# Gradient Alignment in Physics-informed Neural Networks: A Second-Order Optimization Perspective

**Sifan Wang**[1]*, **Ananyae Kumar Bhartari**[2]*, **Bowen Li**[3]*, **Paris Perdikaris**[4]
[1]Institution for Foundation of Data Science, Yale University
[2]Penn Institute for Computational Science, University of Pennsylvania
[3]Department of Mathematics, City University of Hong Kong
[4]Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania
sifan.wang@yale.edu, bowen.li@cityu.edu.hk, {ananyaeb, pgp}@seas.upenn.edu

## Abstract

Physics-informed neural networks (PINNs) have shown significant promise in computational science and engineering, yet they often face optimization challenges and limited accuracy. In this work, we identify directional gradient conflicts during PINN training as a critical bottleneck. We introduce a novel gradient alignment score to systematically diagnose this issue through both theoretical analysis and empirical experiments. Building on these insights, we show that (quasi) second-order optimization methods inherently mitigate gradient conflicts, thereby consistently outperforming the widely used Adam optimizer. Among them, we highlight the effectiveness of SOAP [1] by establishing its connection to Newton's method. Empirically, SOAP achieves state-of-the-art results on 10 challenging PDE benchmarks, including the first successful application of PINNs to turbulent flows at Reynolds numbers up to 10,000. It yields 2–10x accuracy improvements over existing methods while maintaining computational scalability, advancing the frontier of neural PDE solvers for real-world, multi-scale physical systems. All code and datasets used in this work are publicly available at: https://github.com/PredictiveIntelligenceLab/jaxpi/tree/pirate.

## 1 Introduction

Physics-informed neural networks (PINNs) have emerged as a powerful paradigm in scientific machine learning by incorporating physical principles through carefully designed loss functions. These loss functions act as soft constraints, guiding neural networks to learn solutions that respect underlying physical laws while simultaneously fitting experimental data. The elegance and versatility of PINNs have led to their widespread adoption in solving both forward and inverse problems involving partial differential equations (PDEs). Their success spans numerous domains in computational science, from fluid mechanics [2, 3, 4, 5], heat transfer [6, 7, 8] to bio-engineering [9, 10, 11] and materials science [12, 13, 14]. The impact of PINNs extends even further, with significant applications in electromagnetics [15, 16, 17], geosciences [18, 19, 20], etc.

Despite their broad applications, PINNs currently face limitations in convergence speed and accuracy that affect their reliability as forward PDE solvers. This has motivated extensive research efforts to enhance their performance through various methodological innovations. Significant advances have emerged in neural architecture design, including novel network backbones [21, 22, 23, 24, 25, 26, 27], improved activation functions [28, 29], and effective coordinate embeddings [30, 31, 32, 33]. Other improvements have focused on optimizing the training process through enhanced collocation point sampling strategies [34, 35, 36], more efficient optimizers [37, 38, 39, 40], and

*These authors contributed equally to this work.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

advanced training strategies such as sequential training [41, 42, 43] and transfer learning [44, 45, 46]. Researchers have also explored alternative formulations of the learning objective, incorporating numerical differentiation [47], variational principles inspired by Finite Element Methods [48, 49], and specialized regularization terms [50, 51].

A particularly active area of research has centered on addressing gradient pathologies during training [21, 52]. One prominent issue involves imbalanced backpropagated gradients across different loss terms, leading to significant discrepancies in convergence rates and reduced solution accuracy, especially in complex physical systems. This has led to the development of various adaptive weighting strategies [21, 52, 53, 54, 55, 56, 57]. However, the equally critical issue of directional gradient conflicts – where gradients from different losses point in conflicting directions – remains largely underexplored [58, 59]. Our work aims to bridge this gap by systematically investigating, analyzing, and resolving these directional conflicts in PINNs training. The key contributions of this work are summarized as follows:

- We introduce a novel gradient alignment metric that extends cosine similarity to quantify directional conflicts between multiple loss terms.
- We demonstrate that gradient conflicts hinder PINN training, with higher conflict scores linked to slower convergence in various PDE systems.
- We show that second-order optimizers enhance gradient alignment by implicitly preconditioning the loss landscape.
- We reveal that SOAP [1] can be viewed as an efficient approximation of the Newton preconditioner, revolving gradient conflicts.
- We provide comprehensive experimental results across 10 PDE benchmarks, including the first successful PINN application to turbulent flows with Reynolds numbers up to 10,000, achieving 2-10x accuracy improvements.

Taken together, this work advances our understanding of optimization dynamics in PINNs while demonstrating how quasi second-order methods can enable more reliable neural PDE solvers for solving complex physical systems. These insights pave the way for developing next-generation optimizers for physics-informed machine learning, and beyond.

## 2 Overview of PINNs

Multi-task learning in deep neural networks requires simultaneously minimizing multiple competing objectives – a challenge that manifests acutely in physics-informed neural networks (PINNs). Building upon the work of [60], PINNs approximate solutions to partial differential equations by minimizing a composite loss function that enforces both physical constraints and data-fitting objectives. Consider a general PDE system:

$$\mathbf{u}_t + \mathcal{N}[\mathbf{u}] = 0, \;\; t \in [0, T], \; \mathbf{x} \in \Omega, \tag{2.1}$$

with inital and boundary conditions

$$\mathbf{u}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}), \;\; \mathbf{x} \in \Omega, \tag{2.2}$$

$$\mathcal{B}[\mathbf{u}] = 0, \;\; t \in [0, T], \; \mathbf{x} \in \partial\Omega, \tag{2.3}$$

where $\mathcal{N}[\cdot]$ represents a differential operator and $\mathcal{B}[\cdot]$ denotes boundary conditions. The core idea of PINNs is to approximate the solution $\mathbf{u}(t, \mathbf{x})$ using a neural network $\mathbf{u}_\theta(t, \mathbf{x})$. Through automatic differentiation [61], we can compute the PDE residual:

$$\mathcal{R}[u_\theta](t, \mathbf{x}) = \frac{\partial \mathbf{u}_\theta}{\partial t}(t_r, \mathbf{x}_r) + \mathcal{N}[\mathbf{u}_\theta](t_r, \mathbf{x}_r). \tag{2.4}$$

This leads to a composite loss function that encapsulates multiple competing objectives:

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} \left| \mathbf{u}_\theta \left(0, \mathbf{x}_{ic}^i\right) - \mathbf{g}\left(\mathbf{x}_{ic}^i\right) \right|^2}_{\mathcal{L}_{ic}(\theta)} + \underbrace{\frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} \left| \mathcal{B}\left[\mathbf{u}_\theta\right]\left(t_{bc}^i, \mathbf{x}_{bc}^i\right) \right|^2}_{\mathcal{L}_{bc}(\theta)} + \underbrace{\frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{R}\left[\mathbf{u}_\theta\right]\left(t_r^i, \mathbf{x}_r^i\right) \right|^2}_{\mathcal{L}_r(\theta)}. \tag{2.5}$$

These loss functions aim to fit the initial, boundary conditions and the PDE residuals, respectively. And $\{\mathbf{x}_{ic}^i\}_{i=1}^{N_{ic}}$, $\{t_{bc}^i, \mathbf{x}_{bc}^i\}_{i=1}^{N_{bc}}$ and $\{t_r^i, \mathbf{x}_r^i\}_{i=1}^{N_r}$ may be selected either as fixed mesh vertices or through random sampling during each training iteration.

## 3 Gradient Alignment in PINNs

A fundamental challenge in training PINNs is that different loss terms often conflict during optimization. As illustrated in Figure 1, PINNs encounter two modes of gradient conflict during training. The first, identified by [21, 52], involves significant imbalances in gradient magnitudes. In such cases, dominant loss terms overwhelm others, often leading to model failure. Various self-adaptive weighting strategies have been proposed to address this issue [53, 54, 55, 56, 57].

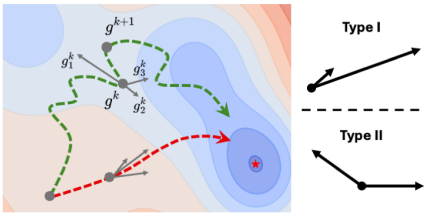


Figure 1: Gradient conflicts and their impact on PINNs optimization. The irregular green trajectory illustrates how the optimization struggles when facing two types of gradient conflicts: Type I, where gradients have similar directions but vastly different magnitudes, and Type II, where gradients have similar magnitudes but opposing directions. The red trajectory shows how appropriate preconditioning through second-order information could mitigate these conflicts by aligning gradients both within and between optimization steps, enabling efficient convergence.

This second mode arises when gradients from different loss terms point in conflicting directions, forcing optimization into inefficient, compromised trajectories. For example, in the Navier–Stokes equations, enforcing no-slip boundary conditions demands precise control of velocity gradients near walls, which can conflict with preserving mass and momentum conservation in the bulk flow. First-order optimizers like gradient descent or Adam must follow the average gradient direction, resulting in inefficient zigzagging between competing objectives. The severity of these conflicts increases with problem complexity, becoming particularly acute for turbulent flows where maintaining physical constraints across multiple scales is crucial. To better understand and address these directional gradient conflicts, we introduce the alignment score, defined as follows.

**Definition 1.** *Suppose that $v_1, v_2, \ldots, v_n$ are vectors, then the alignment score is defined as*

$$\mathcal{A}(v_1, v_2, \ldots, v_n) = 2 \left\| \frac{\sum_{i=1}^n \frac{v_i}{\|v_i\|}}{n} \right\|^2 - 1. \tag{3.1}$$

This score ranges from $[-1, 1]$ and naturally extends the concept of cosine similarity to multiple vectors. As illustrated in Proposition 1, for the special case of $n = 2$, our score exactly recovers the standard cosine similarity $\cos(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|}$, where 1 indicates perfect alignment, 0 suggests orthogonal directions, and -1 represents complete opposition.

**Proposition 1.** *For n=2, the alignment score $\mathcal{A}(v_1, v_2)$ equals the cosine similarity between $v_1$ and $v_2$:*

$$\mathcal{A}(v_1, v_2) = \cos(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|}. \tag{3.2}$$

The proof is provided in Appendix F.1. The alignment score enables us to quantify both the local conflicts between individual loss terms within each gradient descent step and the global conflicts across consecutive steps. Formally:

**Definition 2.** *Let $\mathcal{L} = \sum_{i=1}^n \mathcal{L}_i$ be a composite loss function. At the $k$-th step of gradient descent, let $g^k$ denote the full gradient and $g_1^k, g_2^k, \ldots, g_n^k$ denote the gradients of individual loss terms. We define:*

*(a) The intra-step gradient alignment score:*

$$\mathcal{A}_{intra}^k = \mathcal{A}(g_1^k, g_2^k, \ldots, g_n^k). \tag{3.3}$$

*(b) The inter-step gradient alignment score:*

$$\mathcal{A}_{inter}^k = \mathcal{A}(g^{k-1}, g^k). \tag{3.4}$$

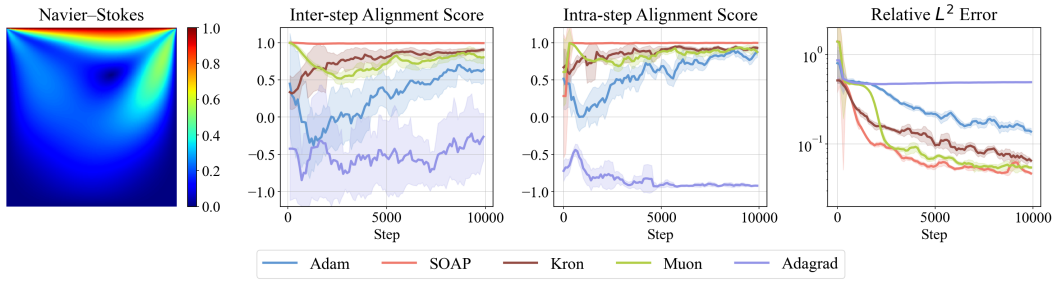


Figure 2: Gradient alignment scores and test errors during PINN training for solving the Navier-Stokes equations with different optimizers. Additional benchmarks are provided in Figure 5, where we observe the consistent phenomenon that first-order optimizers exhibit poor gradient alignment and slow convergence of test errors.

The intra-step gradient alignment score becomes especially useful for PINNs applications where the total loss typically consists of multiple terms. For example, in the case of solving the 2D Navier-Stokes equations, the loss function includes separate components corresponding to momentum equations in the $x$ and $y$ directions, the continuity equation, and boundary conditions for the velocity fields $u$ and $v$. Since these loss terms can have different scales and properties, they should be treated separately rather than grouped together. Our intra-step score effectively quantifies gradient conflicts across all these terms simultaneously.

The impact of alignment can be formalized through the following result, which shows that the rate of loss decay under preconditioned gradient descent depends on the alignment scores.

**Proposition 2.** *Let $\mathcal{L} = \sum_{i=1}^{n} \mathcal{L}_i : \mathbb{R}^d \to \mathbb{R}$ be $L$-smooth (w.r.t. the Euclidean norm). Consider preconditioned gradient descent*

$$\theta_{t+1} = \theta_t - \eta h_t, \qquad h_t = P_t g_t, \qquad g_t = \sum_{i=1}^{n} g_t^i,$$

*where each $P_t \succ 0$ satisfies $\mu \le \lambda_{\min}(P_t) \le \lambda_{\max}(P_t) \le M$. Let $\Delta_t := \mathcal{L}(\theta_t) - \mathcal{L}(\theta_{t+1})$ and assume $\eta \le 1/(LM)$. Then:*

*(i)* ***Single-step drop.*** *For all $t$, if $\|h_t^i\| = \lambda$ for all $i$,*

$$\Delta_t \;\ge\; \left(\tfrac{\eta}{M} - \tfrac{L\eta^2}{2}\right) \frac{n}{2} \left(A_t^{\text{intra}} + 1\right) \sum_{i=1}^{n} \|h_t^i\|^2.$$

*(ii)* ***Two-step cumulative drop.*** *Let $a := \|h_{t-1}\|$, $b := \|h_t\|$. Then*

$$\Delta_{t-1} + \Delta_t \;\ge\; \tfrac{\eta}{M}\Big(1 - \tfrac{L\eta M}{2}\Big) a^2 + \tfrac{\eta}{M}\left(1 - L\eta M\right) ab\, A_t^{\text{inter}} - \tfrac{L\eta^2}{2}\, b^2.$$

This result shows that higher intra- and inter-step alignment scores directly accelerate loss reduction. The assumption $\left|h_t^k\right| = \lambda$ for all $k$ is not restrictive in the PINNs setting, where balancing the scales of different losses is common practice. In fact, we adopt the weighting scheme proposed by Wang et al. [21] (Appendix G.2), which ensures that all weighted gradients have the same norm.

