RANDOM RESAMPLING OF COLLOCATION POINTS CAN AMPLIFY SPECTRAL BIAS IN PINNS

Anonymous authors
Paper under double-blind review

ABSTRACT

Random resampling of collocation points is standard practice in physics-informed neural network (PINN) training, and is widely expected to be especially beneficial for multi-scale problems where resolving fine-scale structure seems to demand sufficient coverage of the domain. We show that this expectation is misleading. Across a range of multi-scale PINN benchmarks, random resampling rapidly recovers the macro-scale solution but substantially delays high-frequency recovery. In contrast, fixed sampling recovers high-frequency components earlier, but introduces global distortion through collocation bias. This delay coincides with the regime where the variance of the residual gradient under resampling exceeds its mean magnitude. Guided by this diagnosis, we introduce spatial and temporal interpolations between fixed and random resampling that consistently reduce the delay in fine-scale recovery. These findings suggest that spectral bias in PINNs is not only a consequence of sparse collocation, but also a phenomenon shaped by how collocation points are resampled during training.

1 INTRODUCTION

Multi-scale phenomena pervade scientific and engineering disciplines, from geomechanics (Bear, 1988; Haghighat et al., 2021) and turbulence modeling (Pope, 2000; Ling et al., 2016; Kochkov et al., 2021) to climate science (Schneider et al., 2017; Kashinath et al., 2021) and nano-scale thermal transport (Cahill et al., 2003; Guo and Wang, 2015). In each setting, smooth macroscopic structures couple with rapidly oscillatory microscopic features, and resolving this scale separation is essential for predictive fidelity (E and Engquist, 2003). The challenge demands learning algorithms that simultaneously capture both low- and high-frequency solution components.

Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019) offer a compelling approach, embedding PDE constraints directly into the training objective to enable mesh-free treatment of forward and inverse problems. Yet PINNs systematically fail on multi-scale problems, recovering low-frequency components while leaving fine-scale structures unresolved (Wang et al., 2021). This failure is commonly attributed to *spectral bias*, preferential learning of low-frequency components in early training (Rahaman et al., 2019; Xu et al., 2020), and has motivated frequency-aware network designs including Fourier feature embeddings (Tancik et al., 2020) and sinusoidal activations (Sitzmann et al., 2020). These approaches demonstrate that architectural choices can substantially mitigate spectral bias, but they do not address whether common training practices can further shape, or even amplify, spectral bias in PINNs.

In this work, we identify a previously overlooked source of spectral bias amplification in PINNs: stochastic gradient noise induced by random resampling of collocation points. Random collocation resampling is widely adopted in PINN training (Lu et al., 2021; Wu et al., 2023), as it provides unbiased estimation of the PDE residual loss without requiring a fixed mesh. We show, however, that this stochasticity disproportionately corrupts the gradient signal for high-frequency residual components, substantially delaying their recovery. Fixed sampling removes this variance but introduces a persistent collocation bias that distorts the overall solution shape. Collocation sampling in PINNs therefore presents a bias-variance trade-off in which the two endpoints fail in qualitatively different ways: random resampling preserves the macro-scale solution but stalls fine-scale recovery, while fixed sampling captures fine-scale structure earlier but at the cost of global distortion. Figure 1 illustrates this contrast across five multi-scale benchmarks.

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.

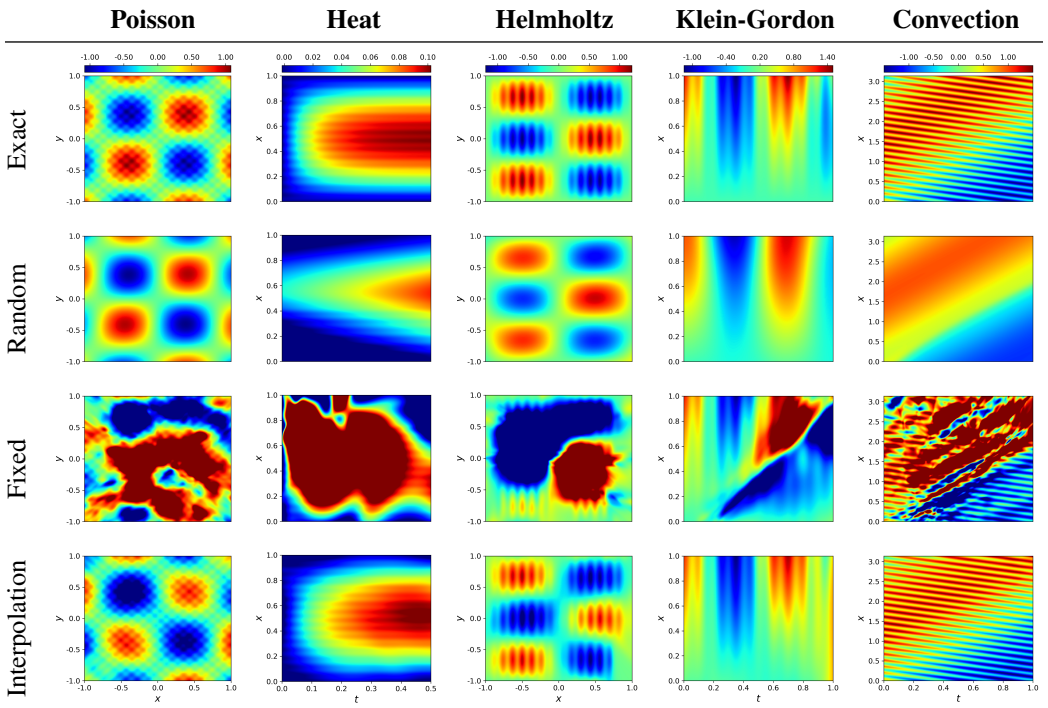


Figure 1: Learned solutions on multi-scale equations. Random resampling resolves only the macro-scale structure, fixed sampling distorts the global solution shape, and interpolation between fixed and random sampling recovers both global and fine-scale structure. Detailed benchmark equations and training settings are provided in Appendix A.

Our analysis reveals that the high-frequency delay under random resampling coincides with a regime in which the variance of the residual gradient estimator dominates its mean, which we refer to as the *noise-dominated regime*. High-frequency errors remain nearly stationary throughout this regime and begin to decay only as it subsides. To examine this at the level of individual modes, we project the residual gradient signal-to-noise ratio onto modal coefficient directions and find that high-frequency modes exhibit the strongest and most persistent noise dominance. This diagnosis suggests how to balance the bias-variance trade-off: rather than choosing between fixed and random sampling, one can interpolate between them. We study spatial and temporal interpolation schemes that partially reuse collocation points, either within each collocation set or across training iterations, and show that they consistently reduce fine-scale delay without inheriting the distortions of fully fixed collocation.

2 Preliminaries

2.1 Physics-Informed Neural Networks

Let $\Omega \subseteq \mathbb{R}^d$ be a spatial domain with boundary $\partial\Omega$, and let $[0, T]$ be the time interval of interest. We consider a general time-dependent boundary value problem of the form

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

where $\mathcal{N}$ denotes a differential operator in $(t, \boldsymbol{x})$, $\mathcal{B}$ encodes the boundary condition, and $\mathcal{I}$ represents the initial condition at $t = 0$.

We approximate the solution u by a neural network $u(t, \boldsymbol{x}; \theta)$ with trainable parameters $\theta \in \mathbb{R}^P$, where P is the number of model parameters. To streamline the notation, we introduce the space–time

domains $\mathcal{D}_r := [0,T] \times \Omega,\ \mathcal{D}_b := [0,T] \times \partial\Omega,\ \mathcal{D}_i := \Omega$. We define the pointwise loss densities

$$\begin{aligned}
\ell_r(t,\boldsymbol{x};\theta) &:= \left|\mathcal{N}[u(\cdot,\cdot;\theta)](t,\boldsymbol{x}) - f(t,\boldsymbol{x})\right|^2, && (t,\boldsymbol{x}) \in \mathcal{D}_r,\\
\ell_b(t,\boldsymbol{x};\theta) &:= \left|\mathcal{B}[u(\cdot,\cdot;\theta)](t,\boldsymbol{x}) - g(t,\boldsymbol{x})\right|^2, && (t,\boldsymbol{x}) \in \mathcal{D}_b,\\
\ell_i(\boldsymbol{x};\theta) &:= \left|\mathcal{I}[u(\cdot,\cdot;\theta)](\boldsymbol{x}) - u(0,\boldsymbol{x})\right|^2, && \boldsymbol{x} \in \mathcal{D}_i.
\end{aligned}$$

The corresponding population losses are defined as normalized integrals over their respective domains:

$$\mathcal{L}_\alpha(\theta) := \frac{1}{|\mathcal{D}_\alpha|}\int_{\mathcal{D}_\alpha} \ell_\alpha(\boldsymbol{z};\theta)\,\mathrm{d}\boldsymbol{z}, \qquad \alpha \in \{r,b,i\}. \tag{2}$$

The variable $\boldsymbol{z}$ denotes the generic coordinate on $\mathcal{D}_\alpha$. Given non-negative weights $\omega_r, \omega_b, \omega_i \geq 0$, the total population loss is defined as $\mathcal{L}(\theta) := \omega_r\mathcal{L}_r(\theta) + \omega_b\mathcal{L}_b(\theta) + \omega_i\mathcal{L}_i(\theta)$.

In practice, the integrals in eq. (2) are approximated by Monte Carlo sums over collocation points. Let $\mathcal{X}_r^{(k)} = \{(t_r^{n,(k)}, \boldsymbol{x}_r^{n,(k)})\}_{n=1}^{N_r} \subset \mathcal{D}_r,\ \mathcal{X}_b^{(k)} = \{(t_b^{n,(k)}, \boldsymbol{x}_b^{n,(k)})\}_{n=1}^{N_b} \subset \mathcal{D}_b,\ \mathcal{X}_i^{(k)} = \{\boldsymbol{x}_i^{n,(k)}\}_{n=1}^{N_i} \subset \mathcal{D}_i$ denote the PDE, boundary, and initial collocation sets used at iteration k.

The empirical losses at iteration k are defined as

$$\widehat{\mathcal{L}}_\alpha(\theta_k) := \frac{1}{N_\alpha}\sum_{\boldsymbol{z}\in\mathcal{X}_\alpha^{(k)}} \ell_\alpha(\boldsymbol{z};\theta_k), \qquad \alpha \in \{r,b,i\}, \tag{3}$$

where $N_\alpha := |\mathcal{X}_\alpha^{(k)}|$. Then the empirical total loss at iteration k is $\widehat{\mathcal{L}}(\theta_k) := \omega_r\widehat{\mathcal{L}}_r(\theta_k) + \omega_b\widehat{\mathcal{L}}_b(\theta_k) + \omega_i\widehat{\mathcal{L}}_i(\theta_k)$, where θ are updated via gradient-based optimizers such as Adam (Kingma and Ba, 2015) or L-BFGS (Liu and Nocedal, 1989).

2.2 Collocation Sampling in PINNs

Recall the PDE residual loss density $\ell_r(t,\boldsymbol{x};\theta)$ and the corresponding population loss $\mathcal{L}_r(\theta)$ defined in eq. (2). For notational convenience, let $\boldsymbol{z} := (t,\boldsymbol{x}) \in \mathcal{D}_r$ and write $\ell_r(\boldsymbol{z};\theta) := \ell_r(t,\boldsymbol{x};\theta)$. Let p_r denote the uniform distribution on $\mathcal{D}_r$. Then the population PDE residual loss and its gradient can be written as

$$\mathcal{L}_r(\theta) = \mathbb{E}_{\boldsymbol{z}\sim p_r}[\ell_r(\boldsymbol{z};\theta)], \qquad g_r(\theta) := \nabla_\theta\mathcal{L}_r(\theta) = \mathbb{E}_{\boldsymbol{z}\sim p_r}[\nabla_\theta\ell_r(\boldsymbol{z};\theta)]. \tag{4}$$

Given any finite collocation set $\mathcal{X}_r = \{\boldsymbol{z}_n\}_{n=1}^{N_r}$, define the empirical residual loss

$$\widehat{\mathcal{L}}_r(\theta;\mathcal{X}_r) := \frac{1}{N_r}\sum_{n=1}^{N_r} \ell_r(\boldsymbol{z}_n;\theta), \tag{5}$$

and its gradient

$$\widehat{g}_r(\theta;\mathcal{X}_r) := \nabla_\theta\widehat{\mathcal{L}}_r(\theta;\mathcal{X}_r) = \frac{1}{N_r}\sum_{n=1}^{N_r} \nabla_\theta\ell_r(\boldsymbol{z}_n;\theta). \tag{6}$$

When the underlying collocation set is clear from context, we simply write $\widehat{g}_r(\theta)$ for $\widehat{g}_r(\theta;\mathcal{X}_r)$.

In the original formulation of PINNs (Raissi et al., 2019), the collocation set is typically sampled once and reused throughout training. Although this choice leads to deterministic optimization, it can be sensitive to the particular collocation draw, especially when the collocation budget is limited. For this reason, many practical implementations of PINN (Wang et al., 2021; Wu et al., 2023; Wang et al., 2022; Cho et al., 2023; Hwang and Lim, 2024) instead redraw collocation points during training. We refer to these two approaches as *fixed sampling* and *random resampling*, respectively, and formalize them below. To isolate the effect of residual collocation sampling, boundary and initial collocation points, when present, are resampled at every iteration for all methods throughout this work.

Fixed sampling. A single collocation set $\mathcal{X}_r$ is drawn once at the beginning of training and then reused throughout optimization. Conditional on this draw, the update direction at iteration k is deterministic:

$$\widehat{g}_r(\theta_k;\mathcal{X}_r) = \nabla_\theta\widehat{\mathcal{L}}_r(\theta_k;\mathcal{X}_r), \qquad \mathrm{Cov}(\widehat{g}_r(\theta_k;\mathcal{X}_r)) = \mathbf{0}.$$

Thus, fixed sampling eliminates per-iteration gradient noise. However, it optimizes the empirical objective $\widehat{\mathcal{L}}_r(\theta; \mathcal{X}_r)$ rather than the population objective $\mathcal{L}_r(\theta)$, and therefore generally introduces a persistent collocation bias.

