and a_2 are integers.

This benchmark is analytically tractable via a manufactured solution approach, with closed-form expressions for both the solution and the forcing term. Parameter variation induces diverse oscillatory behaviors, making the problem a controlled yet challenging test case for evaluating generalization in parameterized PDE spaces.

E.3 Flow Mixing $(2+1)$D

The Flow Mixing extension transports a scalar field in a spatially varying rotational flow on $\Omega = [-4, 4]^2$ over $t \in [0, 4]$:

$$\frac{\partial u}{\partial t} + \omega_\alpha(r) \left(-y \frac{\partial u}{\partial x} + x \frac{\partial u}{\partial y} \right) = 0, \qquad r = \sqrt{x^2 + y^2}, \tag{22}$$

where the angular velocity is

$$\omega_\alpha(r) = \frac{2\alpha}{0.385} \frac{\operatorname{sech}^2(r) \tanh(r)}{r}, \qquad \omega_\alpha(0) = \frac{2\alpha}{0.385}. \tag{23}$$

The constant 0.385 is the velocity normalization used in the implementation; the value at the origin is defined by continuity. The scalar α controls the transport rate, while the spatial domain and initial profile remain fixed. The initial condition is

$$u(x, y, 0) = -\tanh(y/2). \tag{24}$$

Since the radius is constant along characteristics, the analytic reference is

$$u_{\mathrm{ref}}(x, y, t; \alpha) = -\tanh\left(\frac{y \cos(\omega_\alpha(r) t) - x \sin(\omega_\alpha(r) t)}{2} \right). \tag{25}$$

Dirichlet values on all four spatial edges are prescribed by its trace, $u|_{\partial\Omega} = u_{\mathrm{ref}}|_{\partial\Omega}$, for $t \in [0, 4]$. Relative-L^2 errors are evaluated against this continuous analytic reference on a uniform $64 \times 64 \times 64$ space–time grid.

The experiment uses ten offline training conditions spanning $\alpha \in [0.1361, 0.8463]$ and five held-out conditions $\alpha \approx \{0.3857, 0.5295, 0.5412, 0.7850, 0.8560\}$. During online refinement, each update resamples 2,048 residual points uniformly in $\Omega \times [0, 4]$, 512 initial points uniformly in Ω at $t = 0$, and 512 boundary points with uniformly sampled edges, edge coordinates, and times. The residual, initial-condition, and boundary mean-squared losses have weights 1, 50, and 10, respectively. The evaluation grid is used to measure error; refinement uses the sampled physics points and prescribed initial and boundary values. Initializer comparisons and optimization trajectories are reported in Appendix C.1.