In the following, we will demonstrate that gradient direction conflicts widely exist in training PINNs, especially in the early stages of training. To this end, we conduct experiments on five representative PDEs spanning from linear wave propagation to reaction-diffusion systems like the Ginzburg-Landau equation and fluid dynamics governed by the Navier-Stokes equations. The detailed results are presented in Figure 5 and corresponding experimental setup is provided in Appendix G.

Figure 2 presents gradient alignment scores and test errors during PINN training for solving the Navier-Stokes equations with different optimizers. Importantly, we compute these scores from the gradients after applying each optimizer's gradient transformations, which captures the actual update directions and their degree of alignment or conflict. We observe that the scores from first order optimizer oscillate significantly near or below zero at early stages. It provides strong evidence for persistent directional conflicts between gradients throughout the training process. Intuitively, these

conflicting gradients force the network parameters to follow an inefficient zigzag trajectory in the loss landscape, significantly impeding convergence speed.

In contrast, the quasi second-order optimizers (e.g., Muon [62], SOAP [1], Kron [63]) consistently maintains the higher positive values for both inter-step and intra-step gradient alignment scores throughout training. This effective resolution of gradient direction conflicts directly corresponds to significantly faster convergence in test error. It is worth nothing that SOAP achieves the highest alignment scores among all optimizers, demonstrating its superior effectiveness. In the following sections, we provide a theoretical explanation for why PINNs inherently suffer from directional gradient conflicts and establish a formal connection between SOAP and Newton's method, offering deeper insight into its empirical success.

## 4 Intra-step Gradient Alignment of PINNs at Initialization

In this section, we show that gradient conflicts are intrinsic to PINNs near initialization, regardless of the optimizer used. For simplicity, we analyze the intra-step gradient alignment for a two-layer neural network solving the one-dimensional Laplace equation under small weight initialization. This setup allows for a tractable analysis while capturing key phenomena, and the results naturally extend to more general PDEs. Following the setup in Section 2, and without loss of generality, we consider the 1D Laplace equation:

$$\begin{cases} \Delta u = u'' = 0 \quad \text{on } [-1,1], \\ u(\pm 1) = g_{\pm 1}. \end{cases} \tag{4.1}$$

We approximate the solution $u(x)$ by a two-layer network with width $N$:

$$u(x,\theta) = \sum_{i=1}^{N} a_i \sigma(w_i x) = \mathbf{a} \cdot \sigma(\mathbf{w} x)\,, \tag{4.2}$$

where $\mathbf{a} = (a_1, \ldots, a_N), \mathbf{w} = (w_1, \ldots, w_N) \in \mathbb{R}^N$, and $\theta = (\mathbf{a}, \mathbf{w}) \in \mathbb{R}^{2N}$. Moreover, we limit ourselves to the activation function $\sigma(x) = \tanh(x)$. In this case, the loss (2.5) reduces to

$$\min_{\theta=(\mathbf{a},\mathbf{w})} \mathcal{L}(\theta) = \underbrace{\frac{1}{N_r} \sum_{p=1}^{N_r} |u''(x_p,\theta)|^2}_{\mathcal{L}_r(\theta)} + \underbrace{\frac{1}{2} \sum_{s=\pm 1} |u(s,\theta) - g_s|^2}_{\mathcal{L}_{bc}(\theta)}\,. \tag{4.3}$$

**Proposition 3.** *At initialization, we assume that the weights $a_i, w_i$ are initialized by i.i.d. Gaussian $\mathcal{N}(0,\varepsilon^2)$ with small $\varepsilon = o(1)$. Then the alignment score converges to a binary random variable in the infinite width limit:*

$$\lim_{N\to\infty} \mathcal{A}(\square(\nabla\mathcal{L}_b), \square(\nabla\mathcal{L}_r)) = O(\varepsilon^2) + C_\square \begin{cases} \operatorname{sgn}(g_{-1} - g_1) & \text{with prob. } \frac{1}{2}, \\ -\operatorname{sgn}(g_{-1} - g_1) & \text{with prob. } \frac{1}{2}. \end{cases} \tag{4.4}$$

*where $\square = \mathrm{GD}, \mathrm{Adam}, \mathrm{Shampoo},$ or $\mathrm{Soap}$ denotes the corresponding optimizer update rule, and $C_\square$ is a constant depending on the optimizer.*

The proof is provided in Appendix B. As a direct implication, the intra-step alignment score exhibits no preferred direction. Consequently, all optimizers fail to induce consistent intra-step alignment between $\nabla\mathcal{L}_b$ and $\nabla\mathcal{L}_r$ near initialization, as empirically validated in Figure 5. In contrast, SOAP maintains a perfect inter-step alignment score of 1 throughout the entire training process. In the following section, we provide a theoretical explanation for this distinctive behavior.

## 5 Gradient Alignment in Quasi Second-Order Optimization

In this section, we aim to reveal why quasi Second-order optimizers can naturally promote gradient alignment via the following lemma.

**Proposition 4.** *Let $\mathcal{L}(\theta)$ be a twice differentiable loss function with Hessian $H(\theta)$, and let $P(\theta)$ be a preconditioner with uniformly bounded inverse $P^{-1}(\theta)$. Consider the preconditioned gradient descent update with exponent $0 \le s \le 1$ and learning rate $\eta > 0$:*

$$\theta_{t+1} = \theta_t - \eta P^{-s}(\theta_t) \nabla\mathcal{L}(\theta_t). \tag{5.1}$$

*Let $g_t = \nabla \mathcal{L}(\theta_t)$, then the alignment score $\mathcal{A}(g_t, g_{t+1})$ satisfies:*

$$\mathcal{A}(g_t, g_{t+1}) = 1 - \frac{\eta^2}{2} \frac{\|HP^{-s} g_t\|^2}{\|g_t\|^2} + O(\eta^3). \tag{5.2}$$

*To maintain alignment $\mathcal{A}(g_t, g_{t+1}) \geq 1 - \epsilon$ for $\epsilon > 0$, the learning rate $\eta$ must satisfy:*

$$\eta \leq \sqrt{\frac{2\epsilon \|g_t\|^2}{\|HP^{-s} g_t\|^2}}. \tag{5.3}$$

The proof is presented in Appendix C. This bound specializes to the following cases: For vanilla gradient descent ($P = I$), $\eta_{\max} = \sqrt{\frac{2\epsilon}{\lambda^2_{\max}(H)}}$, and for Newton's method ($P = H, s = 1$), the maximum learning rate can be relaxed to $\eta_{\max} = \sqrt{2\epsilon}$. These results imply that preconditioners approximating the Hessian effectively relax learning rate constraints, enabling the use of larger learning rates while maintaining optimization trajectory consistency. Comparing these cases reveals that Newton's method eliminates the dependency on the condition number of the Hessian, allowing for a constant maximum learning rate regardless of problem conditioning. This finding aligns with our theoretical understanding that accurate second-order information mitigates the effects of ill-conditioning, allowing for stable optimization even with aggressive learning rates.

Next, we establish that under some assumptions, the SOAP optimizer [1] can be interpreted as approximating to using Hessian as the preconditioner:

$$w_{t+1} = w_t - \eta \operatorname{Soap}(g_t) \approx w_t - \eta H^{-1} g_t. \tag{5.4}$$

Formal assumptions and proof are provided in Appendix D.

Table 1: Comparison of optimization methods showing preconditioner types, storage and computational complexity for a $n \times n$ weight matrix, and practical compatibility with mini-batches and scalability with large neural networks. The theoretical connections between Shampoo, Muon, and quasi-second-order methods are detailed in Appendix D and E.

| **Method** | **(Approx.) Precond.** | **Storage** | **Computation** | **Mini-batch** | **DNN** |
|---|---|---|---|---|---|
| Natural Gradient [64, 37] | $F^{-1}$ | $O(n^4)$ | $O(n^6)$ | ✗ | ✗ |
| BFGS/L-BFGS [65] | $H^{-1}$ | $O(n^2)$ | $O(n^2)$ | ✗ | ✓ |
| SOAP [1] | $H^{-1}$ | $O(n^2)$ | $O(n^3)$ | ✓ | ✓ |
| Kron [63] | $F^{-1/2}$ | $O(n^2)$ | $O(n^3)$ | ✓ | ✓ |
| Shampoo [66] | $H_{\text{ada}}^{-1/2}$ | $O(n^2)$ | $O(n^3)$ | ✓ | ✓ |
| Muon [62] | $H_{\text{ada}}^{-1/2}$ | $O(n^2)$ | $O(n^3)$ | ✓ | ✓ |
| ConFig [58] | N/A | $O(n^2)$ | $O(n^2)$ | ✓ | ✓ |
| DCGD [59] | N/A | $O(n^2)$ | $O(n^2)$ | ✓ | ✓ |

In Table 1, we compare various popular methods to tackle the identified directional gradient conflicts in the context of PINNs. While all these methods are theoretically promising, some face practical limitations in the context of complex PDE systems. L-BFGS is unsuitable for large-scale or stochastic training, as gradient noise disrupts its Hessian updates and line search procedures. First-order variants such as ConFiG [58] and DCGD [59] attempt to alleviate conflicts via gradient surgery or projection. These methods yield incremental improvements but remain constrained by the inherent limitations of first-order updates.

Natural gradient descent (NGD) requires computing and inverting the Fisher information matrix at every iteration and is restricted to `float64` precision, which is inefficient on GPUs and consumes 2× more memory while being 2-4x slower than `float32`. As a result, NGD has only been demonstrated on relatively simple benchmarks with smooth solutions, where very small networks suffice and convergence issues rarely appear. On more challenging PDEs with sharp transitions or complex dynamics, NGD fails to scale: it is highly sensitive to hyperparameters, lacks mini-batching support, and often diverges.

To illustrate the limitations of NGD, we revisited the 2D Poisson and heat benchmarks of [37], using their official implementation and comparing against SOAP under both `float32` and `float64`

precision. We tested MLPs of varying depth and width, reporting the best results across ten random seeds for each method. While SOAP reliably converged, NGD exhibited some failures for some random seeds, which is also acknowledged in [37]. Even under successful runs, NGD underperforms SOAP in both accuracy and stability, as summarized in Table 2.

Table 2: Comparison of relative $L^2$ errors for different MLP architectures and optimization methods (NGD vs. SOAP) on Poisson and Heat 2D PDE benchmarks, evaluated in both `float32` and `float64` precision.

| PDE | Architecture (Method) | Float32 | Float64 |
|---|---|---|---|
| Poisson | [2, 32, 1] (NGD) | $4.87 \times 10^{-2} \pm 3.61 \times 10^{-2}$ | $3.84 \times 10^{-7} \pm 2.92 \times 10^{-7}$ |
| | [2, 32, 32, 1] (NGD) | $1.65 \times 10^{-1} \pm 3.93 \times 10^{-2}$ | $1.27 \times 10^{-6} \pm 2.86 \times 10^{-7}$ |
| | [2, 32, 32, 32, 1] (NGD) | $2.48 \times 10^{-1} \pm 1.75 \times 10^{-2}$ | $3.14 \times 10^{-6} \pm 5.10 \times 10^{-7}$ |
| | [2, 256, 1] (NGD) | $1.56 \times 10^{-1} \pm 6.34 \times 10^{-2}$ | $6.10 \times 10^{-7} \pm 1.68 \times 10^{-7}$ |
| | [2, 256, 1] (SOAP) | $3.06 \times 10^{-6} \pm 7.12 \times 10^{-7}$ | $6.08 \times 10^{-7} \pm 2.13 \times 10^{-7}$ |
| | [2, 256, 256, 1] (SOAP) | $1.87 \times 10^{-6} \pm 5.19 \times 10^{-7}$ | $4.06 \times 10^{-7} \pm 1.81 \times 10^{-7}$ |
| | [2, 256, 256, 256, 1] (SOAP) | $\mathbf{1.35 \times 10^{-6} \pm 3.45 \times 10^{-7}}$ | $\mathbf{2.99 \times 10^{-7} \pm 1.05 \times 10^{-7}}$ |
| Heat 2D | [2, 32, 1] (NGD) | $5.98 \times 10^{-2} \pm 5.46 \times 10^{-2}$ | $7.68 \times 10^{-6} \pm 1.85 \times 10^{-6}$ |
| | [2, 32, 32, 1] (NGD) | $5.95 \times 10^{-1} \pm 2.29 \times 10^{-4}$ | $2.32 \times 10^{-6} \pm 1.17 \times 10^{-6}$ |
| | [2, 32, 32, 32, 1] (NGD) | $5.95 \times 10^{-1} \pm 4.58 \times 10^{-4}$ | $5.13 \times 10^{-6} \pm 5.25 \times 10^{-7}$ |
| | [2, 256, 1] (NGD) | $5.95 \times 10^{-1} \pm 1.05 \times 10^{-3}$ | $8.69 \times 10^{-6} \pm 6.49 \times 10^{-6}$ |
| | [2, 256, 1] (SOAP) | $4.61 \times 10^{-6} \pm 8.12 \times 10^{-7}$ | $3.03 \times 10^{-6} \pm 6.06 \times 10^{-7}$ |
| | [2, 256, 256, 1] (SOAP) | $2.74 \times 10^{-6} \pm 9.52 \times 10^{-7}$ | $2.04 \times 10^{-6} \pm 4.10 \times 10^{-7}$ |
| | [2, 256, 256, 256, 1] (SOAP) | $\mathbf{2.65 \times 10^{-6} \pm 5.00 \times 10^{-7}}$ | $\mathbf{1.33 \times 10^{-6} \pm 2.61 \times 10^{-7}}$ |

In contrast, quasi-second-order methods (e.g., SOAP [1], Kron [63], Muon [62]) naturally promote directional gradient conflicts via preconditioning and mitigate ill-conditioning in the loss landscape [67] while maintaining computational tractability. As previously illustrated in Figure5, SOAP consistently achieves the highest gradient alignment scores, likely due to its closer approximation to Newton's method – an observation further supported by our theoretical analysis in Appendix D.