Random resampling. At each iteration k, a fresh collocation set $\mathcal{X}_r^{(k)} = \{\boldsymbol{z}_r^{n,(k)}\}_{n=1}^{N_r}$ is drawn independently from p_r. This yields an unbiased estimator of the population residual gradient, i.e., $\mathbb{E}[\widehat{g}_r(\theta_k; \mathcal{X}_r^{(k)})] = g_r(\theta_k)$. However, unlike fixed sampling, the update direction contains stochastic gradient noise whose covariance is

$$\Sigma_r(\theta_k) := \mathrm{Cov}\Big(\widehat{g}_r(\theta_k; \mathcal{X}_r^{(k)})\Big) = \frac{1}{N_r}\Big(\mathbb{E}_{\boldsymbol{z}\sim p_r}\big[\nabla_\theta \ell_r(\boldsymbol{z};\theta_k)\nabla_\theta \ell_r(\boldsymbol{z};\theta_k)^\top\big] - g_r(\theta_k)g_r(\theta_k)^\top\Big). \quad (7)$$

Therefore, random resampling removes collocation bias in expectation, but injects nonzero stochastic gradient noise at every iteration.

3 Random Resampling Amplifies Spectral Bias in PINN Training

We begin by examining whether random resampling, despite improving domain coverage, helps or hinders spectral learning in PINNs. We show that it can substantially delay fine-scale recovery.

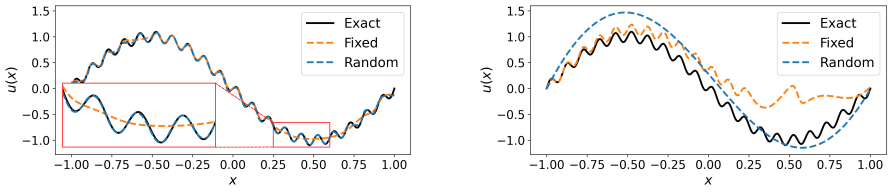
3.1 Contrasting Effects of Random Resampling in Regression and PINNs

In multi-scale learning, a natural intuition is that poor recovery of fine-scale structure often stems from insufficient coverage of the training data over the domain: if the training points do not adequately resolve rapidly oscillating components, one would expect aliasing or failure to capture fine-scale patterns. From this perspective, random resampling should be beneficial, as exposing the model to more diverse input locations across iterations should improve the recovery of multi-scale patterns.

To examine this intuition, we consider a one-dimensional Poisson problem

$$-u''(x) = f(x), \qquad u^\star(x) = \sin\big(\pi(x+1)\big) + 0.1\,\sin\big(20\pi(x+1)\big), \quad (8)$$

on $x \in (-1, 1)$, with homogeneous Dirichlet boundary conditions at both endpoints. The forcing term $f(x)$ is computed analytically from the exact solution $u^\star(x)$. Both regression and PINNs use the same fully connected network with three hidden layers of width 128 and tanh activations. Additional experimental details are provided in Appendix C.2.



(a) Regression (b) PINNs

Figure 2: Regression and PINNs under fixed and random resampling for a 1D Poisson problem.

Figure 2 compares fixed and random resampling on the same multi-scale target, with direct regression in Figure 2(a) and PINNs in Figure 2(b). In regression, random resampling better recovers the oscillatory structure, whereas fixed sampling misses the fine-scale component over part of the domain. This is consistent with the usual intuition that more diverse training points improve the recovery of multi-scale patterns. In PINNs, however, the same intuition does not hold. As shown in Figure 2(b), random resampling fails to recover the high-frequency component, while fixed sampling captures it. At the same time, fixed sampling does not preserve the overall solution shape as well, yielding a distorted global profile. The sampling strategy therefore plays a qualitatively different role in PINN training than in regression, and can itself shape spectral bias.

3.2 Mode-Wise Learning Dynamics under Collocation Sampling

Beyond the illustrative comparison in Figure 2, Figure 1 suggests that a similar contrast persists across diverse equations, with random resampling tending to recover the macro-scale structure while

fixed sampling often distorts the overall solution. To study this phenomenon systematically, we consider the following two-dimensional multi-scale Poisson problem:

$$-\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega, \qquad u(\boldsymbol{x}) = g(\boldsymbol{x}), \qquad \boldsymbol{x} \in \partial\Omega. \tag{9}$$

We use the two mode manufactured solution

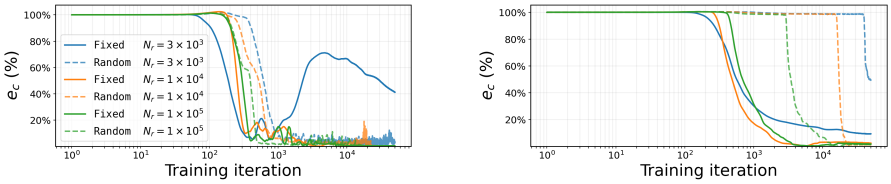
$$u^\star(\boldsymbol{x}) = \sin(c_1 x)\sin(c_1 y) + C\,\sin(c_2 x)\sin(c_2 y), \qquad \boldsymbol{x} = (x, y) \in \Omega, \tag{10}$$

where c_1 and c_2 parameterize the low- and high-frequency modes, respectively, and f and g are defined so that $u^\star$ satisfies eq. (9) exactly. We choose $c_1 = 4$, $c_2 = 40$, $C = 0.1$, $\Omega = [-1, 1]^2$.

To quantify mode-wise convergence, we project the predicted solution onto the target modes in eq. (10). For each frequency $c \in \{c_1, c_2\}$, we define the basis function $\phi_c(\boldsymbol{x}) := \sin(cx)\sin(cy)$ and its projection coefficient on a fixed evaluation grid as

$$\widehat{a}_c := \frac{\langle u(\cdot;\theta), \phi_c\rangle}{\langle \phi_c, \phi_c\rangle}, \qquad e_c := \frac{|\widehat{a}_c - a_c^\star|}{|a_c^\star|}, \qquad c \in \{c_1, c_2\}, \tag{11}$$

where $a_c^\star$ denotes the exact coefficient of ϕ_c in $u^\star$.



(a) $c = 4$ (b) $c = 40$

Figure 3: Mode-wise coefficient error for the Poisson problem ($c = 4$ and $c = 40$) under varying numbers of collocation points N_r.

Figure 3 examines how the number of collocation points N_r affects mode-wise convergence. For the low-frequency mode ($c = 4$, Figure 3a), random resampling converges reliably across all values of N_r, whereas fixed sampling fails to converge when N_r is small, suggesting that a sparse fixed grid introduces collocation bias severe enough to corrupt even the low-frequency component. Otherwise, for the high-frequency mode ($c = 40$, Figure 3b), fixed sampling converges across all N_r, whereas random resampling exhibits substantially delayed convergence that persists even as N_r increases.

These results indicate that the difference between random and fixed sampling cannot be explained solely by the spatial coverage of collocation points. In particular, for the high-frequency mode, fixed sampling can recover the target substantially earlier even with an order of magnitude fewer collocation points than random resampling.

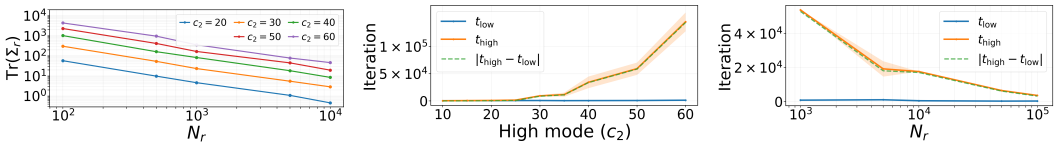
3.3 Effects of Frequency and Collocation Budget on Mode-Wise Convergence

We next examine whether changes that affect the stochastic residual gradient noise also change the delay in high-frequency learning. For a mode c, let e_c denote the relative coefficient error defined in eq. (11) at iteration k. We define the capture time of mode c at threshold τ as

$$t_c := \min\{k \geq 0 : e_c(\theta_k) < \tau\}. \tag{12}$$

For the Poisson problem, we write $t_{\text{low}} := t_{c_1}$ and $t_{\text{high}} := t_{c_2}$. We set $\tau = 0.1$ and use $\text{gap} := |t_{\text{high}} - t_{\text{low}}|$ to measure the difference between low- and high-frequency recovery.

We vary the high-frequency index c_2 and the residual collocation budget N_r under random resampling. Figure 4a shows that $\text{Tr}(\Sigma_r(\theta))$ is larger for higher c_2 and decreases as N_r increases. Figure 16 visualizes the target solutions for varying c_2. The corresponding capture times in Figures 4b and 4c show that increasing c_2 or decreasing N_r primarily delays t_{high}, while t_{low} remains comparatively stable. Thus, the low–high capture gap widens in the same regimes where the residual gradient estimator exhibits larger fluctuations. This selective delay suggests that the observed amplification of spectral bias is closely associated with stochastic gradient noise induced by random resampling. We further conduct a controlled noise injection experiment in Appendix G.3, where the collocation set and the target solution are held fixed and only the magnitude of the injected gradient noise is varied. The gap widens as the noise grows, which indicates that gradient noise by itself can delay high-frequency recovery.



(a) Trace of $\Sigma_r(\theta)$ (b) Varying high-frequency mode c_2 (c) Varying collocation budget N_r

Figure 4: Gradient noise and mode-wise capture times under random resampling on the two-mode Poisson equation. (a) Trace of the stochastic residual gradient covariance. (b) Capture times under varying high-frequency index c_2. (c) Capture times under varying collocation budget N_r.

4 How Resampling Noise Amplifies Spectral Bias

In this section, we connect delayed fine-scale recovery to resampling variance. We analyze the balance between the mean residual gradient signal and stochastic gradient noise, showing that high-frequency learning stalls when the latter dominates.

4.1 Sobolev Seminorm Error for Multi-Scale Evaluation

While the relative L^2 error e_{L^2} is commonly used to evaluate PINNs, it primarily measures global solution accuracy and can remain small even when high-frequency components are poorly recovered. To capture such errors, we also report the relative H^1 seminorm error e_{H^1}, which measures the magnitude of the gradient discrepancy and is therefore sensitive to local oscillatory structure. Formal definitions of both metrics are provided in Appendix B.

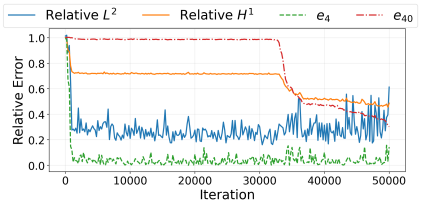
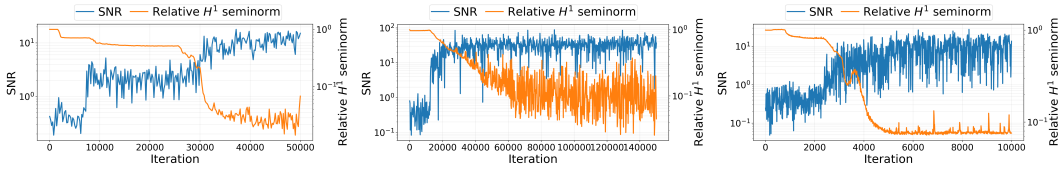


Figure 5: Relative L^2, relative H^1 seminorm, and mode-wise coefficient errors on the Poisson problem.

Figure 5 highlights the difference between these two metrics. The relative L^2 error decreases rapidly during the early stage of training, suggesting an apparent improvement in overall accuracy. In contrast, the relative H^1 seminorm error remains large until the high-frequency error e_{40} starts to decrease. This behavior indicates that the H^1 seminorm error is more sensitive to delayed recovery of high-frequency components.



(a) Darcy (b) Convection (c) Klein–Gordon

Figure 6: Residual gradient SNR and relative H^1 seminorm error during training for each PDE benchmark.

4.2 Residual Gradient Signal-to-Noise Ratio as a Diagnostic

We now ask whether the high-frequency delay under random resampling is accompanied by a weak residual gradient signal relative to the stochastic noise induced by collocation resampling. To quantify this balance, we compare the strength of the population residual gradient with the total variance of its stochastic estimator. Following signal-to-noise (SNR) diagnostics used for stochastic gradient estimators and neural network training (Roberts and Tedrake, 2008; Liu et al., 2020; Jiang et al., 2023; Anagnostopoulos et al., 2025), we define the residual gradient SNR as

$$\mathrm{SNR}(\theta_k) := \frac{\|g_r(\theta_k)\|_2}{\sqrt{\mathrm{Tr}(\Sigma_r(\theta_k))}}. \tag{13}$$

Here, $g_r(\theta_k)$ denotes the population residual gradient and $\Sigma_r(\theta_k)$ denotes the covariance of the stochastic residual gradient estimator under random resampling. A large value of $\mathrm{SNR}(\theta_k)$ indicates

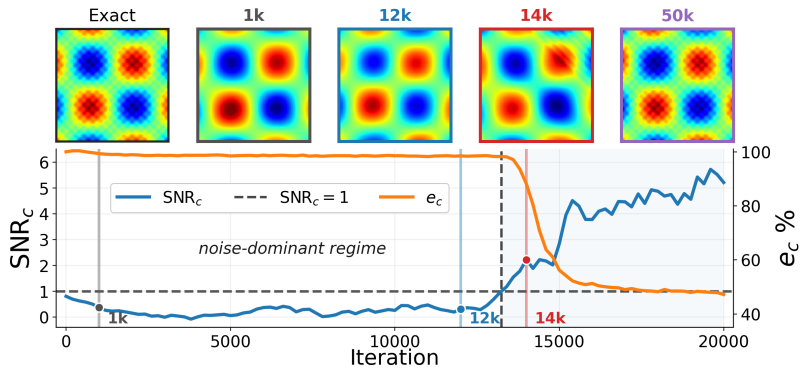


Figure 7: Mode-wise SNR and high-frequency coefficient error on the Poisson problem for $c = 40$. The error e_c remains nearly flat in the noise-dominated regime $\mathrm{SNR}_c < 1$, and begins to decay once SNR_c exceeds this threshold. Top snapshots show the learned solution at selected iterations.

that the deterministic residual gradient signal is large relative to the resampling variance, whereas a small value indicates that the stochastic noise is comparable to, or larger than, the deterministic signal. Detailed implementation of the SNR estimation is provided in Appendix C.3.

Figure 6 compares residual gradient SNR and relative H^1 seminorm error across benchmarks. Low SNR coincides with slow H^1 decay, whereas rising SNR aligns with faster error reduction, especially as high-frequency components begin recovering. These results indicate that the delayed improvement in the H^1 error is closely associated with the noise level of the residual gradient estimator under random resampling, and that such noise may be particularly detrimental to fine-scale learning.

The SNR in eq. (13), however, cannot determine whether the corrective signal for a specific spectral component is itself dominated by stochastic noise, because it aggregates signal and noise over all parameter-space components. We therefore next refine this diagnostic at the level of individual modal errors.