Building on these insights, we now present comprehensive numerical experiments that demonstrate the superior accuracy and convergence behavior of quasi-second-order methods across a diverse set of PDE benchmarks.

## 6 Experiments

To rigorously evaluate the performance of the aforementioned quasi second-order methods, we examine a diverse set of 10 representative and challenging PDEs that govern fundamental physical phenomena. These equations span wave propagation, shock formulation, chaotic systems, reaction-diffusion processes, fluid dynamics, and heat transfer. The detailed description of the problem setup, including the PDE parameters, initial and boundary conditions, numerical implementations, and supplementary visualizations, are presented in Appendix G.

**Baselines.** We focus our comparisons on PINN approaches that are scalable to large neural networks, as scalability is essential for solving realistic, large-scale physical problems. Consequently, we exclude methods based on natural gradients [37, 38] and L-BFGS variants [40], as these typically rely on full-batch training, require high-precision (e.g., float64) computation, and are limited to small network sizes—making them impractical for the complex PDE benchmarks considered in this work.

Our baseline setup is based on the current state-of-the-art training pipeline proposed by [27]. Specifically, we adopt PirateNet [27] as the backbone architecture, which is known for its stability and scalability to deeper networks. All weight matrices are initialized using random weight factorization (RWF) [69], Exact periodic boundary conditions are strictly enforced when applicable [70].

For model training, we use mini-batch gradient descent with the Adam optimizer [71], which has become the de facto standard for training PINNs due to its robust performance and computational efficiency. To improve training efficiency and robustness, we use learning rate annealing [21, 72] for loss balancing. In addition, we employ causal training [73, 72] to address causality violations

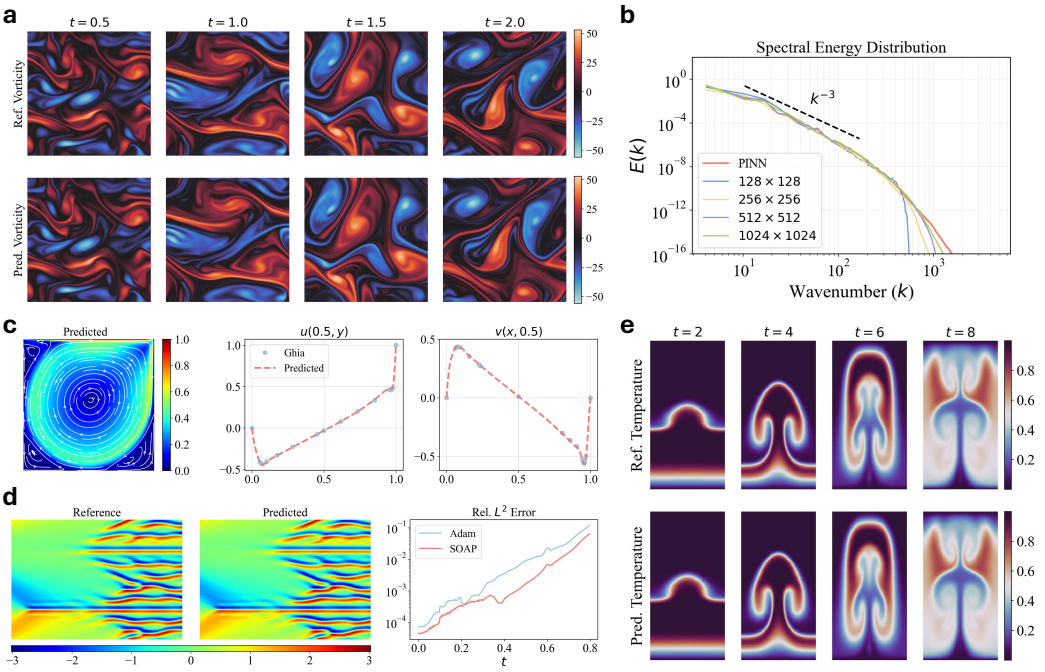


Figure 3: Simulating complex fluid dynamics using PINNs with SOAP optimization. (a) Kolmogorov flow at Re=10,000: comparison between reference solution and PINN predictions demonstrates accurate capture of turbulent structures across multiple time steps. (b) Spectral energy distribution showing PINN's superior resolution of fine-scale dynamics compared to traditional numerical solutions at various grid resolutions. (c) Lid-driven cavity flow at Re=5,000: streamlines and centerline velocity profiles show excellent agreement with benchmark data from [68]. (d) Kuramoto-Sivashinsky equation: PINNs accurately predicts complex spatiotemporal patterns and chaotic dynamics. (e) Rayleigh-Taylor instability (Pr=0.71, Ra=$10^6$): evolution of temperature field shows precise capture of interface dynamics and mushroom-shaped structures characteristic of this flow.

when solving time-dependent PDEs. For challenging benchmarks, we implement time-marching and curriculum learning strategies [42].

Importantly, this baseline represents one of the most accurate and scalable PINN pipelines currently available, achieving state-of-the-art performance across a diverse set of PDE benchmarks [72, 74]. For instance, on the standard Allen-Cahn benchmark, which has been extensively evaluated by various PINN methods, our proposed baseline outperforms most existing approaches, as demonstrated in Table 8. A comprehensive description of the techniques employed and the hyperparameter configurations are provided in Appendices G.1 and G.2, respectively.

**State-of-the-art results.** Table 3 highlights the consistent performance improvements of quasi-second-order optimizers across a wide range of PDE benchmarks. Among them, SOAP achieves the best overall results, likely due to its closer alignment with Newton's method (Table 1), making it particularly well-suited for the multi-objective nature of PINN optimization.

Compared to our baselines, SOAP reduces relative error by 6.4x on the wave equation. For nonlinear 1D problems, it achieves a 6.9x improvement on the Allen-Cahn equation, and approximately 2x improvements on both the Korteweg-de Vries and Kuramoto-Sivashinsky equations. The performance gains become particularly pronounced for coupled diffusion-reaction systems. The Grey-Scott and Ginzburg-Landau equations exhibit an order of magnitude reduction in error. On challenging Navier-Stokes benchmarks, including the lid-driven cavity and Rayleigh-Taylor instability problems, SOAP demonstrates a more than 10x error reduction. We highlight and discuss these substantial improvements in detail below.

**Complex fluid dynamics.** Our most significant achievement is successfully applying PINNs to complex fluid dynamics problems that were previously considered beyond their capabilities. In

Table 3: Comparison of optimizer performance obtained by training PINNs with Adam, Adam+L-BFGS, and SOAP, respectively, across various PDEs, following the training pipeline described in Section 6. The evaluation metric is the relative $L^2$ error over the entire spatial-temporal domain.

| Benchmark | Adam | Adam+L-BFGS | Kron | Muon | SOAP |
|---|---|---|---|---|---|
| Wave | $5.15 \times 10^{-5}$ | $5.08 \times 10^{-5}$ | $8.62 \times 10^{-6}$ | $9.34 \times 10^{-6}$ | $\mathbf{8.05 \times 10^{-6}}$ |
| Burgers | $8.20 \times 10^{-5}$ | $8.20 \times 10^{-5}$ | $4.85 \times 10^{-5}$ | $4.52 \times 10^{-5}$ | $\mathbf{4.03 \times 10^{-5}}$ |
| Allen-Cahn | $2.24 \times 10^{-5}$ | $2.25 \times 10^{-5}$ | $3.63 \times 10^{-6}$ | $4.95 \times 10^{-6}$ | $\mathbf{3.48 \times 10^{-6}}$ |
| Korteweg–De Vries | $7.04 \times 10^{-4}$ | $7.33 \times 10^{-4}$ | $5.48 \times 10^{-4}$ | $4.19 \times 10^{-4}$ | $\mathbf{3.40 \times 10^{-4}}$ |
| Kuramoto-Sivashinsky | $7.48 \times 10^{-2}$ | – | $5.49 \times 10^{-2}$ | $\mathbf{3.51 \times 10^{-2}}$ | $3.86 \times 10^{-2}$ |
| Grey-Scott | $3.61 \times 10^{-3}$ | – | $1.89 \times 10^{-4}$ | $1.95 \times 10^{-4}$ | $\mathbf{1.88 \times 10^{-4}}$ |
| Ginzburg-Landau | $1.49 \times 10^{-2}$ | – | $7.53 \times 10^{-3}$ | $\mathbf{4.58 \times 10^{-3}}$ | $4.78 \times 10^{-3}$ |
| Lid-driven cavity ($\mathrm{Re} = 5 \times 10^3$) | $3.24 \times 10^{-1}$ | – | $7.05 \times 10^{-2}$ | $6.70 \times 10^{-2}$ | $\mathbf{3.99 \times 10^{-2}}$ |
| Kolmogorov flow ($\mathrm{Re} = 10^4$) | $2.04 \times 10^{-1}$ | – | $8.62 \times 10^{-2}$ | $6.89 \times 10^{-2}$ | $\mathbf{3.20 \times 10^{-2}}$ |
| Rayleigh-Taylor instability ($\mathrm{Ra} = 10^6$) | $7.32 \times 10^{-2}$ | – | $5.74 \times 10^{-3}$ | $1.80 \times 10^{-2}$ | $\mathbf{5.22 \times 10^{-3}}$ |

particular, we demonstrate breakthrough results in three challenging cases that combine multiple physical constraints and have historically proven difficult for PINNs, see Figure 3.

For the lid-driven cavity flow at Reynolds number 5,000, SOAP enables a dramatic improvement in accuracy, reducing the relative $L^2$ error from 32.4% to 3.99%. As shown in Figure 3c, our model successfully captures intricate flow features including secondary and tertiary corner vortices, showing excellent agreement with the benchmark results of [68].

The Rayleigh-Taylor instability presents an even more challenging test, requiring simultaneous handling of interface dynamics and coupled velocity-density evolution. SOAP enables accurate prediction of the characteristic mushroom-shaped structures that develop as heavier fluid penetrates into lighter fluid, achieving a relative $L^2$ error of 0.52% – nearly an order of magnitude improvement over the best baseline's 7.32%. Figure 3e demonstrates excellent agreement with reference solutions across multiple time steps, capturing both the initial linear growth phase and subsequent nonlinear development.

Our most impressive result comes from the turbulent Kolmogorov flow at Reynolds number 10,000 – marking the first time PINNs have successfully captured turbulent dynamics at such high Reynolds numbers. Our model achieves a relative $L^2$ error of 3.20%, compared to 20.4% with our baseline. Figure 3a shows that our predictions accurately reproduce both the large-scale flow structures and the complex cascade of smaller eddies characteristic of turbulent flows. Moreover, spectral analysis reveals that our PINN solution maintains higher spectral energy at high wavenumbers compared to traditional numerical solvers, even those using a $1024 \times 1024$ grid resolution. This demonstrates PINNs' potential advantage in resolving fine-scale dynamics without requiring explicit grid discretization – a key capacity for turbulence modeling.

**Ablation studies.** We conduct systematic experiments to evaluate SOAP's performance across different architectures and hyperparameter settings, establishing the robustness of our approach. Our first investigation examines SOAP's effectiveness across three representative architectures: standard MLP, modified MLP [21], and PirateNet [27]. As shown in the top panel of Figure 4, testing each architecture on four benchmark PDEs (Wave, Burgers, Allen-Cahn, and KdV equations), we find that SOAP consistently improves accuracy compared to Adam regardless of the underlying network backbones. In particular, PirateNet seems to be the most effective architecture across all test cases, leading to its selection for our main experiments.

As illustrated in the bottom panel of Figure 4, our results of SOAP's hyperparameters reveal two critical factors affecting performance. The preconditioner update frequency presents a clear trade-off between accuracy and computational cost. While more frequent updates yield better results, the improvements diminish beyond an update frequency of 2, which we selected as the optimal balance for our experiments. The momentum parameter $\beta_1$ proved especially crucial: high momentum ($\beta_1 = 0.99$) consistently achieves the best results, while low momentum ($\beta_1$=0.01) significantly degrades performance across all test cases.

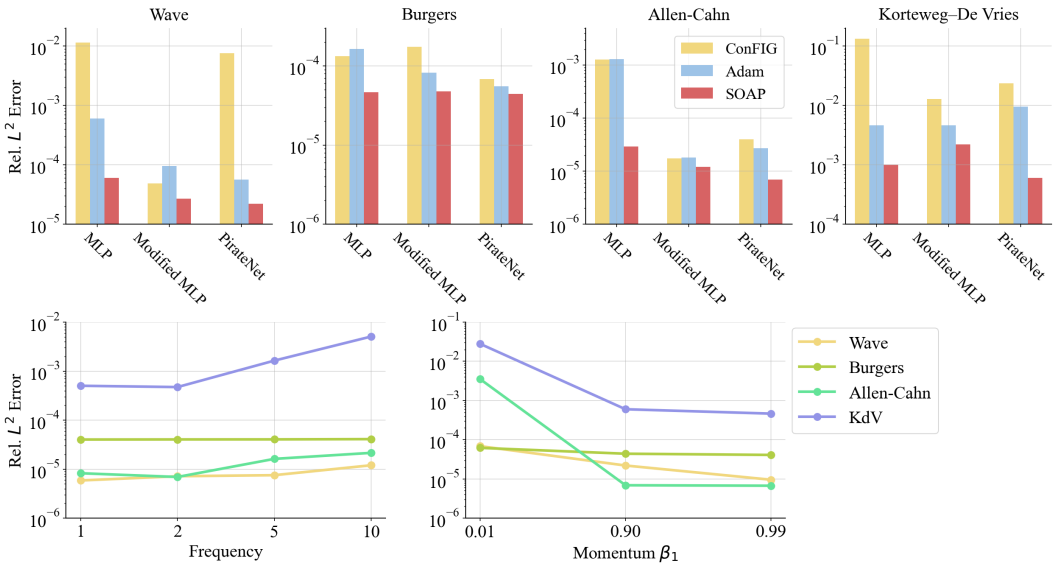


Figure 4: Optimizer performance comparison and ablation studies. Top: Relative $L^2$ error across PDE benchmarks using different optimizers. Bottom left: Relative $L^2$ error for varying preconditioner update frequencies in SOAP optimizer. Bottom right: Relative $L^2$ error with different momentum values in SOAP optimizer.