4.3 Mode-Wise SNR for Diagnosing High-Frequency Delay

To examine whether the resampling noise identified in Section 4 affects high-frequency learning more strongly than low-frequency learning, we return to the two-mode Poisson equation in Section 3.2. Using the modal coefficient $\widehat{a}_c(\theta)$ and coefficient error $e_c(\theta)$ in eq. (11), we introduce the signed coefficient error and its normalized squared error:

$$\delta_c(\theta) := \frac{\widehat{a}_c(\theta) - a_c^\star}{|a_c^\star|}, \qquad \mathcal{E}_c(\theta) := \frac{1}{2}\delta_c(\theta)^2.$$

Let $v_c(\theta) := \nabla_\theta \mathcal{E}_c(\theta)$. For a randomly resampled residual batch $\mathcal{X}_r$, define the first-order decrease of $\mathcal{E}_c$ induced by the stochastic residual gradient as $Y_c(\theta; \mathcal{X}_r) := v_c(\theta)^\top \widehat{g}_r(\theta; \mathcal{X}_r)$, which we refer to as the *linearized descent signal* for mode c. A first-order Taylor expansion then gives

$$\mathcal{E}_c(\theta - \eta\, \widehat{g}_r(\theta; \mathcal{X}_r)) = \mathcal{E}_c(\theta) - \eta\, Y_c(\theta; \mathcal{X}_r) + O(\eta^2), \tag{14}$$

implying that positive values of Y_c reduce the modal error to first order, whereas negative values increase it. Under random resampling, Y_c has mean $v_c(\theta)^\top g_r(\theta)$ and variance $v_c(\theta)^\top \Sigma_r(\theta) v_c(\theta)$. We define the mode-wise SNR as

$$\mathrm{SNR}_c(\theta) := \frac{v_c(\theta)^\top g_r(\theta)}{\left(v_c(\theta)^\top \Sigma_r(\theta) v_c(\theta)\right)^{1/2}} = \mathrm{sign}(\delta_c(\theta)) \frac{J_c(\theta)^\top g_r(\theta)}{\left(J_c(\theta)^\top \Sigma_r(\theta) J_c(\theta)\right)^{1/2}}, \tag{15}$$

where $J_c(\theta) := \nabla_\theta \widehat{a}_c(\theta)$ and $v_c(\theta) = \nabla_\theta \mathcal{E}_c(\theta) = \delta_c(\theta) J_c(\theta)/|a_c^\star|$. The second form shows that SNR_c measures the signed residual gradient signal acting on the modal coefficient relative to the sampling variance seen by the same coefficient sensitivity.

When $\mathrm{SNR}_c < 1$, the mean linearized descent signal for mode c is smaller than the typical noise caused by resampling. In this regime, independent residual batches do not produce a stable correction of the modal error: the sampled update can be weakened, canceled, or reversed before consistent

Table 1: Comparison of collocation strategies across benchmark equations in terms of e_{H^1} and e_{L^2}. Results are reported as mean $\pm$ standard deviation over five random seeds. The **best** results are shown in bold, and the second-best results are underlined.

	Poisson		Heat		Helmholtz		Darcy		Klein–Gordon		Convection	
Method	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}
Random	$0.733_{\pm 0.018}$	$0.377_{\pm 0.109}$	$0.602_{\pm 0.028}$	$0.321_{\pm 0.073}$	$0.775_{\pm 0.004}$	$0.245_{\pm 0.030}$	$0.558_{\pm 0.063}$	$0.178_{\pm 0.073}$	$0.791_{\pm 0.123}$	$2.691_{\pm 1.091}$	$0.965_{\pm 0.053}$	$0.582_{\pm 0.164}$
Fixed	$0.498_{\pm 0.176}$	$0.586_{\pm 0.407}$	$\mathbf{0.299}_{\pm 0.088}$	$\mathbf{0.097}_{\pm 0.032}$	$1.812_{\pm 0.534}$	$2.735_{\pm 0.981}$	$0.655_{\pm 0.274}$	$0.939_{\pm 0.381}$	$1.487_{\pm 1.360}$	$4.396_{\pm 3.577}$	$3.267_{\pm 1.176}$	$2.140_{\pm 0.782}$
Importance	$0.640_{\pm 0.093}$	$0.251_{\pm 0.049}$	$0.650_{\pm 0.078}$	$0.349_{\pm 0.051}$	$0.773_{\pm 0.004}$	$0.298_{\pm 0.117}$	$0.196_{\pm 0.171}$	$0.177_{\pm 0.194}$	$0.650_{\pm 0.175}$	$2.372_{\pm 1.653}$	$0.990_{\pm 0.017}$	$0.484_{\pm 0.332}$
Spatial	$\mathbf{0.241}_{\pm 0.053}$	$\mathbf{0.219}_{\pm 0.088}$	$\underline{0.325}_{\pm 0.163}$	$\underline{0.147}_{\pm 0.070}$	$0.160_{\pm 0.055}$	$\underline{0.239}_{\pm 0.087}$	$0.150_{\pm 0.075}$	$0.118_{\pm 0.077}$	$\underline{0.474}_{\pm 0.192}$	$\mathbf{1.683}_{\pm 0.833}$	$\mathbf{0.188}_{\pm 0.089}$	$\mathbf{0.301}_{\pm 0.098}$
Temporal	$\mathbf{0.241}_{\pm 0.052}$	$\underline{0.221}_{\pm 0.096}$	$0.510_{\pm 0.091}$	$0.211_{\pm 0.049}$	$\mathbf{0.094}_{\pm 0.022}$	$\mathbf{0.187}_{\pm 0.101}$	$\mathbf{0.045}_{\pm 0.014}$	$\mathbf{0.037}_{\pm 0.038}$	$0.650_{\pm 0.315}$	$3.062_{\pm 2.070}$	$0.243_{\pm 0.120}$	$0.414_{\pm 0.061}$
Spatio-Temporal	$0.272_{\pm 0.034}$	$0.240_{\pm 0.057}$	$0.392_{\pm 0.115}$	$0.150_{\pm 0.066}$	$\underline{0.119}_{\pm 0.041}$	$0.258_{\pm 0.193}$	$\underline{0.064}_{\pm 0.005}$	$\underline{0.044}_{\pm 0.005}$	$\mathbf{0.377}_{\pm 0.358}$	$\underline{1.751}_{\pm 1.742}$	$\underline{0.234}_{\pm 0.206}$	$\underline{0.359}_{\pm 0.089}$

progress accumulates. If this noise-dominated regime persists for the high-frequency mode, its recovery can remain stalled even while lower-frequency components continue to improve.

As shown in Figure 7, SNR_c closely tracks the onset of high-frequency recovery for $c = 40$ in the Poisson problem. Across collocation budgets, e_c remains nearly flat while $\mathrm{SNR}_c < 1$, where resampling variance exceed the mean linearized descent signal and prevent stable modal correction. The coefficient begins to decay only after SNR_c rises above this regime over successive iterations. We describe empirical estimation of SNR_c in Appendix C.4. These results clarify how stochastic noise amplifies spectral bias in PINNs. When the gradient signal associated with a fine-scale mode is smaller than resampling variance, successive updates fail to accumulate reliable progress on that mode. Random resampling therefore keeps fine-scale learning in a noise-dominated regime, delaying high-frequency recovery beyond the intrinsic spectral bias of the network.

5 Balancing the Bias and Variance of Collocation Resampling

In this section, we first show that standard variance reduction is insufficient to address the amplification of spectral bias by random resampling. We then introduce interpolation schemes that bridge fixed sampling and random resampling by partially reusing collocation points, which improves multi-scale recovery.

5.1 Standard Variance Reduction Is Insufficient

A natural question is whether standard variance-reduction methods can remove the delay induced by random resampling. We examine importance sampling (Kahn and Marshall, 1953; Katharopoulos and Fleuret, 2018) and stochastic variance-reduced gradients (SVRG) (Johnson and Zhang, 2013; Reddi et al., 2016). Importance sampling uses residual magnitude as a practical proxy for gradient magnitude and is widely used in PINNs (Nabian et al., 2021; Yang et al., 2023; Zhang et al., 2025). SVRG uses a periodically computed large-batch reference gradient to correct stochastic gradient estimates and reduce their variance. A detailed description and implementation of importance sampling and SVRG is provided in Appendix E.

As shown in Figure 8 in Appendix E, importance sampling provides only limited improvement and does not substantially accelerate the decay of e_{40}. SVRG, in contrast, exhibits unstable behavior, suggesting that its reference-gradient correction can be unreliable when the PDE residual field changes rapidly during training. These results indicate that standard variance-reduction methods alone are insufficient to resolve the spectral bias in PINNs.

5.2 Interpolation Between Fixed and Random Resampling

Spatial interpolation. A straightforward way to interpolate between fixed and random sampling is to partition the collocation set into a fixed subset and a randomly refreshed subset. Specifically, a fraction $1 - \rho$ of the collocation points is sampled once and kept fixed throughout training, while the remaining fraction ρ is resampled at each iteration. Thus, ρ parametrizes a continuum from fixed sampling ($\rho = 0$) to standard random resampling ($\rho = 1$). Figure 18b in Appendix G.6 shows that high frequency recovery accelerates as ρ decreases, and even a small persistent subset with $\rho = 0.9$ already shortens the delay.

Temporal interpolation. Another simple interpolation strategy is to control how often the collocation set is refreshed during training. A common instance is periodic random resampling, in which the

collocation set is refreshed once every M iterations and reused in between. More generally, one may view temporal interpolation in terms of a resampling probability p, where the collocation set is refreshed at each iteration with probability p and otherwise reused from the previous step. Under this view, $p = 1$ recovers standard random resampling, $p = 0$ recovers fixed sampling, and intermediate values yield probabilistic reuse across iterations. Figure 19b in Appendix G.6 shows that smaller values of p can accelerate high-frequency recovery.

Spatio-temporal interpolation. Finally, the spatial and temporal interpolation strategies can be combined. Specifically, a fraction $1 - \rho$ of the collocation set is kept fixed throughout training, while the remaining fraction ρ is refreshed with probability p at each iteration.

Empirical results. Table 1 compares the collocation sampling strategies across six multi-scale benchmark equations using the final relative H^1 seminorm error and relative L^2 error. Detailed PDE formulations and training settings are provided in Appendix C.5. Interpolation strategies achieve the lowest errors on all benchmarks except the heat equation, where fixed sampling performs best but remains unreliable elsewhere, suggesting that a static collocation set can introduce persistent bias. We further consider an adaptive interpolation schedule that adjusts the resampling fraction during training using the residual gradient SNR, with details and results provided in Appendix G.7.

6 Related Work and Discussion

Multi-scale PINNs and spectral bias. Multi-scale PDEs are a well-known challenge for PINNs, since fine-scale structures are often much harder to resolve than smooth global components. This difficulty is closely related to spectral bias, the tendency of neural networks to learn low-frequency components before high-frequency ones (Rahaman et al., 2019; Xu et al., 2020). A number of prior works have therefore sought to improve PINNs for multi-scale problems through frequency-aware architectures and related design choices, including Fourier feature embeddings (Tancik et al., 2020) and periodic activations (Sitzmann et al., 2020). Because Fourier feature embeddings are a representative architectural remedy for spectral bias, we further examine resampling effects under Fourier feature embeddings in Appendix F.

Collocation resampling methods. The choice of collocation points is a fundamental component of PINN training. The original PINN formulation samples collocation points to approximate the PDE residual loss, while many modern implementations redraw or adapt these points during training (Raissi et al., 2019; Lu et al., 2021). A large body of work has studied non-adaptive and adaptive sampling strategies for PINNs, including residual-based adaptive refinement, importance sampling, quasi-random sequences such as Sobol sampling for improved spatial coverage, and mesh-based sampling schemes (Wu et al., 2023; Nabian et al., 2021; Yang et al., 2023; Daw et al., 2023). Some of these methods, such as R3 (Daw et al., 2023) and LAS (Jeong et al., 2025), implicitly reuse collocation points across iterations but do not control the degree of reuse explicitly. We discuss this further in Appendix G.8, where we combine our interpolation schemes with existing sampling methods and show that explicit control of collocation reuse improves them on most benchmarks.

7 Conclusion and Limitations

In this work, we revisited random resampling of collocation points in PINNs for multi scale PDEs. Across several multi-scale PDEs, random resampling tends to recover the macro-scale solution while delaying high-frequency recovery, whereas fixed sampling often recovers fine-scale components earlier but can distort the global solution. These results carry two implications. First, spectral bias in PINNs is not solely a property of network architecture: an additional component arises from the gradient noise introduced by random collocation resampling, which architectural remedies such as Fourier feature embeddings do not fully resolve, as shown in Appendix F. Second, collocation sampling is more naturally framed as a bias-variance trade-off than as a search for better spatial coverage, with random and fixed sampling as its variance and bias endpoints.

Limitations and Future Directions. A rigorous theoretical characterization of how gradient variance arising from collocation resampling shapes spectral learning dynamics remains an important open problem. Formalizing this mechanism could yield principled guidance on when and how to resample in PINN training. How this phenomenon manifests under more complex PDEs, boundary conditions, geometries, and architectures warrants further investigation.

AI USE STATEMENT

In this work, we used generative AI tools to assist in implementing methods, specifically in writing code for the collocation sampling strategies and the training and evaluation pipelines. We have not used generative AI tools for generating synthetic data sets, developing theoretical models or conceptual frameworks, formulating or proving mathematical claims, proposing or refining hypotheses, designing research methodology or experiments, cleaning or reformatting data sets, or interpreting results, and translation and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools for creating and editing software code, editing the paper to improve readability. We have reviewed all AI-assisted work. The authors reviewed and tested all code written with AI assistance, and all reported results were produced by running this verified code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide source code as supplementary material that implements collocation sampling strategies and reproduces the main results in Table 1 for all benchmark equations. The appendix further details the benchmark equations, training settings, evaluation metrics, and estimation procedures for all experiments.

REFERENCES

Jacob Bear. *Dynamics of Fluids in Porous Media*. Dover Publications, 1988.

Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021.

Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds-averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118 (21), 2021.

Tapio Schneider, Shiwei Lan, Andrew Stuart, and João Teixeira. Earth system modeling 2.0: A blueprint for models that learn from observations and targeted high-resolution simulations. *Geophysical Research Letters*, 44(24):12396–12417, 2017.

Karthik Kashinath, M. Mustafa, Adrian Albert, J. L. Wu, C. Jiang, Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, R. Wang, A. Chattopadhyay, A. Singh, et al. Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, 379(2194):20200093, 2021.

David G. Cahill et al. Nanoscale thermal transport. *Journal of Applied Physics*, 93(2):793–818, 2003.

Yangyu Guo and Moran Wang. Phonon hydrodynamics and its applications in nanoscale heat transport. *Physics Reports*, 595:1–44, 2015.

Weinan E and Bjorn Engquist. The heterogeneous multiscale methods. *Communications in Mathematical Sciences*, 1(1):87–132, 2003.