For completeness, we also compared SOAP against ConFiG [58], a recently proposed method for addressing gradient conflicts in PINNs that has demonstrated relative good performance compared to established baselines such as PCGrad [75] and IMTL-G [76] in multi-task learning. Despite its promising theoretical foundations, ConFiG showed some sensitivity to hyperparameter settings in our experiments, resulting in inconsistent performance. These results highlight the practical advantages of SOAP's more robust optimization approach.

**Computational costs.** While SOAP requires approximately 2x longer training time compared to baselines (Table 7), our focus is exploring the performance frontier of PINNs through extended training to full convergence. Impressively, error and loss convergence curves (Appendix G.6) indicate that SOAP typically achieves rapid initial convergence, reaching a reasonable accuracy (approximately $10^{-4}$) within the first 10,000 iterations, followed by gradual error reduction in subsequent iterations. This suggests the potential for reducing training time by up to 10x while maintaining competitive performance. These findings motivate future research into designing computationally efficient optimization algorithms and training strategies for PINNs, paving the way for practical and scalable applications in complex physics simulations.

## 7 Conclusion

This work advances our understanding of gradient conflicts in training of PINNs. We proposed gradient alignment scores as a quantitative measure of such conflicts, and provide both theoretical analysis and empirical evidence demonstrating their prevalence during PINN optimization. Furthermore, we show that second-order optimization methods implicitly promote gradient alignment, offering a principled approach to mitigating these conflicts. Particularly, we uncover a connection between SOAP and Newton's method. Extensive experiments across 10 PDE benchmarks confirm the effectiveness of quasi second-order optimizers, with SOAP achieving state-of-the-art performance.

Building on these insights, several promising research directions emerge. While SOAP demonstrates the power of gradient alignment in handling coupled physical constraints, opportunities exist for more efficient preconditioned algorithms that maintain their effectiveness with reduced computational cost. More broadly, our work suggests that the principles of gradient alignment and second-order preconditioning could benefit many deep learning applications involving competing objectives, though challenges remain in scaling these approaches to larger systems. Success in these directions could transform both scientific computing and multi-task optimization.

## Acknowledgment

B.L. would like to acknowledge support from National Key R&D Program of China Grant No. 2024YFA1016000. P.P. and S.W. acknowledge support from the US Department of Energy under the Advanced Scientific Computing Research program (grant DE-SC0024563), the Nvidia Academic Grant Program, and the Institute for Foundations of Data Science at Yale University. We also thank the developers of the software that enabled our research, including JAX [77], Matplotlib [78], and NumPy [79].

## References

[1] Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.

[2] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[3] Muhammad M Almajid and Moataz O Abu-Al-Saud. Prediction of porous media fluid flow using physics informed neural networks. *Journal of Petroleum Science and Engineering*, 208:109205, 2022.

[4] Hamidreza Eivazi, Mojtaba Tahani, Philipp Schlatter, and Ricardo Vinuesa. Physics-informed neural networks for solving reynolds-averaged navier–stokes equations. *Physics of Fluids*, 34(7), 2022.

[5] Zhen Cao, Kai Liu, Kun Luo, Sifan Wang, Liang Jiang, and Jianren Fan. Surrogate modeling of multi-dimensional premixed and non-premixed combustion using pseudo-time stepping physics-informed neural networks. *Physics of Fluids*, 36(11), 2024.

[6] Jiaxuan Xu, Han Wei, and Hua Bao. Physics-informed neural networks for studying heat transfer in porous media. *International Journal of Heat and Mass Transfer*, 217:124671, 2023.

[7] Hassan Bararnia and Mehdi Esmaeilpour. On the application of physics informed neural networks (pinn) to solve boundary layer thermal-fluid problems. *International Communications in Heat and Mass Transfer*, 132:105890, 2022.

[8] Gargya Gokhale, Bert Claessens, and Chris Develder. Physics informed neural networks for control oriented thermal modeling of buildings. *Applied Energy*, 314:118852, 2022.

[9] Georgios Kissas, Yibo Yang, Eileen Hwuang, Walter R Witschey, John A Detre, and Paris Perdikaris. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 358:112623, 2020.

[10] Xuelan Zhang, Baoyan Mao, Yue Che, Jiaheng Kang, Mingyao Luo, Aike Qiao, Youjun Liu, Hitomi Anzai, Makoto Ohta, Yuting Guo, et al. Physics-informed neural networks (pinns) for 4d hemodynamics prediction: an investigation of optimal framework based on vascular morphology. *Computers in Biology and Medicine*, 164:107287, 2023.

[11] Federica Caforio, Francesco Regazzoni, Stefano Pagani, Elias Karabelas, Christoph Augustin, Gundolf Haase, Gernot Plank, and Alfio Quarteroni. Physics-informed neural network estimation of material properties in soft tissue nonlinear biomechanical models. *Computational Mechanics*, pages 1–27, 2024.

[12] Enrui Zhang, Ming Dao, George Em Karniadakis, and Subra Suresh. Analyses of internal structures and defects in materials using physics-informed neural networks. *Science advances*, 8(7):eabk0644, 2022.

[13] Hyogu Jeong, Jinshuai Bai, Chanaka Prabuddha Batuwatta-Gamage, Charith Rathnayaka, Ying Zhou, and YuanTong Gu. A physics-informed neural network-based topology optimization (pinnto) framework for structural optimization. *Engineering Structures*, 278:115484, 2023.

[14] Haoteng Hu, Lehua Qi, and Xujiang Chao. Physics-informed neural networks (pinn) for computational solid mechanics: Numerical frameworks and applications. *Thin-Walled Structures*, page 112495, 2024.

[15] Alexander Kovacs, Lukas Exl, Alexander Kornell, Johann Fischbacher, Markus Hovorka, Markus Gusenbauer, Leoni Breth, Harald Oezelt, Masao Yano, Noritsugu Sakuma, et al. Conditional physics informed neural networks. *Communications in Nonlinear Science and Numerical Simulation*, 104:106041, 2022.

[16] Arbaaz Khan and David A Lowther. Physics informed neural networks for electromagnetic analysis. *IEEE Transactions on Magnetics*, 58(9):1–4, 2022.

[17] Marco Baldan, Paolo Di Barba, and David A Lowther. Physics-informed neural networks for inverse electromagnetic problems. *IEEE Transactions on Magnetics*, 59(5):1–5, 2023.

[18] Jonthan D Smith, Zachary E Ross, Kamyar Azizzadenesheli, and Jack B Muir. Hyposvi: Hypocentre inversion with stein variational inference and physics informed neural networks. *Geophysical Journal International*, 228(1):698–710, 2022.

[19] Chao Song and Yanghua Wang. Simulating seismic multifrequency wavefields with the fourier feature physics-informed neural network. *Geophysical Journal International*, 232(3):1503–1514, 2023.

[20] Pu Ren, Chengping Rao, Su Chen, Jian-Xun Wang, Hao Sun, and Yang Liu. Seismicnet: Physics-informed neural networks for seismic wave modeling in semi-infinite domain. *Computer Physics Communications*, 295:109010, 2024.

[21] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[22] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[23] Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative filter networks. In *International Conference on Learning Representations*, 2021.

[24] Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Finite basis physics-informed neural networks (fbpinns): a scalable domain decomposition approach for solving differential equations. *arXiv preprint arXiv:2107.07871*, 2021.

[25] Namgyu Kang, Byeonghyeon Lee, Youngjoon Hong, Seok-Bae Yun, and Eunbyung Park. Pixel: Physics-informed cell representations for fast and accurate pde solvers. *arXiv preprint arXiv:2207.12800*, 2022.

[26] Junwoo Cho, Seungtae Nam, Hyunmo Yang, Seok-Bae Yun, Youngjoon Hong, and Eunbyung Park. Separable physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[28] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.

[29] Jassem Abbasi and Pål Østebø Andersen. Physical activation functions (pafs): An approach for more efficient induction of physics into physics-informed neural networks (pinns). *Neurocomputing*, 608:128352, 2024.

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

[31] Francisco Sahli Costabal, Simone Pezzuto, and Paris Perdikaris. $\delta$-pinns: physics-informed neural networks on complex geometries. *Engineering Applications of Artificial Intelligence*, 127:107324, 2024.

[32] Chengxi Zeng, Tilo Burghardt, and Alberto M Gambaruto. Rbf-pinn: Non-fourier positional embedding in physics-informed neural networks. *arXiv preprint arXiv:2402.08367*, 2024.

[33] Xinquan Huang and Tariq Alkhalifah. Efficient physics-informed neural networks using hash encoding. *Journal of Computational Physics*, page 112760, 2024.

[34] 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*, 2021.

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

[36] Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023.

[37] Johannes Müller and Marius Zeinhofer. Achieving high accuracy with pinns via energy natural gradient descent. In *International Conference on Machine Learning*, pages 25471–25485. PMLR, 2023.

[38] Anas Jnini, Flavio Vella, and Marius Zeinhofer. Gauss-newton natural gradient descent for physics-informed computational fluid dynamics. *arXiv preprint arXiv:2402.10680*, 2024.

[39] Yongcun Song, Xiaoming Yuan, and Hangrui Yue. The admm-pinns algorithmic framework for nonsmooth pde-constrained optimization: a deep learning approach. *SIAM Journal on Scientific Computing*, 46(6):C659–C687, 2024.

[40] Jorge F Urbán, Petros Stefanou, and José A Pons. Unveiling the optimization process of physics informed neural networks: How accurate and competitive can pinns be? *Journal of Computational Physics*, 523:113656, 2025.

[41] Colby L Wight and Jia Zhao. Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542*, 2020.

[42] Aditi S Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *arXiv preprint arXiv:2109.01050*, 2021.

[43] Wenbo Cao and Weiwei Zhang. Tsonn: Time-stepping-oriented neural network for solving partial differential equations. *arXiv preprint arXiv:2310.16491*, 2023.

[44] Shaan Desai, Marios Mattheakis, Hayden Joy, Pavlos Protopapas, and Stephen Roberts. One-shot transfer learning of physics-informed neural networks. *arXiv preprint arXiv:2110.11286*, 2021.

[45] Somdatta Goswami, Cosmin Anitescu, Souvik Chakraborty, and Timon Rabczuk. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics*, 106:102447, 2020.

[46] Souvik Chakraborty. Transfer learning based multi-fidelity physics informed deep neural network. *Journal of Computational Physics*, 426:109942, 2021.

[47] Pao-Hsiung Chiu, Jian Cheng Wong, Chinchun Ooi, My Ha Dao, and Yew-Soon Ong. Can-pinn: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395:114909, 2022.

[48] Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. hp-vpinns: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021.

[49] Ravi G Patel, Indu Manickam, Nathaniel A Trask, Mitchell A Wood, Myoungkyu Lee, Ignacio Tomas, and Eric C Cyr. Thermodynamically consistent physics-informed neural networks for hyperbolic systems. *Journal of Computational Physics*, 449:110754, 2022.

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

[51] Hwijae Son, Jin Woo Jang, Woo Jin Han, and Hyung Ju Hwang. Sobolev training for physics informed neural networks. *arXiv preprint arXiv:2101.08932*, 2021.

[52] 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.

[53] Wensheng Li, Chao Zhang, Chuncheng Wang, Hanting Guan, and Dacheng Tao. Revisiting pinns: Generative adversarial physics-informed neural networks and point-weighting method. *arXiv preprint arXiv:2205.08754*, 2022.

[54] Wenqian Chen, Amanda A Howard, and Panos Stinis. Self-adaptive weights based on balanced residual decay rate for physics-informed neural networks and deep operator networks. *arXiv preprint arXiv:2407.01613*, 2024.

[55] Sokratis J Anagnostopoulos, Juan Diego Toscano, Nikolaos Stergiopulos, and George Em Karniadakis. Residual-based attention in physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116805, 2024.

[56] Li Liu, Shengping Liu, Hui Xie, Fansheng Xiong, Tengchao Yu, Mengjuan Xiao, Lufeng Liu, and Heng Yong. Discontinuity computing using physics-informed neural networks. *Journal of Scientific Computing*, 98(1):22, 2024.

[57] Jiahao Song, Wenbo Cao, Fei Liao, and Weiwei Zhang. Vw-pinns: A volume weighting method for pde residuals in physics-informed neural networks. *Acta Mechanica Sinica*, 41(3):324140, 2025.

[58] Qiang Liu, Mengyu Chu, and Nils Thuerey. Config: Towards conflict-free training of physics informed neural networks. *arXiv preprint arXiv:2408.11104*, 2024.

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

[60] 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.

[61] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[62] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

[63] Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE transactions on neural networks and learning systems*, 29(5):1454–1466, 2017.

[64] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[65] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[66] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018.

[67] Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training pinns: A loss landscape perspective. *arXiv preprint arXiv:2402.01868*, 2024.

[68] U Ghia, K.N Ghia, and C.T Shin. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics*, 48(3):387–411, 1982.

[69] Sifan Wang, Hanwen Wang, Jacob H Seidman, and Paris Perdikaris. Random weight factorization improves the training of continuous neural representations. *arXiv preprint arXiv:2210.01274*, 2022.

[70] Suchuan Dong and Naxian Ni. A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. *Journal of Computational Physics*, 435:110242, 2021.

[71] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

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

[74] Zhongkai Hao, Jiachen Yao, Chang Su, Hang Su, Ziao Wang, Fanzhi Lu, Zeyu Xia, Yichi Zhang, Songming Liu, Lu Lu, et al. Pinnacle: A comprehensive benchmark of physics-informed neural networks for solving pdes. *arXiv preprint arXiv:2306.08827*, 2023.

[75] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

[76] Liyang Liu, Yi Li, Zhanghui Kuang, J Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. iclr, 2021.

[77] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[78] John D Hunter. Matplotlib: A 2D graphics environment. *IEEE Annals of the History of Computing*, 9(03):90–95, 2007.

[79] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[80] Hanxu Zhou, Zhou Qixuan, Tao Luo, Yaoyu Zhang, and Zhi-Qin Xu. Towards understanding the condensation of neural networks at initial training. *Advances in Neural Information Processing Systems*, 35:2184–2196, 2022.

[81] Zheng-an Chen and Tao Luo. On the dynamics of three-layer neural networks: initial condensation. *arXiv preprint arXiv:2402.15958*, 2024.

[82] Anas Barakat and Pascal Bianchi. Convergence and dynamical behavior of the adam algorithm for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 31(1):244–274, 2021.

[83] Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning*, pages 404–413. PMLR, 2018.

[84] Adepu Ravi Sankar, Yash Khasbage, Rahul Vigneswaran, and Vineeth N Balasubramanian. A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9481–9488, 2021.

[85] Igor Molybog, Peter Albert, Moya Chen, Zachary DeVito, David Esiobu, Naman Goyal, Punit Singh Koura, Sharan Narang, Andrew Poulton, Ruan Silva, et al. A theory on adam instability in large-scale machine learning. *arXiv preprint arXiv:2304.09871*, 2023.

[86] Depen Morwani, Itai Shapira, Nikhil Vyas, Eran Malach, Sham Kakade, and Lucas Janson. A new perspective on shampoo's preconditioner. *arXiv preprint arXiv:2406.17748*, 2024.

[87] Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*, 2020.

[88] Sokratis J Anagnostopoulos, Juan Diego Toscano, Nikolaos Stergiopulos, and George Em Karniadakis. Residual-based attention and connection to information bottleneck theory in pinns. *arXiv preprint arXiv:2307.00379*, 2023.

[89] Xenofon Karakonstantis, Diego Caviedes-Nozal, Antoine Richard, and Efren Fernandez-Grande. Room impulse response reconstruction with physics-informed deep learning. *The Journal of the Acoustical Society of America*, 155(2):1048–1059, 2024.

[90] Jiaming Zhang, David Dalton, Hao Gao, and Dirk Husmeier. Physics-informed deep learning based on the finite difference method for efficient and accurate numerical solution of partial differential equations.

[91] 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. *arXiv preprint arXiv:2006.10739*, 2020.

[92] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[93] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

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

[95] Syver Døving Agdestein, Simone Ciarella, and Benjamin Sanderse. IncompressibleNavierStokes.jl, November 2024.

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

[97] Revanth Mattey and Susanta Ghosh. A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390:114474, 2022.

[98] Vasiliy A Es' kin, Danil V Davydov, Ekaterina D Egorova, Alexey O Malkhanov, Mikhail A Akhukov, and Mikhail E Smorkalov. About optimal loss function for training physics-informed neural networks under respecting causality. *arXiv preprint arXiv:2304.02282*, 2023.

[99] Norman J Zabusky and Martin D Kruskal. Interaction of" solitons" in a collisionless plasma and the recurrence of initial states. *Physical review letters*, 15(6):240, 1965.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions, including: introducing a gradient alignment score to quantify directional conflicts (Sec. 1-2), demonstrating how second-order optimizers enhance gradient alignment (Sec. 3), establishing SOAP's connection to Newton's method (Sec. 4), and showing improved performance across 10 PDE benchmarks (Sec. 5).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses computational costs in Section 5 (paragraph "Computational costs"), noting that SOAP requires approximately 2x longer training time compared to baselines, though it achieves better convergence. The conclusion also acknowledges the need for "more efficient preconditioned algorithms that maintain their effectiveness with reduced computational cost.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper includes theoretical results with clear assumptions and proofs. Proposition 1 in Section 3 establishes the connection between alignment score and cosine similarity. Proposition 2 analyzes gradient alignment at initialization. Proposition 3 connects preconditioned gradient descent to Newton's method. Complete proofs are provided in the appendix (referenced throughout Section 3-4).

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides sufficient information to reproduce the main experimental results. Section 5 describes the baseline setup, and the appendices (referenced in Section 5) contain complete information on experimental settings, hyperparameters, and data generation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [NA]

   Justification: We will release the code once the paper is accepted.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies training and test details in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The paper includes multiple performance comparisons across different PDEs and optimizers, with ablation studies in Figure 4 showing the impact of different hyperparameter settings. The consistent performance improvements across diverse benchmarks (Table 2) and multiple experimental configurations support the statistical significance of the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides information about computational resources in Appendix G.5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. It focuses on fundamental advances in scientific machine learning with no apparent ethical concerns. The work acknowledges all related research properly and presents results transparently.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The conclusion discusses broader positive impacts, noting that the principles of gradient alignment could benefit applications involving competing objectives beyond scientific computing. The research enables more accurate simulation of complex physical systems, which has positive societal impact for scientific and engineering applications. As with any tool that furthers our understanding and ability to predict the outcomes of complex systems, there may be ill-intentioned use cases, but we do not expect any specific negative impact from this work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper focuses on optimization techniques for solving PDEs and does not involve models or data with high risk for misuse. The methods presented are used for scientific computing applications without foreseeable harmful applications.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper properly cites existing work and acknowledges software used (JAX, Matplotlib, Chebfun, and NumPy) in the Acknowledgments section. The baseline implementation builds on prior work that is appropriately cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release code implementing the methods described in this paper upon acceptance.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects. It focuses on algorithmic improvements and numerical experiments for solving PDEs.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The research does not involve human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The paper does not indicate the use of LLMs as part of the core methods or research.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

# A Nomenclature

Table 4: Notation used throughout the paper. We use uppercase letters for matrices and lowercase letters for their vectorized forms. All gradients and Hessians are with respect to the loss function $\mathcal{L}$.

| Symbol | Description |
|---|---|
| $\mathcal{L}$ | Loss function |
| $\mathcal{L}_{ics}$ | Initial condition losses |
| $\mathcal{L}_{res}$ | PDE residual losses |
| $\theta$ | Neural network parameters |
| $W$ | Weight matrix for a given layer |
| $w$ | Vectorized weight matrix, $w = \text{Vec}(W)$ |
| $G$ | Gradient matrix, $G = \nabla_W \mathcal{L}$ |
| $g$ | Vectorized gradient, $g = \text{Vec}(G)$ |
| $F$ | Fisher information matrix |
| $H$ | Hessian matrix, $H = \nabla^2_\theta \mathcal{L}$ |
| $H_{\text{Ada}}$ | Full Adagrad preconditioner matrix |
| $H_{\text{GN}}$ | Gauss-Newton approximation of the Hessian |
| $\mathcal{A}$ | Gradient alignment score between loss components |

# B Analysis of Intra-step Gradient Alignment

We present some preliminary analysis to understand intra-step gradient conflicts in training PINNs via standard gradient descent, Adam [71], and Shampoo algorithms [66], and how SOAP can effectively resolve them during training. For simplicity, we consider the simplest case of using PINNs with the two-layer NN to solve the one-dimensional Laplace equation and focus on the analysis of the intra-step gradient alignment (3.3) with small initialization. The analysis can be easily extended to other types of PDEs. Following the general setup in Section 2, without loss of generality, we consider 1D Laplace equation as follows

$$\begin{cases} \Delta u = u'' = 0 \quad \text{on } [-1, 1], \\ u(\pm 1) = g_{\pm 1}. \end{cases} \tag{B.1}$$

We approximate the solution $u(x)$ by a two-layer network with width $N$:

$$u(x, \theta) = \sum_{i=1}^{N} a_i \sigma(w_i x) = \mathbf{a} \cdot \sigma(\mathbf{w} x)\,, \tag{B.2}$$

where $\mathbf{a} = (a_1, \ldots, a_N), \mathbf{w} = (w_1, \ldots, w_N) \in \mathbb{R}^N$, and $\theta = (\mathbf{a}, \mathbf{w}) \in \mathbb{R}^{2N}$. Moreover, we limit ourselves to the activation function $\sigma(x) = \tanh(x)$. In this case, the loss (2.5) reduces to

$$\min_{\theta=(\mathbf{a},\mathbf{w})} \mathcal{L}(\theta) = \underbrace{\frac{1}{N_r} \sum_{p=1}^{N_r} |u''(x_p, \theta)|^2}_{\mathcal{L}_r(\theta)} + \underbrace{\frac{1}{2} \sum_{s=\pm 1} |u(s, \theta) - g_s|^2}_{\mathcal{L}_{bc}(\theta)}\,. \tag{B.3}$$

To analyze the gradient conflict phenomenon in training PINNs, we consider the *small initialization* regime.

**Assumption 1.** *The weights $a_i, w_i$ are initialized by i.i.d. Gaussian $\mathcal{N}(0, \varepsilon^2)$ with small $\varepsilon = o(1)$.*

This allows us to introduce the normalized parameters:

$$\bar{\mathbf{a}} = \varepsilon^{-1} \mathbf{a}\,, \quad \bar{\mathbf{w}} = \varepsilon^{-1} \mathbf{w}\,,$$

initialized as standard Gaussian.

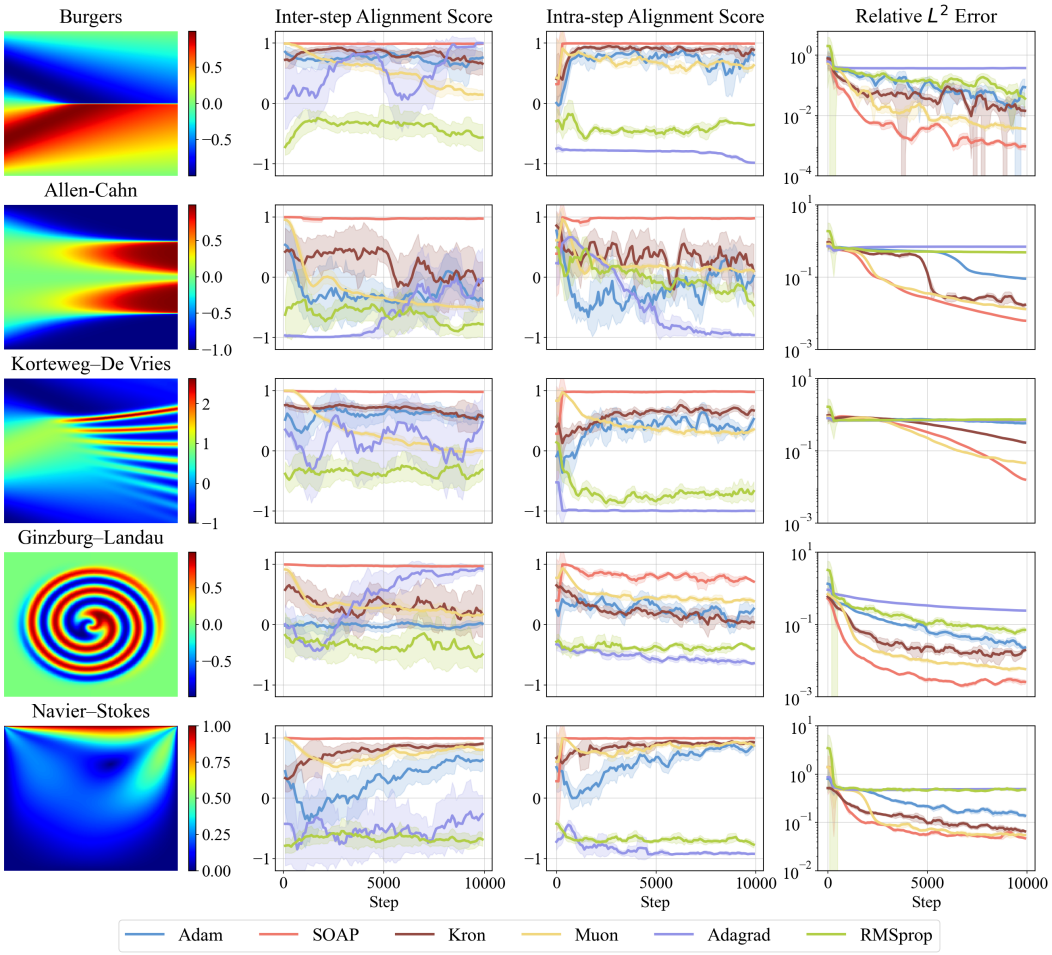


Figure 5: Gradient alignment scores and test errors obtained by training PINNs with different optimizers across different PDEs. From left to right: ground truth PDE solution, intra-step gradient alignment scores (Eq. (3.3)), inter-step gradient alignment scores (Eq. (3.4)), and test error convergence during training.

**Lemma 1.** *Under small initialization, the gradients of the residual and boundary loss terms can be approximated as:*

$$\nabla_\theta \mathcal{L}_r(\theta) = \varepsilon^7 G^r(\bar{\mathbf{a}}, \bar{\mathbf{w}}) + O(\varepsilon^9), \tag{B.4}$$

$$\nabla\theta \mathcal{L}_{bc}(\theta) = \varepsilon G^{bc}(\bar{\mathbf{a}}, \bar{\mathbf{w}}) + O(\varepsilon^3), \tag{B.5}$$

*where*

$$G^r(\bar{\mathbf{a}}, \bar{\mathbf{w}}) = c_r \bar{\mathbf{a}} \cdot \bar{\mathbf{w}}^{\odot 3} \left(\bar{\mathbf{w}}^{\odot 3}, 3\bar{\mathbf{a}} \odot \bar{\mathbf{w}}^{\odot 2}\right), \tag{B.6}$$

$$G^{bc}(\bar{\mathbf{a}}, \bar{\mathbf{w}}) = (g_{-1} - g_1)(\bar{\mathbf{w}}, \bar{\mathbf{a}}). \tag{B.7}$$

*Here $\bar{\mathbf{a}} = \varepsilon^{-1}\mathbf{a}$, $\bar{\mathbf{w}} = \varepsilon^{-1}\mathbf{w}$ are the normalized parameters, and $G^r$, $G^{bc}$ are the effective gradient terms.*

We remark that these elementary computations also provide insights into the gradient magnitude imbalance discussed in Section 3, noting $\|\nabla_\theta \mathcal{L}_r(\theta)\| = O(\varepsilon^7)$ while $\|\nabla_\theta \mathcal{L}_{bc}(\theta)\| = O(\varepsilon)$.