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.

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

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310. PMLR, 2019.

Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *Communications in Computational Physics*, 28(5):1746–1767, 2020.

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

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, volume 33, pages 7462–7473, 2020.

Lu Lu et al. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

Chenxi Wu et al. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large-scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

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

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

Youngsik Hwang and Dong-Young Lim. Dual cone gradient descent for training physics-informed neural networks. *Advances in Neural Information Processing Systems*, 37:98563–98595, 2024.

John W. Roberts and Russ Tedrake. Signal-to-Noise Ratio Analysis of Policy Gradient Algorithms. In *Advances in Neural Information Processing Systems*, volume 21, 2008.

Jinlong Liu, Yunzhi Bai, Guoqing Jiang, Ting Chen, and Huayan Wang. Understanding Why Neural Networks Generalize Well Through GSNR of Parameters. In *International Conference on Learning Representations*, 2020.

Guo-qing Jiang, Jinlong Liu, Zixiang Ding, Lin Guo, and Wei Lin. Accelerating Large Batch Training via Gradient Signal to Noise Ratio (GSNR). *arXiv preprint arXiv:2309.13681*, 2023.

Sokratis J Anagnostopoulos, Juan Diego Toscano, Nikolaos Stergiopulos, and George Em Karniadakis. Learning in PINNs: Phase transition, diffusion equilibrium, and generalization. *Neural Networks*, page 107983, 2025.

H. Kahn and A. W. Marshall. Methods of reducing sample size in Monte Carlo computations. *Operations Research*, 1(5):263–278, 1953.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2525–2534. PMLR, 2018.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, volume 26, pages 315–323, 2013.

Sashank J. Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 314–323. PMLR, 2016.

Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 36(8): 962–977, 2021.

Zijiang Yang, Zhongwei Qiu, and Dongmei Fu. DMIS: Dynamic mesh-based importance sampling for training physics-informed neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Zhengqi Zhang, Jing Li, and Bin Liu. Annealed adaptive importance sampling method in PINNs for solving high-dimensional partial differential equations. *Journal of Computational Physics*, 2025.

Arka Daw, Jie Bu, Sifan Wang, Paris Perdikaris, and Anuj Karpatne. Mitigating propagation failures in physics-informed neural networks using retain-resample-release (R3) sampling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7264–7302. PMLR, 2023.

Minseok Jeong, Giup Seo, and Euiseok Hwang. Mitigating instability in high residual adaptive sampling for PINNs via langevin dynamics. In *Advances in Neural Information Processing Systems*, 2025.

A BENCHMARK EQUATIONS

A.1 POISSON EQUATION

The Poisson equation is given by

$$\begin{aligned} -\Delta u(x,y) &= f(x,y), && (x,y) \in \Omega, \\ u(x,y) &= g(x,y), && (x,y) \in \partial\Omega, \\ \Omega &= [-1,1] \times [-1,1]. \end{aligned} \tag{16}$$

We use a manufactured multi-scale solution

$$u^\star(x,y) = \sin(c_1 x)\sin(c_1 y) + C\,\sin(c_2 x)\sin(c_2 y), \qquad (x,y) \in \Omega, \tag{17}$$

where $f(x,y), g(x,y)$ is derived by substituting $u^*(x,y)$ into the given equation. In this experiment, we choose parameters: $c_1 = 4, c_2 = 40, C = 0.1$.

We employ 3-hidden layers with 128 nodes per layer and use hyperbolic tangent function. We used Adam optimizer with learning rate $\lambda = 10^{-3}$, and set $(\omega_r, \omega_b) = (1, 100)$, $(N_r, N_b) = (2000, 100)$.

A.2 HELMHOLTZ EQUATION

The Helmholtz equation is described by

$$\begin{aligned} \Delta u(x,y) + k^2 u(x,y) &= f(x,y), && (x,y) \in \Omega, \\ u(x,y) &= 0, && (x,y) \in \partial\Omega, \\ \Omega &= [-1,1] \times [-1,1]. \end{aligned} \tag{18}$$

The solution is given by $u^*(x,y) = \sin(a_1\pi x)\cos(a_2\pi y)(1 + 0.2\sin(m\pi x))$, where $a_1 \in \mathbb{Z}$, and $a_2 = n + 1/2$ for $n \in \mathbb{Z}$.

$$\begin{aligned} f(x,y) = \cos(a_2\pi y)\Big[&\left(k^2 - (a_1^2\pi^2 + a_2^2\pi^2)\right)\sin(a_1\pi x)\,(1 + 0.2\sin(m\pi x)) \\ &+ 0.4 a_1 m\pi^2 \cos(a_1\pi x)\cos(m\pi x) - 0.2 m^2\pi^2 \sin(a_1\pi x)\sin(m\pi x)\Big]. \end{aligned} \tag{19}$$

In our experiment, we choose parameters: $k = 1, a_1 = 1, a_2 = 1.5, m = 15$.

We employ 4-hidden layers with 64 nodes per layer and use hyperbolic tangent function for helmholtz experiment. We used Adam optimizer with learning rate $\lambda = 5 \times 10^{-4}$, and set $(\omega_r, \omega_b) = (1, 100)$, $(N_r, N_b) = (2000, 100)$.

A.3 KLEIN-GORDON EQUATION

The Klein–Gordon equation is given by

$$\begin{aligned} u_{tt}(t,x) - u_{xx}(t,x) + \gamma u^k(t,x) &= f(t,x), && (t,x) \in [0,T] \times \Omega, \\ u(0,x) &= g_1(x), && x \in \Omega, \\ u_t(0,x) &= g_2(x), && x \in \Omega, \\ u(t,x) &= h(t,x), && (t,x) \in [0,T] \times \partial\Omega, \\ \Omega &= [0,1]. \end{aligned} \tag{20}$$

We set parameters to $k = 3$, $\gamma = 1$, $T = 1$ and the initial conditions, $g_1(x) = (1 + C)x$, $g_2(x) = 0$ for all $x \in \Omega$. Here, we use a solution with an additional multiscale term,

$$u^*(t,x) = x(\cos(3\pi t) + C\cos(m\pi t)) + (tx)^3, \tag{21}$$

where $f(t,x), h(t,x)$ is derived by substituting $u^*(t,x)$ into the given equation. In this experiment, we choose $m = 20, C = 0.2$.

We employ 4-hidden layers with 64 nodes per layer and use hyperbolic tangent activation for Klein-Gordon experiment. We used Adam optimizer with learning rate $\lambda = 5 \times 10^{-4}$, and set $(\omega_r, \omega_b, \omega_i) = (1, 1000, 1000)$, $(N_r, N_b, N_i) = (500, 100, 100)$.

For the experiments in Table 1, the parameters were modified to $m = 40$, $C = 0.5$, and $T = 0.5$ with $N_r = 1500$, while all other configurations remained unchanged.

A.4 Heat Equation

The Heat equation is

$$\begin{aligned} \frac{\partial u}{\partial t}(x,t) - \nu \frac{\partial^2 u}{\partial x^2}(x,t) &= f(x), \qquad (x,t) \in \Omega \times [0,T], \\ u(0,t) &= 0, \qquad t \in [0,T], \\ u(1,t) &= 0, \qquad t \in [0,T], \\ u(x,0) &= 0, \qquad x \in \Omega, \\ \Omega &= [0,1]. \end{aligned} \tag{22}$$

We use a multi-scale forcing term

$$f(x) = \sum_{j=1}^{2} a_j \sin(n_j \pi x), \tag{23}$$

which gives the manufactured solution

$$u^\star(x,t) = \sum_{j=1}^{2} \frac{a_j}{\nu (n_j \pi)^2} \left(1 - \exp\left(-\nu (n_j \pi)^2 t\right)\right) \sin(n_j \pi x). \tag{24}$$

In this formulation, ν is the diffusion coefficient, n_j is the spatial frequency of each sine mode, and a_j controls the amplitude of the corresponding source mode. Substituting $u^\star(x,t)$ into $\partial_t u - \nu \partial_{xx} u$ recovers $f(x)$, while $u^\star(x,0) = 0$ satisfies the initial condition.

In our experiment, we choose parameters: $\nu = 1$, $T = 0.5$, $(n_1, n_2) = (1, 30)$, and $(a_1, a_2) = (1, 15)$.

We employ 3-hidden layers with 128 nodes per layer and use hyperbolic tangent function for heat experiment. We used Adam optimizer with learning rate $\lambda = 10^{-3}$, and set $(\omega_r, \omega_b, \omega_i) = (1, 30, 30)$, $(N_r, N_b, N_i) = (2000, 100, 100)$.

A.5 Darcy Flow Equation

We consider the following Darcy flow problem:

$$\begin{aligned} -\nabla \cdot (a(x,y) \nabla u(x,y)) &= f(x,y), \qquad (x,y) \in \Omega, \\ u(x,y) &= 0, \qquad (x,y) \in \partial\Omega, \\ \Omega &= [-1,1] \times [-1,1]. \end{aligned} \tag{25}$$

where the permeability field is given by $a(x,y) = 1 + \frac{1}{2}\cos(\pi x)\cos(\pi y)$. The source term $f(x,y)$ is determined to satisfy the specific exact solution.

The manufactured solution is given by

$$u^*(x,y) = \sin(c_1 \pi x)\sin(c_1 \pi y) + C \sin(c_2 \pi x)\sin(c_2 \pi y), \tag{26}$$

where c_1, c_2 are natural numbers.

By substituting $u^*(x, y)$ into the governing equation, the explicit form of $f(x, y)$ is derived as:

$$\begin{aligned} f(x,y) = &\frac{\pi^2}{2}\sin(\pi x)\cos(\pi y)\left[c_1\cos(c_1\pi x)\sin(c_1\pi y) + Cc_2\cos(c_2\pi x)\sin(c_2\pi y)\right] \\ &+ \frac{\pi^2}{2}\cos(\pi x)\sin(\pi y)\left[c_1\sin(c_1\pi x)\cos(c_1\pi y) + Cc_2\sin(c_2\pi x)\cos(c_2\pi y)\right] \\ &+ a(x,y)\left[2(c_1\pi)^2\sin(c_1\pi x)\sin(c_1\pi y) + 2C(c_2\pi)^2\sin(c_2\pi x)\sin(c_2\pi y)\right]. \end{aligned} \tag{27}$$

In our experiment, we choose parameters: $c_1 = 1, c_2 = 10$, and $C = 0.1$.

We employ 3-hidden layers with 128 nodes per layer and use hyperbolic tangent function for Darcy Flow experiment. We used Adam optimizer with learning rate $\lambda = 10^{-3}$, and set $(\omega_r, \omega_b) = (1, 1000)$, $(N_r, N_b) = (1500, 100)$.

A.6 Convection Equation

The Convection equation is given by

$$\begin{aligned} \frac{\partial u}{\partial t}(t,x) + \beta\frac{\partial u}{\partial x}(t,x) &= f(t,x), \quad (t,x) \in [0,T]\times\Omega, \\ u(0,x) &= h(x), \qquad x \in \Omega, \\ u(t,0) &= g_1(t), \qquad t \in [0,T], \\ u(t,\pi) &= g_2(t), \qquad t \in [0,T], \\ \Omega &= [0,\pi]. \end{aligned} \tag{28}$$

We use the manufactured solution

$$u^*(t,x) = \sin(x - \beta t) + C\sin\big(m(\beta x + t)\big), \tag{29}$$

with the boundary condition $h(x) = u^*(0, x)$. The forcing term $f(t, x)$, and initial condition $g_1(t), g_2(t)$ are obtained by substituting u^* into the convection equation.

In this experiment, we choose $C = 0.5, \beta = 2, m = 20$.

We employ 4-hidden layers with 128 nodes per layer and use hyperbolic tangent function for convection experiment. We used Adam optimizer with learning rate $\lambda = 10^{-3}$, and set $(\omega_r, \omega_b, \omega_i) = (1, 1, 1)$, $(N_r, N_b, N_i) = (2500, 100, 100)$.

A.7 Justification for Large Boundary and Initial Weights

We use large weights for the boundary and initial losses throughout our experiments. This follows the observation of Hwang and Lim (2024) that PINN training can suffer from gradient conflicts between the PDE residual loss and the boundary or initial losses. When these gradients conflict, reducing the residual loss may increase the boundary loss, resulting in a solution that matches the residual objective but violates the prescribed constraints. Large w_b and w_i are therefore used as a safeguard to enforce the boundary and initial conditions reliably, so that the experiments isolate the effect of residual collocation resampling on multi-scale learning.

A.8 Hyperparameters for Spatio-Temporal Interpolation

The hyperparameters for the spatio-temporal interpolation sampling used in Figure 1 are summarized in Table 2.

B Evaluation Metrics

We report three complementary quantities: the relative L^2 error, the relative L^∞ error, and the relative H^1 seminorm error. Lower values indicate better agreement with the reference solution.

Relative L^2 error. The relative L^2 error measures overall agreement between the predicted and reference solutions:

$$e_{L^2} := \frac{\|u(\cdot;\theta) - u^\star\|_{L^2(\mathcal{D}_r)}}{\|u^\star\|_{L^2(\mathcal{D}_r)}}. \tag{30}$$

Table 2: Hyperparameter settings for spatio-temporal interpolation sampling.

Equation	Resampling Fraction (ρ)	Resampling Probability (p)
Poisson	0.3	0.10
Heat	0.9	0.05
Helmholtz	0.3	0.10
Darcy	0.3	0.05
Klein–Gordon	0.5	0.05
Convection	0.7	0.20

Relative H^1 seminorm error. To quantify derivative-level discrepancies, we report the relative H^1 seminorm error:

$$e_{H^1} := \frac{|u(\cdot;\theta) - u^\star|_{H^1(\mathcal{D}_r)}}{|u^\star|_{H^1(\mathcal{D}_r)}}, \tag{31}$$

where the H^1 seminorm is defined by

$$|f|_{H^1(\mathcal{D}_r)} := \left(\int_{\mathcal{D}_r} \|\nabla f(z)\|_2^2 \, dz \right)^{1/2}. \tag{32}$$

Unlike the L^2 error, which compares function values directly, the H^1 seminorm measures the magnitude of derivatives. It is therefore sensitive to local variation and fine-scale structure in the solution.

C Experimental Details

C.1 Software and Hardware Environments

We performed all experiments on a server running Ubuntu 20.04.6 LTS, equipped with an Intel® Core™ i9-10900X CPU and four NVIDIA GeForce RTX 4090 GPUs. The software environment was configured with Python 3.8.20, PyTorch 2.0.1, CUDA 11.7, and CUDNN 8.5.0.

C.2 Implementation Details for Figure 2

We provide the implementation details for the one-dimensional comparison shown in Figure 2. The manufactured solution, forcing term, and homogeneous Dirichlet boundary conditions follow the setup defined in Section 3.1.