*Proof.* We recall the Taylor expansions of the activation function $\sigma(x) = \tanh(x)$ and its derivatives for later use:

$$\begin{aligned} \sigma(x) &= x - \frac{x^3}{3} + O(x^5), \quad \sigma'(x) = 1 - x^2 + O(x^4), \\ \sigma''(x) &= -2x + \frac{8x^3}{3} + O(x^5), \quad \sigma'''(x) = -2 + 8x^2 + O(x^4). \end{aligned} \tag{B.8}$$

The gradient of loss function $\mathcal{L}(\theta)$ consists of two parts computed as follows:

$$\nabla_{\theta=(\mathbf{a},\mathbf{w})}\mathcal{L}_r(\theta) = \frac{2}{N_r}\sum_{p=1}^{N_r} u_{xx}(x_p,\theta)\nabla_{\theta=(\mathbf{a},\mathbf{w})}u_{xx}(x_p\,,\theta)\,,$$

and

$$\nabla_{\theta=(\mathbf{a},\mathbf{w})}\mathcal{L}_{bc}(\theta) = \sum_{s=\pm 1}(u(s,\theta)-g(s))\nabla_{\theta=(\mathbf{a},\mathbf{w})}u(s,\theta)\,,$$

with, thanks to (4.2) and (B.8),

$$\begin{aligned}\nabla_{\mathbf{a}}u(x,\theta) &= \sigma(\mathbf{w}\mathbf{x}) = \varepsilon\bar{\mathbf{w}}x + O(\varepsilon^3)\,,\\ \nabla_{w_i}u(x,\theta) &= a_i\sigma'(w_i x)x = \varepsilon\bar{a_i}x - \varepsilon^3\bar{a_i}\overline{w_i}^2x^3 + O(\varepsilon^4)\,,\end{aligned}$$

and

$$\begin{aligned}\nabla_{a_i}u_{xx}(x,\theta) &= \sigma''(w_i x)|w_i|^2 = -2\varepsilon^3\overline{w_i}^3x + O(\varepsilon^5)\,,\\ \nabla_{w_i}u_{xx}(\mathbf{x},\theta) &= a_i\sigma'''(w_i x)|w_i|^2x + 2a_i\sigma''(w_i x)w_i\\ &= -6\varepsilon^3\bar{a_i}\overline{w_i}^2x + O(\varepsilon^5)\,.\end{aligned}$$

We also compute

$$u(x,\theta) = \varepsilon^2\bar{\mathbf{a}}\cdot\bar{\mathbf{w}}x + O(\varepsilon^4)\,,$$

and

$$u_{xx}(x,\theta) = \sum_i a_i\sigma''(w_i x)|w_i|^2 = -2\varepsilon^4\sum_i \bar{a_i}\overline{w_i}^3x + O(\varepsilon^6)\,.$$

For convenience, we define the componentwise power $\mathbf{x}^{\odot k} = (x_1^k,\dots,x_N^k)$ and product $\mathbf{x}\odot\mathbf{y} = (x_1y_1,\dots,x_Ny_N)$ for $\mathbf{x},\mathbf{y}\in\mathbb{R}^N$. By the above computation, it follows that at the initialization, there hold

$$\begin{aligned}\nabla_{\theta=(\mathbf{a},\mathbf{w})}\mathcal{L}_r(\theta) &= \frac{2}{N_r}\sum_{p=1}^{N_r}\left(-2\varepsilon^4\bar{\mathbf{a}}\cdot\bar{\mathbf{w}}^{\odot 3}x_p\right)\left(-2\varepsilon^3\bar{\mathbf{w}}^{\odot 3}x_p, -6\varepsilon^3\bar{\mathbf{a}}\odot\bar{\mathbf{w}}^{\odot 2}x_p\right) + O(\varepsilon^9)\\ &= \varepsilon^7\frac{8}{N_r}\sum_{p=1}^{N_r}\bar{\mathbf{a}}\cdot\bar{\mathbf{w}}^{\odot 3}\left(\bar{\mathbf{w}}^{\odot 3}, 3\bar{\mathbf{a}}\odot\bar{\mathbf{w}}^{\odot 2}\right)x_p^2 + O(\varepsilon^9)\,,\end{aligned}$$

and

$$\begin{aligned}\nabla_{\theta=(\mathbf{a},\mathbf{w})}\mathcal{L}_{bc}(\theta) &= \sum_{s=\pm 1}\left(\varepsilon^2\bar{\mathbf{a}}\cdot\bar{\mathbf{w}}s + O(\varepsilon^4) - g(s)\right)\left(\varepsilon\bar{\mathbf{w}}s + O(\varepsilon^3), \varepsilon\bar{\mathbf{a}}s - \varepsilon^3\bar{\mathbf{a}}\odot\bar{\mathbf{w}}^{\odot 2}s^3 + O(\varepsilon^4)\right)\\ &= -\varepsilon\sum_{s=\pm 1} sg(s)\left(\bar{\mathbf{w}},\bar{\mathbf{a}}\right) + O(\varepsilon^3) = \varepsilon(g_{-1}-g_1)\left(\bar{\mathbf{w}},\bar{\mathbf{a}}\right) + O(\varepsilon^3)\,.\end{aligned}$$

We then define constant $c_r = 8N_r^{-1}\sum_{p=1}^{N_r}x_p^2 > 0$ and the effective gradients

$$G^r(\bar{\mathbf{a}},\bar{\mathbf{w}}) = (G_a^r(\bar{\mathbf{a}},\bar{\mathbf{w}}), G_w^r(\bar{\mathbf{a}},\bar{\mathbf{w}})) = c_r\bar{\mathbf{a}}\cdot\bar{\mathbf{w}}^{\odot 3}\left(\bar{\mathbf{w}}^{\odot 3}, 3\bar{\mathbf{a}}\odot\bar{\mathbf{w}}^{\odot 2}\right)\,, \tag{B.9}$$

and

$$G^{bc}(\bar{\mathbf{a}},\bar{\mathbf{w}}) = \left(G_a^{bc}(\bar{\mathbf{a}},\bar{\mathbf{w}}), G_w^{bc}(\bar{\mathbf{a}},\bar{\mathbf{w}})\right) = (g_{-1}-g_1)\left(\bar{\mathbf{w}},\bar{\mathbf{a}}\right)\,, \tag{B.10}$$

enabling us to write

$$\nabla_\theta\mathcal{L}_r(\theta) = \varepsilon^7 G^r(\bar{\mathbf{a}},\bar{\mathbf{w}}) + O(\varepsilon^9)\,,\quad \nabla_\theta\mathcal{L}_{bc}(\theta) = \varepsilon G^{bc}(\bar{\mathbf{a}},\bar{\mathbf{w}}) + O(\varepsilon^3)\,. \tag{B.11}$$

□

We are now ready to understand the gradient conflict for various optimizers applied to the residual and boundary loss terms separately.

**Proposition 3** *At initialization, the alignment score converges to a binary random variable in the infinite width limit:*

$$\lim_{N\to\infty} \mathcal{A}(\square(\nabla\mathcal{L}_b), \square(\nabla\mathcal{L}_r)) = O(\varepsilon^2) + C_\square \begin{cases} \operatorname{sgn}(g_{-1}-g_1) & \text{with prob. } \frac{1}{2}, \\ -\operatorname{sgn}(g_{-1}-g_1) & \text{with prob. } \frac{1}{2}. \end{cases} \tag{B.12}$$

*where* $\square = \mathrm{GD}, \mathrm{Adam}, \mathrm{Shampoo},$ *or* $\mathrm{Soap}$ *denotes the optimizer update rule, and* $C_\square$ *is a constant depending on the optimizer.*

We can see that these optimizers fail to resolve intra-step gradient conflicts in the initialization, aligning with the near-zero initial intra-step gradient scores shown in Figure 5.

*Proof.* **Gradient descent.** We start with the standard continuous-time gradient descent:

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = -\nabla_\theta \mathcal{L}(\theta)\,. \tag{B.13}$$

Motivated by [80, 81], under small initialization Assumption 1, in the *initial stage* of training dynamics where the leading-order expansion (B.11) holds for the weights $\mathbf{a}, \mathbf{w}$, the gradients $\nabla_\theta\mathcal{L}_r(\theta)$ and $\nabla_\theta\mathcal{L}_{bc}(\theta)$ can be effectively described by $G^r(\bar{\mathbf{a}}, \bar{\mathbf{w}})$ and $G^{bc}(\bar{\mathbf{a}}, \bar{\mathbf{w}})$, respectively, up to some scaling factors, then the gradient flow (B.13) can be approximated by the effective dynamics for the normalized parameter $\bar\theta = (\bar{\mathbf{a}}, \bar{\mathbf{w}})$:

$$\varepsilon\frac{\mathrm{d}\bar\theta}{\mathrm{d}t} = -\left(\varepsilon^7 G^r(\bar\theta) + \varepsilon G^{bc}(\bar\theta)\right)\,. \tag{B.14}$$

Recalling Definition 1, under our assumptions, we have the intra alignment score:

$$\mathcal{A}\left(\nabla_\theta\mathcal{L}_r(\theta), \nabla_\theta\mathcal{L}_{bc}(\theta)\right) = \mathcal{A}\left(G^r(\bar\theta), G^{bc}(\bar\theta)\right) + O(\varepsilon^2)\,,$$

where

$$\mathcal{A}\left(G^r(\bar\theta), G^{bc}(\bar\theta)\right) = \operatorname{sgn}(\bar{\mathbf{a}}\cdot\bar{\mathbf{w}}^{\odot 3})\operatorname{sgn}(g_{-1}-g_1)\frac{\sum_i \overline{w_i}^4 + 3\sum_i \bar{a_i}^2\overline{w_i}^2}{\sqrt{\sum_i \overline{w_i}^6 + 9\bar{a_i}^2\overline{w_i}^4}\sqrt{\sum_i \overline{w_i}^2 + \bar{a_i}^2}}\,.$$

Then we find that at the initialization, by $\bar{a_i}, \overline{w_i} \sim \mathcal{N}(0,1)$ from Assumption 1 and the law of large numbers,

$$\frac{\frac{1}{N}\sum_i \overline{w_i}^4 + 3\frac{1}{N}\sum_i \bar{a_i}^2\overline{w_i}^2}{\sqrt{\frac{1}{N}\sum_i \overline{w_i}^6 + 9\bar{a_i}^2\overline{w_i}^4}\sqrt{\frac{1}{N}\sum_i \overline{w_i}^2 + \bar{a_i}^2}} \longrightarrow \frac{6}{\sqrt{15+27\sqrt{2}}} = \frac{3}{\sqrt{21}}\,, \quad \text{almost surely.}$$

Also, by the symmetry of Gaussian, there holds $\mathbb{P}(\sum_i \bar{a_i}\overline{w_i}^3 > 0) = \frac{1}{2}$. It follows that the alignment score $\mathcal{A}(G^r(\bar\theta), G^{bc}(\bar\theta))_{t=0}$ converges to a binary random variable with expectation zero in the infinite width limit:

$$\mathcal{A}\left(G^r(\bar\theta), G^{bc}(\bar\theta)\right)_{t=0} \longrightarrow A = \begin{cases} \operatorname{sgn}(g_{-1}-g_1)\frac{3}{\sqrt{21}} & \text{with prob. } \frac{1}{2}\,, \\ -\operatorname{sgn}(g_{-1}-g_1)\frac{3}{\sqrt{21}} & \text{with prob. } \frac{1}{2}\,. \end{cases}$$

**Adam.** We now consider the deterministic version of the Adam optimizer [71], recalled below for completeness. Let $f(x)$ be a differentiable objective function on $\mathbb{R}^d$. The Adam iteration is defined by $z_n = T_{\gamma,\alpha,\beta}(n, z_{n-1})$ for $z_n = (x_n, m_n, v_n) \in \mathbb{R}^d\times\mathbb{R}^d\times\mathbb{R}^d$ with $z_0 = (x_0, 0, 0)$, where

$$T_{\gamma,\alpha,\beta}(n,z) = \begin{pmatrix} x - \frac{\gamma(1-\alpha^n)^{-1}(\alpha m + (1-\alpha)\nabla f(x))}{\epsilon + (1-\beta^n)^{-1/2}\sqrt{\beta v + (1-\beta)\nabla f(x)^{\odot 2}}} \\ \alpha m + (1-\alpha)\nabla f(x) \\ \beta v + (1-\beta)\nabla f(x)^{\odot 2} \end{pmatrix}\,.$$

We still consider the gradient conflict at the initialization, since the Adam dynamics is more complicated than the gradient flow one. From [82], we have that starting from $(x_0, 0, 0) \in \mathbb{R}^{3d}$, the Adam dynamics at $t = 0$ satisfies $\dot m(0) \propto \nabla f(x_0)$, $\dot v(0) \propto \nabla f(x_0)^{\odot 2}$, and

$$\dot x(0) = -\frac{\nabla f(x_0)}{\epsilon + \sqrt{\nabla f(x_0)^{\odot 2}}} \underset{\epsilon = o(1)}{\approx} -\frac{\nabla f(x_0)}{\sqrt{\nabla f(x_0)^{\odot 2}}}\,,$$

indicating that at early iterations of Adam, the algorithm performance would be similar to the *sign gradient descent* [83]. Back to our problem (4.3), by the above discussion, if we apply Adam to the loss functions $\mathcal{L}_r(\theta)$ and $\mathcal{L}_{bc}(\theta)$, respectively, at the initialization, the normalized weights $\bar{\theta} = \varepsilon^{-1}\theta$ will be updated along the directions:

$$\frac{\nabla \mathcal{L}_r(\theta)}{\sqrt{\nabla \mathcal{L}_r(\theta)^{\odot 2}}} = \frac{G^r(\bar{\theta})}{\sqrt{G^r(\bar{\theta})^{\odot 2}}} + O(\varepsilon^2)\,, \quad \frac{\nabla \mathcal{L}_{bc}(\theta)}{\sqrt{\nabla \mathcal{L}_{bc}(\theta)^{\odot 2}}} = \frac{G^{bc}(\bar{\theta})}{\sqrt{G^{bc}(\bar{\theta})^{\odot 2}}} + O(\varepsilon^2)\,.$$

Then, by (B.6) and (B.7), one can compute

$$G^r_{\mathrm{Adam}}(\bar{\theta}) = \frac{G^r(\bar{\theta})}{\sqrt{G^r(\bar{\theta})^{\odot 2}}} = \mathrm{sgn}(\bar{\mathbf{a}} \cdot \bar{\mathbf{w}}^{\odot 3})\,(\mathrm{sgn}(\bar{\mathbf{w}}), \mathrm{sgn}(\bar{\mathbf{a}}))\,,$$

and

$$G^{bc}_{\mathrm{Adam}}(\bar{\theta}) = \frac{G^{bc}(\bar{\theta})}{\sqrt{G^{bc}(\bar{\theta})^{\odot 2}}} = \mathrm{sgn}(g_{-1} - g_1)\,(\mathrm{sgn}(\bar{\mathbf{w}}), \mathrm{sgn}(\bar{\mathbf{a}}))\,.$$

It follows that the alignment score is

$$\mathcal{A}\left(G^r_{\mathrm{Adam}}(\bar{\theta}), G^{bc}_{\mathrm{Adam}}(\bar{\theta})\right) = \mathrm{sgn}(\bar{\mathbf{a}} \cdot \bar{\mathbf{w}}^{\odot 3})\,\mathrm{sgn}(g_{-1} - g_1) = \begin{cases} \mathrm{sgn}(g_{-1} - g_1) & \text{with prob. } \frac{1}{2}\,, \\ -\mathrm{sgn}(g_{-1} - g_1) & \text{with prob. } \frac{1}{2}\,, \end{cases}$$

which holds for any two-layer NN with width $N$.

**Shampoo and SOAP.** We proceed to consider Shampoo [66], which is a second-order optimizer with Kronecker product preconditioners. For the reader's convenience, we recall the Shampoo iterations for training neural networks. Following the notation in Section D, let $W_t, G_t \in \mathbb{R}^{m\times n}$ be the weight matrix and gradient matrix for a layer at time step $t$, respectively. Shampoo generates left and right preconditioners:

$$L_t = L_{t-1} + G_t G_t^T\,, \quad R_t = R_{t-1} + G_t^T G_t\,,$$

and then updates the weight matrix by

$$W_{t+1} = W_t - \eta L_t^{-1/4} G_t R_t^{-1/4}\,,$$

with step size $\eta > 0$. If we disable the accumulation in the preconditioners and set $L_t = G_t G_t^T$ and $R_t = G_t^T G_t$, then the Shampoo optimizer is simplified to

$$W_{t+1} = W_t - \eta\,\mathrm{Shampoo}(G_t)\,, \quad \mathrm{Shampoo}(G_t) := (G_t G_t^T)^{-1/4} G_t (G_t^T G_t)^{-1/4}\,,$$

with $\mathrm{Shampoo}(G_t) = U_t V_t^T$, where $U_t$ and $V_t$ are from the reduced singular value decomposition of $G_t = U_t \Sigma_t V_t^T$. It is clear that if we apply Shampoo to $\mathcal{L}_r(\theta)$ or $\mathcal{L}_{bc}(\theta)$ with the two-layer NN (4.2), then under small initialization Assumption 1, at the initialization, the updates of the normalized weights $\bar{\theta} = (\bar{\mathbf{a}}, \bar{\mathbf{w}}) = \varepsilon^{-1}\theta$ would be

$$\bar{\mathbf{a}} \leftarrow \bar{\mathbf{a}} - \eta\,\mathrm{Shampoo}\left(G_a^{r\,or\,bc}(\bar{\mathbf{a}}, \bar{\mathbf{w}})\right), \quad \bar{\mathbf{w}} \leftarrow \bar{\mathbf{w}} - \eta\,\mathrm{Shampoo}\left(G_w^{r\,or\,bc}(\bar{\mathbf{a}}, \bar{\mathbf{w}})\right).$$

Here $G_a^{r\,or\,bc}, G_w^{r\,or\,bc} \in \mathbb{R}^N$ are given in (B.6) and (B.7). Moreover, note that for any vector $x \in \mathbb{R}^d$, $\mathrm{Shampoo}(x)$ is simply $x/\|x\|$. Therefore, we can compute the (effective) initial Shampoo gradient directions for $\mathcal{L}_r(\theta)$ and $\mathcal{L}_{bc}(\theta)$:

$$G^r_{\mathrm{Shampoo}}(\bar{\theta}) := \left(\mathrm{Shampoo}\left(G_a^r(\bar{\theta})\right), \mathrm{Shampoo}\left(G_a^r(\bar{\theta})\right)\right) = \mathrm{sgn}(\bar{\mathbf{a}} \cdot \bar{\mathbf{w}}^{\odot 3}) \left(\frac{\bar{\mathbf{w}}^{\odot 3}}{\|\bar{\mathbf{w}}^{\odot 3}\|}, \frac{\bar{\mathbf{a}} \odot \bar{\mathbf{w}}^{\odot 2}}{\|\bar{\mathbf{a}} \odot \bar{\mathbf{w}}^{\odot 2}\|}\right).$$

and

$$G^{bc}_{\mathrm{Shampoo}}(\bar{\theta}) := \left(\mathrm{Shampoo}\left(G_a^{bc}(\bar{\theta})\right), \mathrm{Shampoo}\left(G_a^{bc}(\bar{\theta})\right)\right) = \mathrm{sgn}(g_{-1} - g_1) \left(\frac{\bar{\mathbf{w}}}{\|\bar{\mathbf{w}}\|}, \frac{\bar{\mathbf{a}}}{\|\bar{\mathbf{a}}\|}\right).$$

It follows that the alignment score is, as $N \to \infty$,

$$\mathcal{A}\left(G^r_{\mathrm{Shampoo}}(\bar{\theta}), G^{bc}_{\mathrm{Shampoo}}(\bar{\theta})\right) \longrightarrow C_{\mathrm{Shampoo}} \begin{cases} \mathrm{sgn}(g_{-1} - g_1) & \text{with prob. } \frac{1}{2}\,, \\ -\mathrm{sgn}(g_{-1} - g_1) & \text{with prob. } \frac{1}{2}\,. \end{cases}$$

We finally consider SOAP. Following the notations in Appendix D, if $G_t$ is a vector, then $\widetilde{G}_t = [1, 0, \cdots, 0]^\top$ and $\mathrm{Adam}(\widetilde{G}_t) = \widetilde{G}_t$. We transform this gradient back and obtain $G_t/\|G_t\|$. It means that at initialization,

$$\mathrm{Shampoo}\left(G_a^{r\,or\,bc}(\bar{\mathbf{a}}, \bar{\mathbf{w}})\right) = \mathrm{SOAP}\left(G_a^{r\,or\,bc}(\bar{\mathbf{a}}, \bar{\mathbf{w}})\right).$$

Therefore, the initial gradient conflict of SOAP follows from the case of Shampoo.

□

## C Inter-step gradient alignment of preconditioned gradient descent

**Proposition 4** *Let $\mathcal{L}(\theta)$ be a twice differentiable loss function with Hessian $H(\theta)$, and let $P(\theta)$ be a positive definite preconditioner with uniformly bounded inverse $P^{-1}(\theta)$. Consider the preconditioned gradient descent update with exponent $0 \leq s \leq 1$ and learning rate $\eta > 0$:*

$$\theta_{t+1} = \theta_t - \eta P^{-s}(\theta_t)\nabla\mathcal{L}(\theta_t) \tag{C.1}$$

*For consecutive gradient vectors $g_t = \nabla\mathcal{L}(\theta_t)$ and $g_{t+1} = \nabla\mathcal{L}(\theta_{t+1})$, the alignment score $\mathcal{A}(g_t, g_{t+1}) = \frac{g_t^T g_{t+1}}{\|g_t\|\|g_{t+1}\|}$ satisfies:*

$$\mathcal{A}(g_t, g_{t+1}) = 1 - \frac{\eta^2}{2}\frac{\|HP^{-s}g_t\|^2}{\|g_t\|^2} + O(\eta^3) \tag{C.2}$$

*Furthermore, to ensure gradient alignment $\mathcal{A}(g_t, g_{t+1}) \geq 1 - \epsilon$ for some $\epsilon > 0$, the learning rate must satisfy:*

$$\eta \leq \sqrt{\frac{2\epsilon\|g_t\|^2}{\|HP^{-s}g_t\|^2}} \tag{C.3}$$

*This bound specializes to the following cases:*

1. *Vanilla Gradient Descent ($P = I$):* $\eta_{\max} = \sqrt{\frac{2\epsilon}{\lambda_{\max}^2(H)}}$
2. *Newton's Method ($P = H, s = 1$):* $\eta_{\max} = \sqrt{2\epsilon}$

*Proof.* Let $P(\theta)$ be a positive definite preconditioner with uniformly bounded inverse $P^{-1}(\theta)$. For learning rate $\eta > 0$ and exponent $0 < s \leq 1$, the parameter update is:

$$\theta_{t+1} = \theta_t - \eta P^{-s}\left(\theta_t\right) g_t \tag{C.4}$$

where $g_t = \nabla\mathcal{L}\left(\theta_t\right)$.

We are interested in the behavior of the alignment between successive gradients under this update. To study this, we expand $g_{t+1} = \nabla\mathcal{L}(\theta_{t+1})$ via a second-order Taylor series around $\theta_t$, using the Hessian $H(\theta_t) = \nabla^2\mathcal{L}(\theta_t)$:

$$g_{t+1} = g_t + H\left(\theta_t\right)\Delta\theta_t + \frac{1}{2}D^3\mathcal{L}\left(\theta_t\right)\left[\Delta\theta_t, \Delta\theta_t\right] + O\left(\left\|\Delta\theta_t\right\|^3\right) \tag{C.5}$$

Substituting $\Delta\theta_t$:

$$g_{t+1} = g_t - \eta HP^{-s}g_t + \frac{\eta^2}{2}D^3\mathcal{L}\left(\theta_t\right)\left[P^{-s}g_t, P^{-s}g_t\right] + O\left(\eta^3\left\|P^{-s}g_t\right\|^3\right). \tag{C.6}$$

Assuming $\eta$ is small, we retain only the first-order term:

$$g_{t+1} = g_t - \eta HP^{-s}g_t + O\left(\eta^2\left\|P^{-s}g_t\right\|^2\right) \tag{C.7}$$

We compute the inner product between the current and next gradient:

$$g_t^\top g_{t+1} = \left\|g_t\right\|^2 - \eta g_t^\top HP^{-s}g_t + O\left(\eta^2\left\|P^{-s}g_t\right\|^2\left\|g_t\right\|\right). \tag{C.8}$$

We also expand the norm of the new gradient:

$$\left\|g_{t+1}\right\|^2 = \left\|g_t\right\|^2 - 2\eta g_t^\top HP^{-s}g_t + \eta^2\left\|HP^{-s}g_t\right\|^2 + O\left(\eta^3\right). \tag{C.9}$$

Taking the square root and using the approximation $\sqrt{1-x} \approx 1 - \frac{x}{2}$ for small $x$, we get:

$$\left\|g_{t+1}\right\| = \left\|g_t\right\|\left(1 - \eta\frac{g_t^\top HP^{-s}g_t}{\left\|g_t\right\|^2} + \frac{\eta^2}{2}\frac{\left\|HP^{-s}g_t\right\|^2}{\left\|g_t\right\|^2}\right) + O\left(\eta^3\right). \tag{C.10}$$

The cosine similarity (alignment score) between $g_t$ and $g_{t+1}$ is defined as:

$$\mathcal{A}(g_t, g_{t+1}) = \frac{g_t^\top g_{t+1}}{\|g_t\| \, \|g_{t+1}\|} \tag{C.11}$$

Substituting the approximations above:

$$\mathcal{A}(g_t, g_{t+1}) = \frac{\|g_t\|^2 - \eta g_t^\top H P^{-s} g_t + O\left(\eta^2\right)}{\|g_t\|^2 \left(1 - \eta \frac{g_t^\top H P^{-s} g_t}{\|g_t\|^2} + \frac{\eta^2}{2} \frac{\|H P^{-s} g_t\|^2}{\|g_t\|^2} + O\left(\eta^3\right)\right)}. \tag{C.12}$$

Factor $\|g_t\|_2^2$ in denominator:

$$\mathcal{A}(g_t, g_{t+1}) = \frac{1 - \eta \frac{g_t^\top H P^{-s} g_t}{\|g_t\|^2} + O\left(\eta^2\right)}{1 - \eta \frac{g_t^\top H P^{-s} g_t}{\|g_t\|^2} + \frac{\eta^2}{2} \frac{\|H P P^{-s} g_t\|^2}{\|g_t\|^2} + O\left(\eta^3\right)}. \tag{C.13}$$

Using $(1 - a + b)^{-1} = 1 + a - b + O\left(a^2 + b^2\right)$ for small $a, b$ :

$$\mathcal{A}(g_t, g_{t+1}) = \left(1 - \eta \frac{g_t^\top H P^{-s} g_t}{\|g_t\|^2}\right) \left(1 + \eta \frac{g_t^\top H P^{-s} g_t}{\|g_t\|^2} - \frac{\eta^2}{2} \frac{\|H P^{-s} g_t\|^2}{\|g_t\|^2}\right) + O\left(\eta^3\right). \tag{C.14}$$

Multiplying and simplifying to $O\left(\eta^2\right)$ :

$$\mathcal{A}(g_t, g_{t+1}) = 1 - \frac{\eta^2}{2} \frac{\|H P^{-s} g_t\|^2}{\|g_t\|^2} + O\left(\eta^3\right) \tag{C.15}$$

Thus, the alignment between gradients remains close to 1 if the second-order term is small. To ensure the gradients stay well-aligned, i.e., $\mathcal{A}(g_t, g_{t+1}) \geq 1 - \epsilon$ for a given tolerance $\epsilon > 0$, we require:

$$\frac{\eta^2}{2} \frac{\|H P^{-s} g_t\|^2}{\|g_t\|^2} \leq \epsilon \tag{C.16}$$

Solving for the maximum permissible learning rate $\eta$ yields:

$$\eta \leq \sqrt{\frac{2\epsilon \|g_t\|^2}{\|H P^{-s} g_t\|^2}} \tag{C.17}$$

We now examine this upper bound in a few special cases of interest:

1. Vanilla GD $(P = I)$ :

$$\eta_{\max} = \sqrt{\frac{2\epsilon}{\lambda_{\max}^2(H)}} \tag{C.18}$$

where $\lambda_{\max}(H)$ is the largest eigenvalue of $H$.

2. Newton's Method ( $P = H, s = 1$ ):

$$H P^{-1} = I \Longrightarrow \sigma_{\max}\left(H P^{-1}\right) = 1 \Longrightarrow \eta_{\max} = \sqrt{2\epsilon} \tag{C.19}$$

□

# D Connection between SOAP and Newton's method

SOAP [1] enhances Shampoo's efficiency by performing optimization in a transformed space aligned with the preconditioner's principal directions. For each layer's weight matrix and gradient $G_t \in \mathbb{R}^{m \times n}$, SOAP maintains two covariance matrices using exponential moving averages:

$$L_t = \beta_2 L_{t-1} + (1 - \beta_2) G_t G_t^T, \tag{D.1}$$

$$R_t = \beta_2 R_{t-1} + (1 - \beta_2) G_t^T G_t. \tag{D.2}$$

These matrices are then eigendecomposed as $L_t = Q_L \Lambda_L Q_L^T$ and $R_t = Q_R \Lambda_R Q_R^T$, where $\Lambda_L$ and $\Lambda_R$ contain the eigenvalues that capture the principal curvature directions of the loss landscape.