We compare direct regression and PINN training under the same neural architecture. The regression model is trained by minimizing the mean squared error on solution values using 1000 training points. The PINN model is trained by minimizing the Poisson residual using $N_r = 1000$ residual collocation points, together with a boundary penalty enforcing the homogeneous Dirichlet conditions.

Both models use a fully connected multilayer perceptron with three hidden layers of width 128 and $\tanh$ activations. All linear layers are initialized with Xavier normal weights and zero biases, and optimization is performed with Adam. For direct regression, we use a learning rate of 10^{-2} and train for 10000 iterations. For PINN training, we use a learning rate of 10^{-3}, train for 5000 iterations, and set the boundary loss weight to $w_b = 0.04$.

C.3 Residual Gradient SNR Estimation

We estimate the residual-gradient SNR using only the interior PDE residual loss. Boundary and initial-condition losses are excluded from this diagnostic. Recall that p_r denotes the uniform distribution on $\mathcal{D}_r$, and that $\widehat{g}_r(\theta; \mathcal{X}_r)$ denotes the empirical residual gradient computed from a residual collocation set $\mathcal{X}_r$.

At each checkpoint k, we approximate the population residual gradient $g_r(\theta_k)$ using an independent reference collocation set

$$\mathcal{X}_{\mathrm{ref}}^{(k)} = \{\boldsymbol{z}_{\mathrm{ref}}^{n,(k)}\}_{n=1}^{N_{\mathrm{ref}}}, \qquad \boldsymbol{z}_{\mathrm{ref}}^{n,(k)} \sim p_r, \qquad N_{\mathrm{ref}} = 10N_r.$$

The corresponding reference gradient is

$$g_{\mathrm{ref}}(\theta_k) := \widehat{g}_r(\theta_k; \mathcal{X}_{\mathrm{ref}}^{(k)}).$$

To estimate the covariance term in the denominator, we independently draw M residual collocation sets of the training batch size,

$$\mathcal{X}_{r,j}^{(k)} = \{\boldsymbol{z}_j^{n,(k)}\}_{n=1}^{N_r}, \qquad \boldsymbol{z}_j^{n,(k)} \sim p_r, \qquad j = 1, \dots, M,$$

and compute

$$\widehat{g}_{r,j}(\theta_k) := \widehat{g}_r(\theta_k; \mathcal{X}_{r,j}^{(k)}), \qquad j = 1, \dots, M.$$

Let

$$\overline{g}_r(\theta_k) := \frac{1}{M} \sum_{j=1}^{M} \widehat{g}_{r,j}(\theta_k).$$

We estimate the trace of the residual-gradient covariance by

$$\widehat{\mathrm{Tr}(\Sigma_r)}(\theta_k) := \frac{1}{M-1} \sum_{j=1}^{M} \|\widehat{g}_{r,j}(\theta_k) - \overline{g}_r(\theta_k)\|_2^2 .$$

The empirical residual-gradient SNR is then

$$\widehat{\mathrm{SNR}}(\theta_k) := \frac{\|g_{\mathrm{ref}}(\theta_k)\|_2}{\sqrt{\widehat{\mathrm{Tr}(\Sigma_r)}(\theta_k)}}.$$

In the convection equation and Darcy equation, we use $N_r = 3 \times 10^3$, hence $N_{\mathrm{ref}} = 3 \times 10^4$, and set $M = 8$. The SNR is evaluated at iteration 0 and every 200 training iterations. For the Klein–Gordon experiment, we use $N_r = 10^3$ and set $M = 8$. The SNR is evaluated at iteration 0 and every 10 training iterations.

C.4 Mode-Wise SNR Estimation

For the Poisson experiment in Section 4.3, mode-wise convergence is quantified by the coefficient error obtained from the modal projection in Section 3.2. The mode-wise SNR is constructed from this same error functional. Its purpose is to measure whether the stochastic residual-gradient update provides a reliable descent signal for each modal coefficient, rather than merely a large signal in the full parameter space. As in Appendix C.3, the estimate is based on the interior PDE residual gradient, and boundary losses are not included in this statistic.

Let $\mathcal{Q} = \{\boldsymbol{z}_q\}_{q=1}^{N_{\mathcal{Q}}}$ denote the fixed evaluation grid used for the modal projection, and let $\phi_c(x, y) = \sin(cx)\sin(cy)$ be the target basis function associated with mode $c \in \{c_1, c_2\}$. At checkpoint k, the projected coefficient and the signed relative coefficient error are computed as

$$\widehat{a}_c(\theta_k) := \frac{\sum_{q=1}^{N_{\mathcal{Q}}} u_{\theta_k}(\boldsymbol{z}_q)\phi_c(\boldsymbol{z}_q)}{\sum_{q=1}^{N_{\mathcal{Q}}} \phi_c(\boldsymbol{z}_q)^2}, \qquad \widehat{\delta}_c(\theta_k) := \frac{\widehat{a}_c(\theta_k) - a_c^\star}{|a_c^\star|}.$$

We regard the normalized squared modal error as the scalar objective whose first-order descent is to be tested. Differentiating the discrete projection with respect to the network parameters gives

$$\widehat{J}_c(\theta_k) := \nabla_\theta \widehat{a}_c(\theta_k), \qquad \widehat{v}_c(\theta_k) := \nabla_\theta \left(\frac{1}{2}\widehat{\delta}_c(\theta_k)^2\right) = \frac{\widehat{\delta}_c(\theta_k)}{|a_c^\star|}\widehat{J}_c(\theta_k).$$

The stochasticity of random resampling is then evaluated after projecting the residual-gradient estimator onto this modal-error direction. We draw M independent residual collocation batches of the same size used during training,

$$\mathcal{X}_{r,j}^{(k)} := \{\boldsymbol{z}_j^{n,(k)}\}_{n=1}^{N_r}, \qquad \boldsymbol{z}_j^{n,(k)} \sim p_r, \qquad j = 1, \dots, M,$$

and compute the corresponding residual gradients

$$\widehat{g}_{r,j}(\theta_k) := \widehat{g}_r(\theta_k, \mathcal{X}_{r,j}^{(k)}), \qquad j = 1, \dots, M.$$

For each mode and each residual batch, the induced first-order coefficient step and the corresponding first-order decrease of the normalized squared modal error are

$$\begin{aligned}
\widehat{s}_{c,j}^{(k)} &:= -\widehat{J}_c(\theta_k)^\top \widehat{g}_{r,j}(\theta_k), \\
\widehat{a}_c(\theta_k - \eta \widehat{g}_{r,j}(\theta_k)) &= \widehat{a}_c(\theta_k) + \eta\, \widehat{s}_{c,j}^{(k)} + O(\eta^2), \\
\widehat{Y}_{c,j}^{(k)} &:= \widehat{v}_c(\theta_k)^\top \widehat{g}_{r,j}(\theta_k) = \frac{\widehat{\delta}_c(\theta_k)}{|a_c^\star|} \widehat{J}_c(\theta_k)^\top \widehat{g}_{r,j}(\theta_k) = -\frac{\widehat{\delta}_c(\theta_k)}{|a_c^\star|} \widehat{s}_{c,j}^{(k)}, \\
\mathcal{E}_c(\theta_k - \eta \widehat{g}_{r,j}(\theta_k)) &= \mathcal{E}_c(\theta_k) - \eta\, \widehat{Y}_{c,j}^{(k)} + O(\eta^2),
\end{aligned}$$

where $\mathcal{E}_c(\theta) = \frac{1}{2}\widehat{\delta}_c(\theta)^2$. Thus $\widehat{Y}_{c,j}^{(k)}$ is the sampled linearized improvement signal for mode c. Positive values decrease the modal error to first order, whereas negative values increase it.

The empirical mode-wise SNR is obtained by comparing the mean of this scalar descent signal with its standard deviation across independently resampled residual batches:

$$\begin{aligned}
\overline{Y}_c^{(k)} &:= \frac{1}{M} \sum_{j=1}^{M} \widehat{Y}_{c,j}^{(k)}, \\
\widehat{\sigma}_{Y,c}^{(k)} &:= \left[\frac{1}{M-1} \sum_{j=1}^{M} \left(\widehat{Y}_{c,j}^{(k)} - \overline{Y}_c^{(k)} \right)^2 \right]^{1/2}, \\
\widehat{\mathrm{SNR}}_c(\theta_k) &:= \frac{\overline{Y}_c^{(k)}}{\widehat{\sigma}_{Y,c}^{(k)}}.
\end{aligned}$$

This scalar estimator is the empirical counterpart of Eq. eq. (15). The quantity $\overline{Y}_c^{(k)}$ estimates $v_c(\theta_k)^\top g_r(\theta_k)$, and $(\widehat{\sigma}_{Y,c}^{(k)})^2$ estimates $v_c(\theta_k)^\top \Sigma_r(\theta_k) v_c(\theta_k)$. Therefore $\widehat{\mathrm{SNR}}_c(\theta_k) < 1$ indicates that the average mode-wise descent signal is smaller than the batch-to-batch fluctuation induced by random resampling. The statistic is evaluated at the same checkpoints as the residual-gradient SNR, using the training residual batch size N_r for each stochastic draw.

For our experiments, we use $N_b = 100$ points on each boundary segment and evaluate the modal projection on a fixed 121×121 grid, so that $N_{\mathcal{Q}} = 121^2$. All runs use boundary weight $\omega_b = 40$, learning rate $\lambda = 10^{-3}$. The SNR statistic is evaluated every 200 iterations. The mode-wise SNR is estimated from $M = 16$ independently resampled residual batches. The reference residual-gradient batch size is $N_{\mathrm{ref}} = 10N_r$.

C.5 Details for Table 1 in Section 5.2

Benchmark equations. We construct multi-scale variants of six benchmark equations widely used in the PINN literature: Poisson, Heat, Helmholtz, Darcy, Klein–Gordon, and Convection. These benchmarks are designed to evaluate whether a method can resolve spectral bias under controlled multi-scale settings. Detailed formulations of the benchmark equations and the corresponding training settings are provided in Appendix A.

Baselines. We consider six collocation sampling strategies. Random resampling and fixed sampling represent the two endpoint strategies. We include importance sampling as a standard variance-reduction baseline, using a proposal population of size $10 \times N_r$. Spatial interpolation varies the refresh ratio over $\rho \in \{0.3, 0.5, 0.7\}$, temporal interpolation varies the resampling probability over $p \in \{0.1, 0.05, 0.01\}$, and spatio-temporal interpolation combines the two using the same grids.

All methods are evaluated over five independent random seeds. For each run, we select the checkpoint with the lowest training loss and evaluate the relative H^1 seminorm error at that checkpoint. We then report the mean and standard deviation across seeds. For the interpolation-based methods, we report the best-performing hyperparameter configuration for each benchmark from the grids above.

D COMPUTATIONAL OVERHEAD

In this section, we evaluate whether the interpolation schemes introduce additional runtime or memory overhead compared with standard collocation sampling strategies. We report iteration throughput, measured in iterations per second and averaged over 100 training iterations, and peak GPU memory usage for each benchmark equation.

Table 3: Iteration throughput of each sampling strategy measured in iterations per second (iter/s).

Method	Poisson	Heat	Helmholtz	Darcy	Klein–Gordon	Convection
Random Resampling	47.80	55.67	45.62	42.43	47.16	62.32
Fixed Sampling	51.72	61.58	45.01	45.83	49.99	63.85
Importance Sampling	28.40	26.83	29.54	28.17	30.82	44.54
Spatial Interpolation	53.68	64.02	43.85	45.33	49.16	64.46
Temporal Interpolation	52.22	61.78	44.74	45.05	52.38	63.06
Spatio-Temporal Interpolation	47.89	57.79	41.76	43.65	48.21	60.33

Table 3 reports the iteration throughput of each sampling strategy, measured in iterations per second and averaged over 100 training iterations. For the interpolation schemes, we use $\rho = 0.3$ for spatial interpolation, $p = 0.1$ for temporal interpolation, and $(\rho, p) = (0.3, 0.1)$ for spatio-temporal interpolation. For importance sampling, we use a candidate population of size $10N_r$ at each iteration.

Importance sampling is substantially slower because it evaluates the residual on the enlarged candidate population and computes residual-based sampling probabilities before drawing the training collocation points. Random resampling incurs only a marginal overhead from refreshing the full residual collocation set at every iteration, and its throughput remains close to fixed sampling and the interpolation schemes. Spatial, temporal, and spatio-temporal interpolation reuse collocation points either fully or partially across iterations, achieving comparable throughput without additional computational cost.

Table 4: Peak memory usage recorded during training for each benchmark equation.

Metric	Poisson	Heat	Helmholtz	Darcy	Klein–Gordon	Convection
Peak memory (MB)	1668.2	547.1	1125.0	1658.2	1117.3	687.4

Table 4 reports the peak memory usage recorded during training for each benchmark equation.

E STANDARD VARIANCE REDUCTION METHODS FAIL TO MITIGATE SPECTRAL BIAS

Two canonical strategies for reducing stochastic gradient variance are importance sampling (Kahn and Marshall, 1953; Katharopoulos and Fleuret, 2018) and stochastic variance-reduced gradient (SVRG) (Johnson and Zhang, 2013; Reddi et al., 2016). Importance sampling has been extensively studied in the PINN literature (Yang et al., 2023; Zhang et al., 2025; Nabian et al., 2021), where collocation points are concentrated in regions of large PDE residual. Since the true gradient magnitude is intractable to compute, practical methods often use the residual loss as a proxy for the sampling density,

$$q_\theta(\boldsymbol{z}) \propto \ell_r(\boldsymbol{z}; \theta)^\kappa + \varepsilon.$$

We choose $\kappa = 1$. In our implementation, importance sampling is performed through a finite candidate population. At each iteration, we first draw a uniformly sampled candidate set

$$\mathcal{C}_m^{(k)} = \{\widetilde{\boldsymbol{z}}_s^{(k)}\}_{s=1}^{mN_r}, \qquad \widetilde{\boldsymbol{z}}_s^{(k)} \sim p_r,$$

where $m \in \{5, 10, 20\}$. We then assign discrete sampling probabilities

$$\pi_s^{(k)} = \frac{\ell_r(\widetilde{\boldsymbol{z}}_s^{(k)}; \theta_k)^\kappa + \varepsilon}{\sum_{q=1}^{mN_r} \left(\ell_r(\widetilde{\boldsymbol{z}}_q^{(k)}; \theta_k)^\kappa + \varepsilon\right)},$$

and sample N_r residual collocation points from $\mathcal{C}_m^{(k)}$ according to $\{\pi_s^{(k)}\}_{s=1}^{mN_r}$.

To keep the estimator unbiased with respect to the uniform residual objective, we apply importance weights rather than optimizing the unweighted residual loss over the biased sample. Conditional on $\mathcal{C}_m^{(k)}$, if $i_n \sim \{\pi_s^{(k)}\}_{s=1}^{mN_r}$, we use

$$\widehat{g}_{\mathrm{IS}}(\theta_k) = \frac{1}{N_r} \sum_{n=1}^{N_r} \frac{1/(mN_r)}{\pi_{i_n}^{(k)}} \nabla_\theta \ell_r(\widetilde{\boldsymbol{z}}_{i_n}^{(k)}; \theta_k).$$

This satisfies

$$\mathbb{E}[\widehat{g}_{\mathrm{IS}}(\theta_k) \mid \mathcal{C}_m^{(k)}] = \frac{1}{mN_r} \sum_{s=1}^{mN_r} \nabla_\theta \ell_r(\widetilde{\boldsymbol{z}}_s^{(k)}; \theta_k),$$

so the estimator remains unbiased for $g_r(\theta_k)$ after averaging over the uniformly drawn candidate set.

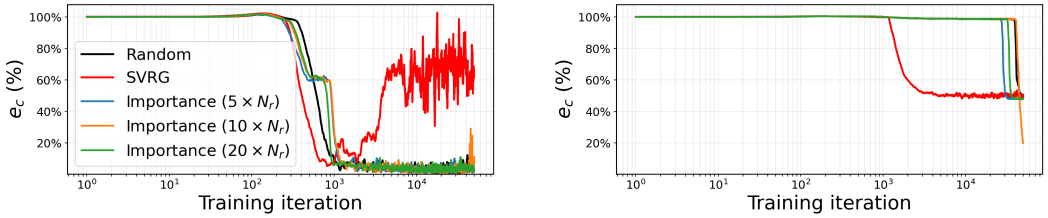
SVRG, on the other hand, periodically computes a large-batch reference gradient at snapshot parameters and uses it to correct the stochastic gradient estimate. Let $\widetilde{\theta}$ denote the snapshot parameter. Every 10^3 iterations, we draw a reference collocation set $\mathcal{X}_{\mathrm{ref}}$ with $|\mathcal{X}_{\mathrm{ref}}| = 10N_r$, and compute

$$g_{\mathrm{ref}}(\widetilde{\theta}) := \widehat{g}_r(\widetilde{\theta}; \mathcal{X}_{\mathrm{ref}}).$$

For a mini-batch residual collocation set $\mathcal{X}_r^{(k)}$, the SVRG corrected residual gradient is

$$g_{\mathrm{SVRG}}(\theta_k) := \widehat{g}_r(\theta_k; \mathcal{X}_r^{(k)}) - \widehat{g}_r(\widetilde{\theta}; \mathcal{X}_r^{(k)}) + g_{\mathrm{ref}}(\widetilde{\theta}).$$

We use this corrected residual gradient within the Adam optimizer. While SVRG has been widely adopted in conventional machine learning, its application to PINNs remains largely unexplored.



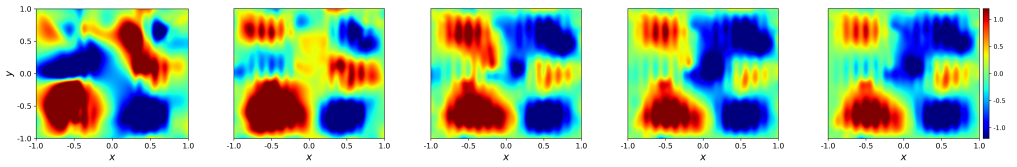
(a) $c = 4$ (b) $c = 40$

Figure 8: Mode-wise coefficient error for variance-reduction methods on the two-mode Poisson equation. Importance $(m \times N_r)$ samples N_r collocation points from a uniformly drawn candidate population of size mN_r, with $m \in \{5, 10, 20\}$, using probabilities proportional to the residual loss.

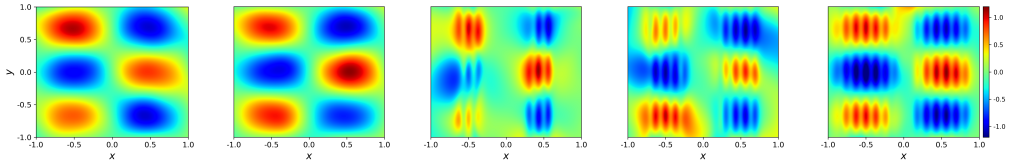
We compare these two methods against random resampling on the two mode Poisson equation in eq. (9)–eq. (10). As shown in Figure 8, importance sampling provides only limited improvement even when the candidate population is enlarged from $5N_r$ to $20N_r$. SVRG can induce an earlier decrease of the high-frequency error, but the resulting trajectory is unstable and does not yield reliable convergence. This instability reflects the mismatch between the stale reference correction used by SVRG and the rapidly evolving PDE residual field in PINN training. The reference gradient is computed at a snapshot parameter and refreshed only every 10^3 iterations, while the current residual-gradient field can change substantially as the network evolves. The corrected gradient can therefore become poorly aligned with the current residual-gradient estimator, destabilizing the overall optimization dynamics. These results suggest that standard variance reduction alone is insufficient to remove the resampling-induced delay in fine-scale learning. Effective mitigation requires controlling the collocation-induced bias–variance trade-off during training.

F Resampling Effects with Fourier Feature Embeddings

Fourier feature embedding is one of the frequency-aware architectures used to address multi-scale problems. In this section, we investigate the effect of resampling on spectral bias with Fourier embeddings, presenting detailed iteration-level results for the Helmholtz equation.

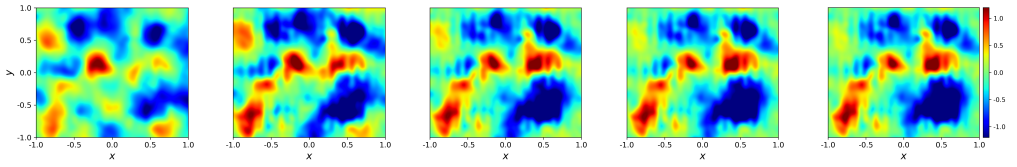


(a) $k = 500$ (b) $k = 1000$ (c) $k = 2000$ (d) $k = 3000$ (e) $k = 5000$

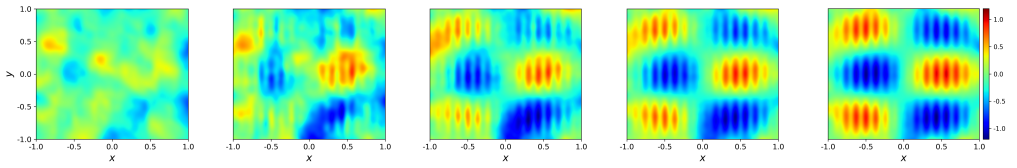


(f) $k = 500$ (g) $k = 1000$ (h) $k = 2000$ (i) $k = 3000$ (j) $k = 5000$

Figure 9: Helmholtz equation with Fourier feature embeddings ($\sigma = 2$). Panels (a)–(e) and (f)–(j) show iteration-wise snapshots under fixed sampling and random resampling, respectively, with subcaptions indicating the training iteration k.



(a) $k = 100$ (b) $k = 300$ (c) $k = 500$ (d) $k = 700$ (e) $k = 1000$



(f) $k = 100$ (g) $k = 300$ (h) $k = 500$ (i) $k = 700$ (j) $k = 1000$

Figure 10: Helmholtz equation with Fourier feature embeddings ($\sigma = 6$). Panels (a)–(e) and (f)–(j) show iteration-wise snapshots under fixed sampling and random resampling, respectively, with subcaptions indicating the training iteration k.

Gaussian Fourier embeddings are defined as,

$$\gamma(\mathbf{v}) = [\cos(2\pi \mathbf{B}\mathbf{v}), \sin(2\pi \mathbf{B}\mathbf{v})]^{\mathrm{T}} \tag{33}$$

where $\mathbf{v} \in \mathbb{R}^d$ is the input coordinate, and $\mathbf{B} \in \mathbb{R}^{m \times d}$ is a random weight matrix. Each entry of $\mathbf{B}$ is sampled from an isotropic Gaussian distribution $\mathcal{N}(0, \sigma^2)$, where σ is a hyperparameter determining the scale of the frequencies.

We conducted experiments on the Helmholtz equation described in Appendix A.2, with different σ values. Both experiments were done with 4 hidden layers with 64 nodes per layer and used the hyperbolic tangent function. Also, the Adam optimizer was used with learning rate $\lambda = 10^{-4}$, and we set $(\omega_r, \omega_b) = (1, 100)$, $(N_r, N_b) = (1000, 100)$, and set the Fourier dimension $m = 64$.

These experiments show that Fourier feature embeddings change the spectral learning dynamics in a bandwidth dependent manner. For the moderate bandwidth $\sigma = 2$, Figure 9 shows that residual spectral bias remains. Fine-scale oscillations emerge gradually, and the contrast between fixed sampling and random resampling persists. Fixed sampling introduces high-frequency patterns earlier but distorts the global structure, whereas random resampling recovers the coarse solution more faithfully before resolving the fine scales.

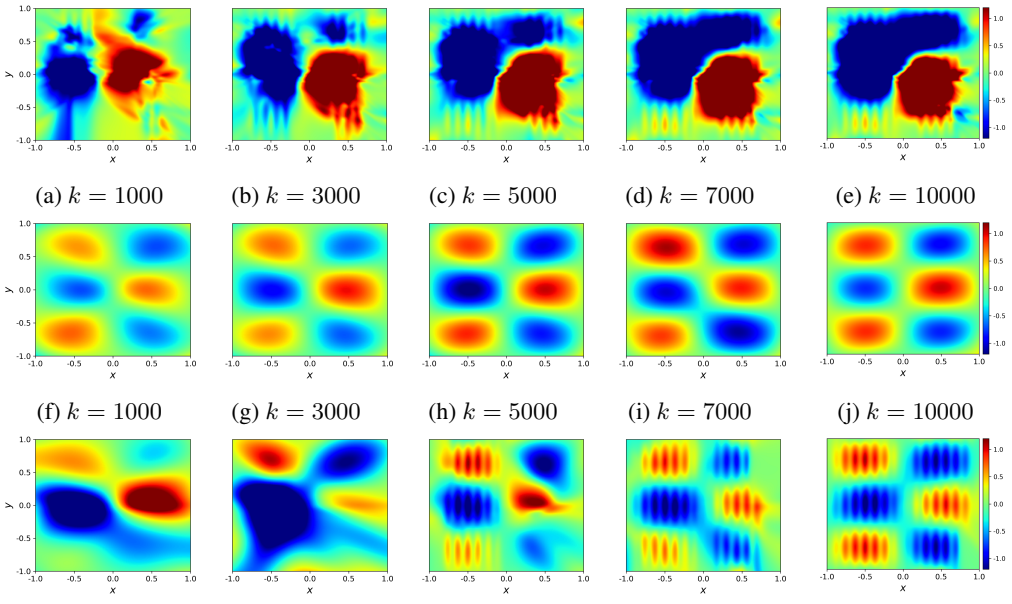
For the larger bandwidth $\sigma = 6$, Figure 10 shows that the low-frequency first ordering is much weaker. Fine-scale oscillations appear early, even before the macro-scale solution is fully formed. In this case, random resampling no longer shows a clear amplification of spectral bias, since the Fourier embedding makes high-frequency components accessible early in training. Understanding how representation bandwidth and collocation sampling jointly shape the spectral learning dynamics of PINNs remains an interesting direction for future work.

G Additional Results

G.1 Iteration-wise Learning Dynamics for Figure 1

Figure 1 shows that random resampling tends to recover the macro-scale structure, fixed sampling often distorts the global solution, and interpolation between fixed and random sampling recovers both global and fine-scale structure across equations. Here, we provide iteration-wise snapshots of the learned solutions to visualize how each collocation strategy evolves during training.

Detailed formulations of the benchmark equations and the corresponding training settings are provided in Appendix A. The hyperparameters for spatio-temporal interpolation sampling are summarized in Table 2.

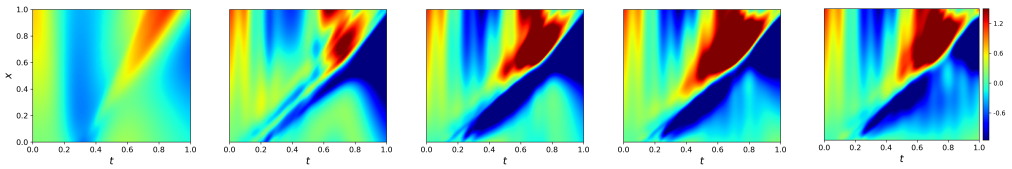


(a) $k = 1000$ (b) $k = 3000$ (c) $k = 5000$ (d) $k = 7000$ (e) $k = 10000$

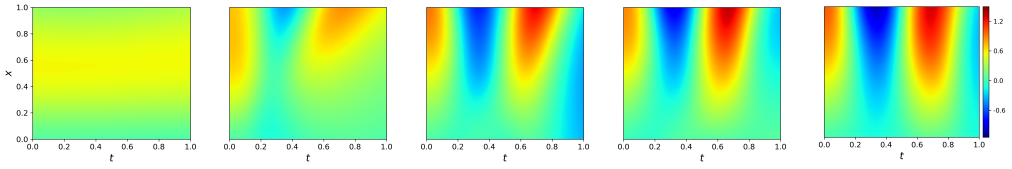
(f) $k = 1000$ (g) $k = 3000$ (h) $k = 5000$ (i) $k = 7000$ (j) $k = 10000$

(k) $k = 1000$ (l) $k = 3000$ (m) $k = 5000$ (n) $k = 7000$ (o) $k = 10000$

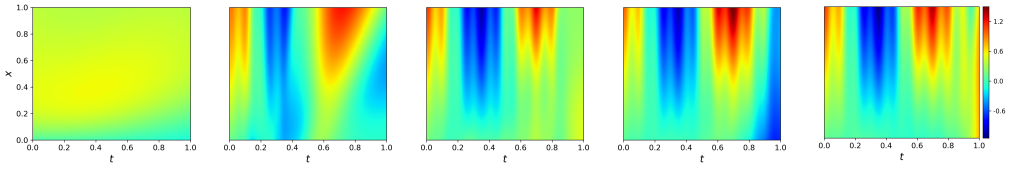
Figure 11: Helmholtz equation. Panels show iteration-wise snapshots, with subcaptions indicating the training iteration k. Panels (a)–(e), (f)–(j), and (k)–(o) correspond to fixed sampling, random resampling, and spatio-temporal interpolation, respectively.



(a) $k = 500$ (b) $k = 1000$ (c) $k = 1500$ (d) $k = 2000$ (e) $k = 2500$

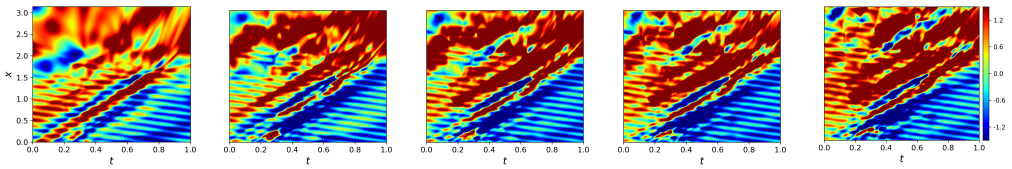


(f) $k = 500$ (g) $k = 1000$ (h) $k = 1500$ (i) $k = 2000$ (j) $k = 2500$

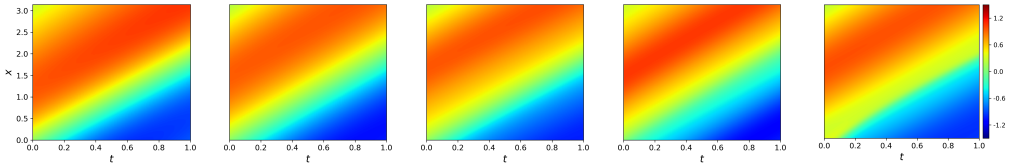


(k) $k = 500$ (l) $k = 1000$ (m) $k = 1500$ (n) $k = 2000$ (o) $k = 2500$

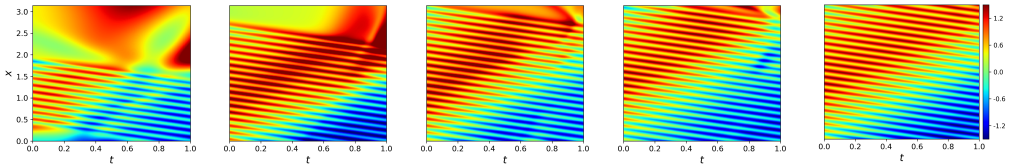
Figure 12: Klein–Gordon equation. Panels show iteration-wise snapshots, with subcaptions indicating the training iteration k. Panels (a)–(e), (f)–(j), and (k)–(o) correspond to fixed sampling, random resampling, and spatio-temporal interpolation, respectively.



(a) $k = 1000$ (b) $k = 3000$ (c) $k = 5000$ (d) $k = 7000$ (e) $k = 10000$



(f) $k = 1000$ (g) $k = 3000$ (h) $k = 5000$ (i) $k = 7000$ (j) $k = 10000$

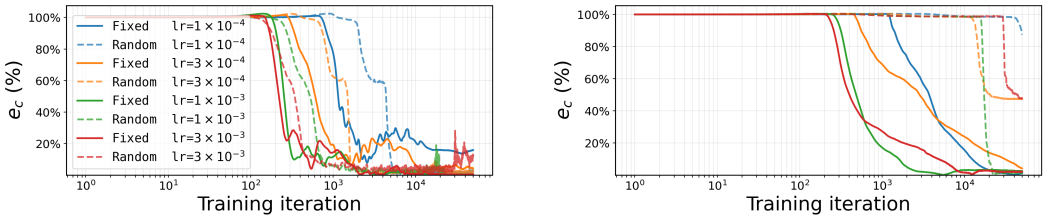


(k) $k = 1000$ (l) $k = 3000$ (m) $k = 5000$ (n) $k = 7000$ (o) $k = 10000$

Figure 13: Convection equation. Panels show iteration-wise snapshots, with subcaptions indicating the training iteration k. Panels (a)–(e), (f)–(j), and (k)–(o) correspond to fixed sampling, random resampling, and spatio-temporal interpolation, respectively.

G.2 Learning Rate Sensitivity

Learning rate is a key optimization hyperparameter and can substantially affect the speed and stability of PINN training. However, our goal is not to tune the learning rate separately for each sampling strategy, but to examine whether the observed difference between fixed and random resampling can be explained by a learning rate effect. To control for this factor, all main experiments use the same learning rate for fixed and random resampling.



(a) $c = 4$ (b) $c = 40$

Figure 14: Mode-wise coefficient error on the two-mode Poisson equation under different learning rates. Solid and dashed curves denote fixed and random resampling, respectively.

Figure 14 compares fixed and random resampling under several learning rates. We consider the two-dimensional Poisson equation from Section 3.2 with $a_4^\star = 1$ and $a_{40}^\star = 0.1$ for modes $(4, 4)$ and $(40, 40)$, respectively. We use a fully connected network with 3 hidden layers of 128 units each, $\tanh$ activations, and Xavier-normal initialization. The model is trained with the Adam optimizer for 5×10^4 iterations, while varying the learning rate over $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}\}$. At each step, we use 400 boundary points and $N = 10^4$ interior collocation points. While changing the learning rate affects the overall training speed, it does not eliminate the qualitative gap between the two sampling strategies. In particular, random resampling remains substantially delayed for the high-frequency mode, whereas fixed sampling captures the mode much earlier across the tested learning rates. This suggests that the fixed–random contrast observed in our experiments is not simply an artifact of using a particular learning rate.

G.3 Noise Injection to Fixed Sampling Gradient

To further isolate the effect of gradient noise on spectral learning dynamics, we conduct a controlled noise-injection experiment on the same two-mode Poisson equation in eq. (9)–eq. (10). We use fixed sampling with $N_r = 10^4$ collocation points throughout training, so that the collocation set itself does not change across iterations. We then perturb the fixed sampling gradient by adding isotropic Gaussian noise $\xi \sim \mathcal{N}(0, \sigma^2 I)$ at each iteration.

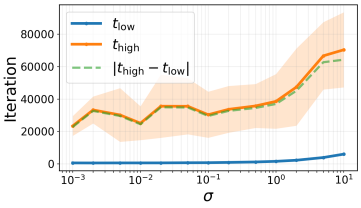


Figure 15: Capture iterations $t_{\text{low}}(\sigma)$, $t_{\text{high}}(\sigma)$, and $\text{gap}(\sigma)$ under varying σ.

This injected noise is not intended to reproduce the exact noise induced by random resampling. In random resampling, the gradient fluctuation is structured, state-dependent, and determined by the covariance of the stochastic residual gradient estimator. By contrast, the Gaussian perturbation used here is isotropic and independent of the residual field. The purpose of this experiment is therefore not to model resampling noise faithfully, but to provide a controlled intervention in which the collocation set is fixed and only the gradient noise level is varied.

As shown in Figure 15, $\text{gap}(\sigma)$ increases with the noise level σ. Thus, even when the collocation points are fixed, stronger gradient perturbations can widen the separation between the capture times of the low and high frequency modes. Although this experiment does not reproduce the structured noise of random resampling, whose covariance depends on the current parameters and residual field, it provides a controlled indication that gradient noise can preferentially delay the recovery of high frequency components. This observation is consistent with our interpretation that sampling variance contributes to the amplification of spectral bias in PINNs.

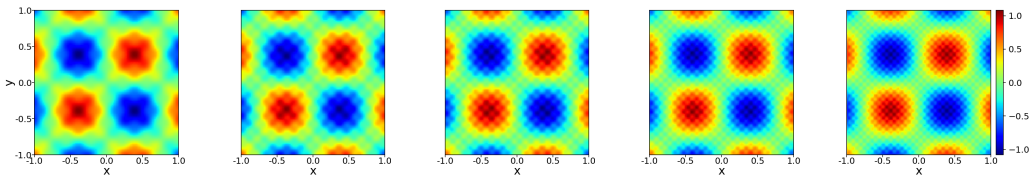
To inject noise that carries the structure of the resampling gradient, we further replace the isotropic perturbation with a covariance matched one. We set $c_2 = 35$ to allow faster convergence and keep the other settings unchanged. For each seed, we estimate the empirical covariance $\widehat{\Sigma}$ of the PDE residual gradient from $S = 50$ independently resampled collocation batches of size 1000 at the initialization, and add $\xi_k \sim \mathcal{N}(0, \alpha^2\widehat{\Sigma})$ to the update direction at each iteration k. The injected noise thus follows the anisotropic structure of the resampling gradient at initialization, while α controls only its magnitude. Table 5 shows that the gap widens monotonically with α, and the increase in t_{high} accounts for most of this widening. This provides more direct evidence that noise with the structure of resampling noise delays high-frequency recovery.

Table 5: Capture times under covariance matched noise injection on the two-mode Poisson equation with $c_2 = 35$. Results are reported as mean $\pm$ standard deviation over three random seeds.

α	t_{low}	t_{high}	gap
0	$(6.030_{\pm 0.210}) \times 10^2$	$(9.092_{\pm 2.005}) \times 10^3$	$(8.489_{\pm 1.989}) \times 10^3$
0.1	$(5.843_{\pm 0.274}) \times 10^2$	$(9.634_{\pm 1.378}) \times 10^3$	$(9.050_{\pm 1.357}) \times 10^3$
1	$(6.133_{\pm 0.329}) \times 10^2$	$(1.037_{\pm 0.246}) \times 10^4$	$(9.755_{\pm 2.484}) \times 10^3$
10	$(1.324_{\pm 0.726}) \times 10^3$	$(1.286_{\pm 0.464}) \times 10^4$	$(1.153_{\pm 0.398}) \times 10^4$
50	$(2.087_{\pm 0.822}) \times 10^3$	$(2.020_{\pm 0.603}) \times 10^4$	$(1.812_{\pm 0.537}) \times 10^4$
100	$(2.673_{\pm 1.412}) \times 10^3$	$(2.603_{\pm 0.606}) \times 10^4$	$(2.336_{\pm 0.475}) \times 10^4$

G.4 Target Solutions under Varying High-Frequency Modes

We visualize the exact target solutions used in the frequency-dependence experiment of Section 3.3. Across all cases, the low-frequency component is kept fixed, while only the high-frequency index c_2 is varied.



(a) $c_2 = 20$ (b) $c_2 = 30$ (c) $c_2 = 40$ (d) $c_2 = 50$ (e) $c_2 = 60$

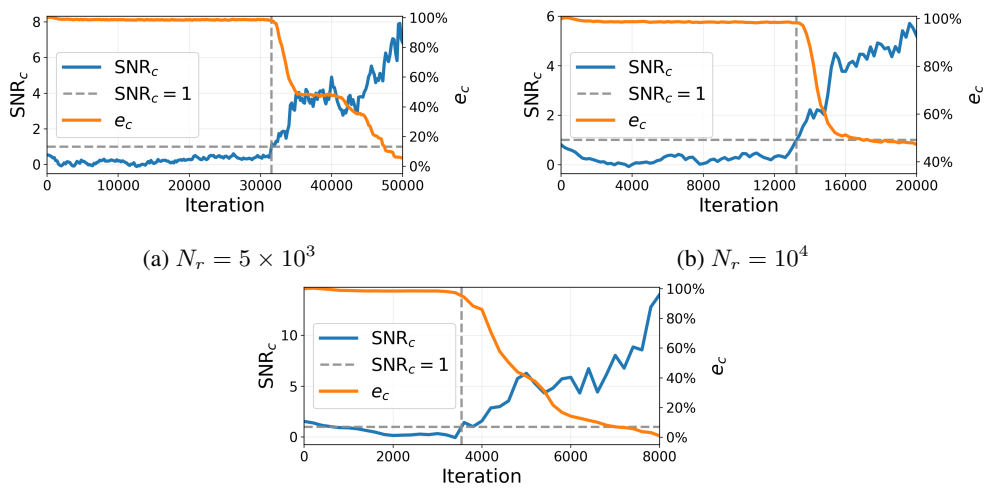
Figure 16: Exact target solutions used in the frequency-sweep experiment for the two-mode Poisson problem. The low-frequency mode $c_1 = 4$ is fixed, while the high-frequency index c_2 is varied from 20 to 60.

G.5 Mode-wise SNR under Varying Collocation Budgets

Figure 17 reports the mode-wise SNR diagnostic for the high-frequency mode $c = 40$ under different residual collocation budgets N_r. Across all budgets, the coefficient error e_{40} remains nearly unchanged while $\text{SNR}_{40} < 1$, indicating that the projected descent signal for the high-frequency mode is dominated by resampling variance. Once SNR_{40} rises above this noise dominated regime and remains above it over successive iterations, e_{40} begins to decay. Increasing N_r shifts this transition earlier, consistent with the reduction of the residual gradient covariance as the collocation budget grows. These results show that the alignment between mode-wise SNR and high-frequency error decay is not specific to a single choice of N_r.

G.6 Interpolation Between Fixed Sampling and Random Resampling

We consider the two-dimensional Poisson equation from Section 3.2 with $a_4^\star = 1$ and $a_{40}^\star = 0.1$ for modes $(4, 4)$ and $(40, 40)$, respectively. We use a fully connected network with 3 hidden layers of 128 units each, $\tanh$ activations, and Xavier normal initialization, trained with Adam optimizer



(a) $N_r = 5 \times 10^3$ (b) $N_r = 10^4$

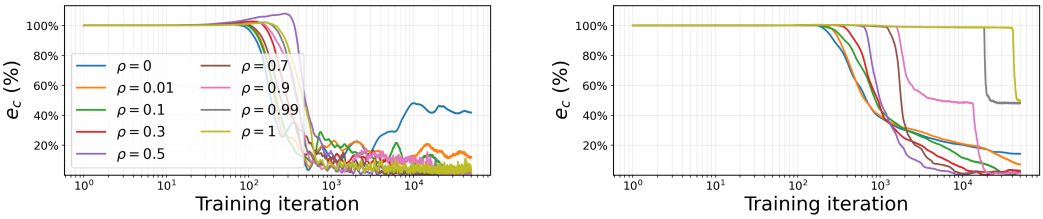
(c) $N_r = 5 \times 10^4$

Figure 17: Mode-wise SNR and high-frequency coefficient error on the Poisson problem.

at a learning rate of 10^{-3} for 5×10^4 iterations. At each step, we use 400 boundary points and $N = 3 \times 10^3$ interior collocation points. For spatial interpolation, the residual collocation set is divided into a fixed subset of size $\lfloor (1-\rho)N_r \rfloor$ and a randomly refreshed subset of size $\lfloor \rho N_r \rfloor$, with

$$\rho \in \{0,\ 0.01,\ 0.1,\ 0.3,\ 0.5,\ 0.7,\ 0.9,\ 0.99,\ 1.0\}.$$

Here, $\rho = 0$ recovers fixed sampling, while $\rho = 1$ recovers random resampling.



(a) $c = 4$ (b) $c = 40$

Figure 18: Mode-wise coefficient error under spatial interpolation on the two-mode Poisson equation for different values of ρ.

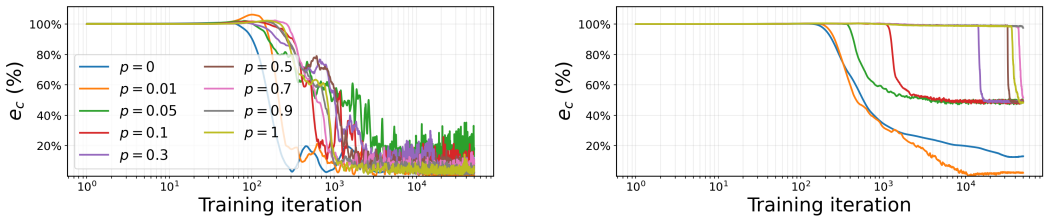
Figure 18 reports the full sensitivity to ρ. Reducing the refreshed fraction generally accelerates recovery of the high-frequency mode, showing that partial reuse of collocation points can mitigate the delay induced by random resampling. The endpoint $\rho = 0$, however, corresponds to fully fixed sampling and can inherit collocation bias.

For temporal interpolation, we set $\rho = 1$ and refresh the full residual collocation set with probability p at each iteration, with

$$p \in \{0,\ 0.01,\ 0.05,\ 0.1,\ 0.3,\ 0.5,\ 0.7,\ 0.9,\ 1.0\}.$$

Here, $p = 0$ recovers fixed sampling, while $p = 1$ recovers random resampling.

Figure 19 reports the corresponding sensitivity to p. The dependence on p is less gradual than that observed for ρ. For the high-frequency mode, very small refresh probabilities such as $p = 0.01$ lead to substantially faster recovery, whereas larger values of p often remain close to the random resampling behavior or plateau at high error. The endpoint $p = 0$ recovers fixed sampling and therefore follows a distinct trajectory.



(a) $c = 4$ (b) $c = 40$

Figure 19: Mode-wise coefficient error under temporal interpolation on the two-mode Poisson equation for different values of p.

Algorithm 1 Adaptive interpolation scheduling

Require: Decay factor $\gamma \in (0, 1)$, update interval K, number of iterations T
Initialize $\rho \leftarrow 1$
for $k = 1, \ldots, T$ **do**
  Perform one training step using spatial interpolation with the current ρ
  if $k \bmod K = 0$ **then**
    Estimate the residual gradient SNR in eq. (13)
    if $\widehat{\mathrm{SNR}}(\theta_k) < 1$ **then**
      $\rho \leftarrow \gamma\rho$
    end if
  end if
end for

G.7 Adaptive Interpolation Scheduling

The interpolation parameters in the main experiments are selected from a grid based on the final error with respect to the exact solution. In practical settings, however, the exact solution is unavailable, so this selection rule cannot be applied. We show that the residual gradient SNR introduced in Section 4.2 provides a training signal for adaptively controlling the degree of collocation resampling without such oracle access.

The key observation is that $\mathrm{SNR} < 1$ indicates the noise-dominated regime, in which the residual gradient signal is weaker than the stochastic variation induced by resampling. We therefore begin from standard random resampling and progressively increase collocation reuse whenever the estimated SNR falls below this threshold. For spatial interpolation, we initialize $\rho = 1$ and, every K iterations, update $\rho \leftarrow \gamma\rho$ if $\widehat{\mathrm{SNR}}(\theta_k) < 1$, where $\gamma \in (0, 1)$ is a decay factor. Otherwise, ρ remains unchanged. Algorithm 1 summarizes the procedure. The same rule applies to temporal interpolation by replacing ρ with the resampling probability p. Because the SNR is estimated solely from residual gradients on sampled collocation sets, the procedure requires neither the exact solution nor modal error information.

We evaluate the scheduler on all six benchmark equations with $\gamma = 0.8$ shared across all equations, and estimate the SNR as described in Appendix C.3. Table 6 compares the scheduler with fixed sampling, random resampling, and spatial interpolation with prespecified ρ. In e_{H^1}, the scheduler improves upon random resampling on all six benchmarks and ranks first or second on five of them. In e_{L^2}, it improves upon random resampling on all benchmarks except Helmholtz. Since no configuration is selected using the exact solution, these results indicate that the SNR diagnostic yields a practical rule for setting the interpolation level without oracle tuning.

G.8 Combining Interpolation with Existing Sampling Methods

The interpolation schemes in Section 5.2 control how much and how often the residual collocation set is refreshed, but they do not specify where new points are placed. They can therefore be combined with existing sampling methods that determine the placement of new points. We examine this compatibility with two representative methods, R3 (Daw et al., 2023) and LAS (Jeong et al., 2025).

Table 6: Comparison of adaptive interpolation scheduling with fixed sampling, random resampling, and spatial interpolation with prespecified ρ. Results are reported as mean $\pm$ standard deviation over five random seeds. The best result is shown in **bold**, and the second-best result is underlined.

				Spatial interpolation			
Equation	Metric	Adaptive	Fixed Sampling	$\rho = 0.3$	$\rho = 0.5$	$\rho = 0.7$	Random Resampling
Poisson	e_{H^1}	$\mathbf{0.229}_{\pm 0.028}$	$0.498_{\pm 0.176}$	$0.371_{\pm 0.074}$	$\underline{0.241}_{\pm 0.053}$	$0.456_{\pm 0.167}$	$0.733_{\pm 0.018}$
	e_{L^2}	$\mathbf{0.168}_{\pm 0.023}$	$0.586_{\pm 0.407}$	$0.295_{\pm 0.115}$	$\underline{0.219}_{\pm 0.088}$	$0.721_{\pm 0.365}$	$0.377_{\pm 0.109}$
Heat	e_{H^1}	$0.462_{\pm 0.100}$	$\mathbf{0.299}_{\pm 0.088}$	$0.581_{\pm 0.010}$	$0.483_{\pm 0.130}$	$\underline{0.325}_{\pm 0.163}$	$0.602_{\pm 0.028}$
	e_{L^2}	$0.224_{\pm 0.063}$	$\mathbf{0.097}_{\pm 0.032}$	$0.276_{\pm 0.041}$	$0.185_{\pm 0.058}$	$\underline{0.147}_{\pm 0.070}$	$0.321_{\pm 0.073}$
Helmholtz	e_{H^1}	$\underline{0.167}_{\pm 0.043}$	$1.812_{\pm 0.534}$	$\mathbf{0.160}_{\pm 0.055}$	$0.172_{\pm 0.053}$	$0.185_{\pm 0.045}$	$0.775_{\pm 0.004}$
	e_{L^2}	$0.279_{\pm 0.139}$	$2.735_{\pm 0.981}$	$\mathbf{0.239}_{\pm 0.087}$	$0.254_{\pm 0.143}$	$0.358_{\pm 0.229}$	$\underline{0.245}_{\pm 0.030}$
Darcy	e_{H^1}	$\underline{0.192}_{\pm 0.214}$	$0.655_{\pm 0.274}$	$0.274_{\pm 0.175}$	$0.226_{\pm 0.055}$	$\mathbf{0.150}_{\pm 0.075}$	$0.558_{\pm 0.063}$
	e_{L^2}	$\mathbf{0.110}_{\pm 0.115}$	$0.939_{\pm 0.381}$	$0.169_{\pm 0.110}$	$0.187_{\pm 0.102}$	$\underline{0.118}_{\pm 0.077}$	$0.178_{\pm 0.073}$
Klein–Gordon	e_{H^1}	$\underline{0.518}_{\pm 0.182}$	$1.487_{\pm 1.360}$	$\mathbf{0.474}_{\pm 0.192}$	$0.666_{\pm 0.093}$	$0.780_{\pm 0.416}$	$0.791_{\pm 0.123}$
	e_{L^2}	$\underline{2.431}_{\pm 0.825}$	$4.396_{\pm 3.577}$	$\mathbf{1.683}_{\pm 0.833}$	$2.572_{\pm 0.658}$	$5.194_{\pm 3.048}$	$2.691_{\pm 1.091}$
Convection	e_{H^1}	$\underline{0.401}_{\pm 0.246}$	$3.267_{\pm 1.176}$	$\mathbf{0.188}_{\pm 0.089}$	$0.427_{\pm 0.399}$	$0.961_{\pm 0.530}$	$0.965_{\pm 0.053}$
	e_{L^2}	$\underline{0.313}_{\pm 0.145}$	$2.140_{\pm 0.782}$	$\mathbf{0.301}_{\pm 0.098}$	$0.327_{\pm 0.265}$	$0.546_{\pm 0.308}$	$0.582_{\pm 0.164}$

Table 7: Relative H^1 seminorm error e_{H^1} and relative L^2 error e_{L^2} when interpolation is combined with R3 and LAS. Results are reported as mean $\pm$ standard deviation over five random seeds. Within each base method, the best result is shown in **bold** and the second best is underlined.

	Poisson		Heat		Helmholtz		Darcy		Klein–Gordon		Convection	
Method	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}	e_{H^1}	e_{L^2}
R3	$0.603_{\pm 0.032}$	$0.697_{\pm 0.250}$	$1.213_{\pm 0.229}$	$2.163_{\pm 1.499}$	$0.459_{\pm 0.041}$	$0.955_{\pm 0.236}$	$0.559_{\pm 0.192}$	$0.600_{\pm 0.379}$	$1.302_{\pm 0.570}$	$10.150_{\pm 4.469}$	$\underline{1.025}_{\pm 0.028}$	$\mathbf{0.509}_{\pm 0.021}$
+ Spatial	$0.271_{\pm 0.207}$	$\underline{0.414}_{\pm 0.415}$	$1.018_{\pm 0.072}$	$0.927_{\pm 0.275}$	$0.112_{\pm 0.051}$	$0.290_{\pm 0.201}$	$\underline{0.151}_{\pm 0.058}$	$0.314_{\pm 0.196}$	$\mathbf{0.473}_{\pm 0.447}$	$\underline{3.763}_{\pm 4.078}$	$1.627_{\pm 0.747}$	$0.733_{\pm 0.252}$
+ Temporal	$\underline{0.236}_{\pm 0.186}$	$\mathbf{0.313}_{\pm 0.253}$	$\mathbf{0.987}_{\pm 0.033}$	$\underline{0.893}_{\pm 0.300}$	$\mathbf{0.050}_{\pm 0.011}$	$\mathbf{0.058}_{\pm 0.028}$	$0.174_{\pm 0.289}$	$\underline{0.135}_{\pm 0.221}$	$0.692_{\pm 0.559}$	$5.585_{\pm 4.540}$	$1.714_{\pm 1.334}$	$0.669_{\pm 0.455}$
+ Spatio-Temporal	$\mathbf{0.224}_{\pm 0.256}$	$0.548_{\pm 0.788}$	$\underline{0.988}_{\pm 0.041}$	$\mathbf{0.754}_{\pm 0.107}$	$\underline{0.060}_{\pm 0.018}$	$\underline{0.087}_{\pm 0.038}$	$\mathbf{0.044}_{\pm 0.008}$	$\mathbf{0.040}_{\pm 0.009}$	$\underline{0.645}_{\pm 0.613}$	$\mathbf{3.339}_{\pm 2.270}$	$\mathbf{0.928}_{\pm 1.112}$	$\underline{0.591}_{\pm 0.156}$
LAS	$0.766_{\pm 0.014}$	$0.530_{\pm 0.076}$	$0.650_{\pm 0.035}$	$0.314_{\pm 0.101}$	$0.637_{\pm 0.097}$	$1.127_{\pm 0.433}$	$0.661_{\pm 0.086}$	$0.536_{\pm 0.299}$	$0.799_{\pm 0.037}$	$\underline{2.160}_{\pm 1.106}$	$0.706_{\pm 0.406}$	$0.502_{\pm 0.082}$
+ Spatial	$0.334_{\pm 0.134}$	$0.513_{\pm 0.424}$	$\underline{0.556}_{\pm 0.114}$	$\underline{0.226}_{\pm 0.071}$	$0.237_{\pm 0.103}$	$0.430_{\pm 0.218}$	$0.355_{\pm 0.091}$	$0.301_{\pm 0.130}$	$\underline{0.689}_{\pm 0.148}$	$2.633_{\pm 1.130}$	$0.451_{\pm 0.317}$	$0.431_{\pm 0.105}$
+ Temporal	$\mathbf{0.221}_{\pm 0.074}$	$\mathbf{0.255}_{\pm 0.141}$	$0.699_{\pm 0.050}$	$0.317_{\pm 0.070}$	$\underline{0.157}_{\pm 0.048}$	$\underline{0.343}_{\pm 0.109}$	$\mathbf{0.067}_{\pm 0.016}$	$\mathbf{0.057}_{\pm 0.027}$	$0.777_{\pm 0.081}$	$2.488_{\pm 0.675}$	$\mathbf{0.104}_{\pm 0.046}$	$\mathbf{0.117}_{\pm 0.053}$
+ Spatio-Temporal	$\underline{0.311}_{\pm 0.070}$	$\underline{0.351}_{\pm 0.155}$	$\mathbf{0.477}_{\pm 0.156}$	$\mathbf{0.219}_{\pm 0.080}$	$\mathbf{0.130}_{\pm 0.035}$	$\mathbf{0.309}_{\pm 0.183}$	$\underline{0.073}_{\pm 0.014}$	$\underline{0.058}_{\pm 0.021}$	$\mathbf{0.557}_{\pm 0.249}$	$\mathbf{1.889}_{\pm 1.121}$	$\underline{0.199}_{\pm 0.136}$	$\underline{0.245}_{\pm 0.196}$

Both methods already reuse collocation points across iterations in an implicit form. R3 retains points whose residual exceeds the mean residual and resamples the remaining points uniformly, so a subset of the collocation set persists across iterations. LAS moves existing points by Langevin dynamics, so consecutive collocation sets remain close when the step size is small. In both methods, however, the degree of reuse follows from the residual distribution or the Langevin step size. Neither method exposes it as a parameter that controls the trade-off between collocation bias and resampling variance.

We keep each base method unchanged as the rule that generates new points. Whenever the interpolation scheme refreshes collocation points, the base method generates the replacements, while the interpolation scheme determines which points are refreshed and how often.

Table 7 shows that interpolation improves both base methods on most benchmarks. With LAS, spatial and spatio-temporal interpolation reduce both e_{H^1} and e_{L^2} on all six benchmarks. With R3, all three variants reduce both errors on Poisson, Heat, Helmholtz, and Darcy. These results indicate that explicit control of collocation reuse complements the placement criteria of existing sampling methods.