At each iteration $t$, SOAP updates each layer's weight matrix $W_t$ using its corresponding gradient $G_t$ as follows:

1. Project the weight and gradient into the eigenspace:
$$\widetilde{W}_t = Q_L^T W_t Q_R, \widetilde{G}_t = Q_L^T G_t Q_R.$$
2. Apply the Adam update in the *rotated* space:
$$\widetilde{W}_{t+1} = \widetilde{W}_t - \eta \ \mathrm{Adam}(\widetilde{G}_t).$$
3. Transform back to the original parameter space:
$$W_{t+1} = Q_L \widetilde{W}_{t+1} Q_R^T.$$

To reduce computational overhead, the preconditioners $L_t$ and $R_t$ are updated with frequency $f$ in practice. We will analyze the impact of update frequency and momentum parameters through ablation studies in Section 6.

Before diving into the formal analysis, let us build some intuition for why SOAP is particularly effective at resolving gradient conflicts. The key insight comes from understanding how second-order information captures interactions between different loss terms. When gradients conflict, it typically indicates that improving one objective requires coordinated changes across multiple parameters – information that is encoded in the Hessian matrix's off-diagonal elements.

SOAP approximates this second-order information in two complementary ways: (i) Its block-diagonal structure naturally captures parameter interactions within each network layer; (ii) Its adaptive preconditioner accumulates information about gradient correlations across training steps. This allows SOAP to implicitly identify and exploit parameter update directions that simultaneously improve multiple objectives. Rather than simply following the average gradient, SOAP can utilize the local loss landscape geometry to find more direct paths to good solutions. The following sections make this intuition precise through formal analysis of SOAP's convergence properties and gradient alignment characteristics.

To establish SOAP's connection to Newton's method, we begin by examining how the Hessian matrix can be approximated in neural networks. We limit our analysis with networks trained with cross-entropy loss. Here the Gauss-Newton approximation takes the form:

$$H_{\text{GN}} = \mathbb{E}\left[\frac{\partial f}{\partial W}\frac{\partial^2 \mathcal{L}}{\partial f^2}\frac{\partial f^T}{\partial W}\right] = \mathbb{E}\left[gg^T\right], \tag{D.3}$$

where $\mathcal{L}$ denotes the loss function, $f$ represents network outputs, and $G = \frac{\partial \mathcal{L}}{\partial W}$ is the gradient matrix with vectorization $g = \mathrm{vec}(G)$. Empirical evidence from [84] supports a key simplifying assumption:

**Assumption 2.** *The Gauss-Newton component provides a good approximation to the true Hessian: $H_{GN} \approx H$.*

For our purpose, we begin by noting that there exists a one-to-one correspondence between the original parameter space and the rotated space that preserves the matrix-vector multiplication.

**Lemma 2.** *Let $Q_L \in \mathbb{R}^{m\times m}$ and $Q_R \in \mathbb{R}^{n\times n}$ be two orthogonal matrices. For any matrix $A \in \mathbb{R}^{mn\times mn}$ and vector $v \in \mathbb{R}^{mn}$, define $\widetilde{v} := (Q_L \otimes Q_R)v$ and $\widetilde{A} := (Q_L \otimes Q_R)A(Q_L^T \otimes Q_R^T)$. Then there holds*
$$\widetilde{Av} = (Q_L \otimes Q_R)Av = \widetilde{A}\widetilde{v}\,.$$

The proof follows directly by applying the transformation $Q_L \otimes Q_R$ to $Av$ and the definitions of $\widetilde{A}$ and $\widetilde{v}$. Building on the above lemma, one can easily transform the preconditioned gradient descent in the original space to the rotated one and vice versa.

we can now establish the equivalence between the preconditioned gradient descent in the original and rotated spaces.

**Corollary 1.** *Let $W_t, G_t \in \mathbb{R}^{m\times n}$ be the weight matrix and gradient matrix for a layer at iteration $t$, respectively, with vectorizations $w_t = \mathrm{vec}(W_t)$ and $g_t = \mathrm{vec}(G_t)$. The preconditioned gradient descent update:*
$$w_{t+1} = w_t - \eta P^{-1} g_t\,, \tag{D.4}$$
*is equivalent to performing preconditioning in the rotated space:*
$$\widetilde{w}_{t+1} = \widetilde{w}_t - \eta \widetilde{P}^{-1}\widetilde{g}_t\,, \tag{D.5}$$
*where $P \in \mathbb{R}^{mn\times mn}$ is the preconditioner, and $\widetilde{w}$, $\widetilde{g}$, and $\widetilde{P}$ are the rotated weight, gradient, and preconditioner defined by the transformations in Lemma 2.*

**Proposition 5.** *Let $L_t = \mathbb{E}\left[G_t G_t^T\right]$ and $R_t = \mathbb{E}\left[G_t^T G_t\right]$ have eigendecompositions $L_t = Q_L \Lambda_L Q_L^T$ and $R_t = Q_R \Lambda_R Q_R^T$. Under the assumption of Lemma 4, the equivalent preconditioner in the rotated space is diagonal, i.e., $\widetilde{H}_{GN} = \operatorname{diag}(\widetilde{H}_{GN})$.*

*Proof.* The proof follows from the combination of Lemma 2 and Lemma 4. First, we express $H_{\text{GN}}$ using the Kronecker approximation:

$$H_{\text{GN}} = L_t^{1/2} \otimes R_t^{1/2} / \operatorname{Tr}\left(\mathbb{E}\left[GG^T\right]\right). \tag{D.6}$$

Then, we derive the rotated preconditioner:

$$\begin{aligned}
\widetilde{H}_{\text{GN}} &= (Q_L \otimes Q_R)\, H_{\text{GN}} \left(Q_L^T \otimes Q_R^T\right) \\
&= (Q_L \otimes Q_R)\left(L_t^{1/2} \otimes R_t^{1/2}\right)\left(Q_L^T \otimes Q_R^T\right) / \operatorname{Tr}\left(\mathbb{E}\left[GG^T\right]\right) \\
&= \left(Q_L L_t^{1/2} Q_L^T\right) \otimes \left(Q_R R_t^{1/2} Q_R^T\right) / \operatorname{Tr}\left(\mathbb{E}\left[\Lambda_L\right]\right) \\
&= \Lambda_L^{1/2} \otimes \Lambda_R^{1/2} / \operatorname{Tr}\left(\mathbb{E}\left[\Lambda_L\right]\right).
\end{aligned}$$

The final expression shows that $\widetilde{H}_{\text{GN}}$ is diagonal, as it is the Kronecker product of diagonal matrices scaled by a scalar factor. □

Finally, we connect our analysis to Adam's update rule by adapting the following result from Molybog et al [85]:

**Proposition 6** (Adapt from [85])**.** *Suppose that $\theta^*$ is a local minimum and assume that $\theta - \theta^* \sim \mathcal{N}(0, \sigma^2 I)$. For Adam update rule denoted by $\theta_{t+1} = \theta_t - \eta \operatorname{Adam}(g_t)$, we have*

$$\operatorname{Adam}(g_t) \approx \operatorname{diag}(H)^{-1} g_t. \tag{D.7}$$

*Proof.* The Adam optimizer follows the update rule:

$$\begin{aligned}
&m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \\
&v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t \odot g_t, \\
&\hat{m}_t = m_t / (1-\beta_1^t), \\
&\hat{v}t = v_t / (1-\beta_2^t), \\
&\theta_t = \theta t - 1 - \eta \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon).
\end{aligned}$$

Taking a first-order Taylor expansion of the gradient around a local minimum $\theta^*$:

$$g_\theta \approx g_{\theta^*} + H_{\theta^*}(\theta - \theta^*) \approx H_{\theta^*}(\theta - \theta^*).$$

This yields

$$g_\theta g_\theta^\top \approx H_{\theta^*}(\theta - \theta^*)(\theta - \theta^*)^\top H_{\theta^*}^\top.$$

Under our assumption that $\theta - \theta^* \sim \mathcal{N}(0, \sigma^2 I)$,

$$\mathbb{E}\left[g_\theta g_\theta^\top\right] \approx H_\theta \mathbb{E}\left[(\theta - \theta^*)(\theta - \theta^*)^\top\right] H_{\theta^*}^\top = \sigma^2 H_{\theta^*} H_{\theta^*}^\top.$$

By construction, $v_t$ approximates the diagonal of $\mathbb{E}_{\theta \sim \theta_\tau}[g_\theta g_\theta^T]$, where $\theta_\tau$ represents the distribution of model weights over the past $O(1/(1-\beta_2))$ steps:

$$v_t \approx \operatorname{diag}\left(\mathbb{E}_{\theta \sim \theta_\tau}\left[g_\theta g_\theta^T\right]\right) \approx \sigma^2 \operatorname{diag}(H_{\theta^*}^2).$$

Finally, assuming $m_t \approx g_t$:

$$\operatorname{Adam}(g_t) \approx \frac{m_t}{\sqrt{v_t} + \varepsilon} \approx \frac{m_t}{\sqrt{v_t}} \approx \operatorname{diag}(H)^{-1} g_t.$$

□

**Theorem 1.** *Under assumption of Proposition 6, SOAP's update approximates Newton's method:*

$$w_{t+1} = w_t - \eta \operatorname{Soap}(g_t) \approx w_t - \eta H^{-1} g_t. \tag{D.8}$$

*Proof.* Combining Propositions 5 and 6, we obtain

$$\operatorname{Adam}(\widetilde{G}_t) \approx \operatorname{diag}(\widetilde{H})^{-1}\widetilde{g}_t \approx \operatorname{diag}(\widetilde{H}_{\text{GN}})^{-1}\widetilde{g}_t = \widetilde{H}_{\text{GN}}^{-1}\widetilde{g}_t \approx \widetilde{H}^{-1}\widetilde{g}_t. \tag{D.9}$$

By Corollary 1, this is equivalent to the Newton update in the original space:

$$w_{t+1} = w_t - H^{-1} g_t. \tag{D.10}$$

As a direction implication, the Hessian matrix is approximately diagonal in the rotated space. □

The key insight is that SOAP effectively approximates the block-diagonal Gaussian Newton component of Hessian in a rotated space, with each block corresponding to a layer-wise Kronecker factorization. This structure naturally promotes gradient alignment across optimization steps, as we demonstrated theoretically in Proposition 4 and observed empirically in Figure 5.

**Remark 1.** *While SOAP effectively approximates Newton's method through its block-diagonal structure, other optimizers make different compromises in their approximations. Adam can approximate Newton's method, but requires the highly restrictive assumption that the Hessian matrix is diagonal. Similarly, Shampoo takes a different approach by using the square root of the Gauss-Newton component as its preconditioner [86]:*

$$\operatorname{Shampoo}(g_t) \approx H_{GN}^{-1/2} g_t \approx H^{-1/2} g_t. \tag{D.11}$$

These structural differences help explain why SOAP achieves better gradient alignment than both Adam and Shampoo.

# E Connection of Shampoo and Muon to Quasi-second-order Methods

We review the connections between Shampoo, Muon, and quasi-second-order optimization methods, building upon results from [87] and [86]. This section aims to provide a self-contained exposition of these relationships.

**Adagrad.** Adagrad is a preconditioned online learning algorithm that leverages the accumulated covariance of gradients as a preconditioner. Let $\theta_t \in \mathbb{R}^p$ denote the parameters at time $t$ and $g_t \in \mathbb{R}^p$ denote the corresponding gradient. Adagrad maintains a preconditioner $H_{\text{Ada}} = \sum_{t=1}^{T} g_t g_t^\top$. The parameter update with learning rate $\eta$ is given by:

$$\theta_{T+1} = \theta_T - \eta H_{\text{Ada}}^{-1/2} g_T \tag{E.1}$$

**Shampoo.** Shampoo tracks two statistical matrices throughout training, $L_t \in \mathbb{R}^{m\times m}$ and $R_t \in \mathbb{R}^{n\times n}$, defined as:

$$L_t = \epsilon I_m + \sum_{s=1}^{t} G_s G_s^\top; \quad R_t = \epsilon I_n + \sum_{s=1}^{t} G_s^\top G_s \tag{E.2}$$

where $G_s \in \mathbb{R}^{m\times n}$ is the gradient matrix at step $s$, and $\epsilon > 0$ is a small constant for numerical stability.

The full matrix Adagrad preconditioner $H_t$ can be approximated as $(L_t \otimes R_t)^{1/2}$. This approximation transforms the Adagrad update rule $w_{t+1} = w_t - \eta H_t^{-1/2} g_t$ into the Shampoo update rule for parameter matrix $W$:

$$W_{t+1} = W_t - \eta L_t^{-1/4} G_t R_t^{-1/4} \tag{E.3}$$

The theoretical foundation for this approximation is provided by the following lemma:

**Lemma 3** ([87], Lemma 1)**.** *Let $G_1, \ldots, G_t \in \mathbb{R}^{m\times n}$ be matrices of rank at most $r$. Let $g_s = \operatorname{vec}(G_s)$ and define $\widehat{H}_t = \epsilon I_{mn} + \sum_{s=1}^{t} g_s g_s^\top$. Define $L_t, R_t$ as above: $L_t = \epsilon I_m + \sum_{s=1}^{t} G_s G_s^\top$, $R_t = \epsilon I_n + \sum_{s=1}^{t} G_s^\top G_s$. Then for any $p, q > 0$ such that $1/p + 1/q = 1$, we have $\widehat{H}_t \le r L_t^{1/p} \otimes R_t^{1/q}$.*

It follows from this lemma that for any $p, q > 0$ with $1/p + 1/q = 1$, the full Adagrad preconditioned gradient $H_{\text{Ada}}^{-1/2} g_t$ can be approximated by $(L_t^{1/p} \otimes R_t^{1/q})^{-1/2} g_t = \operatorname{vec}(L_t^{-1/2p} G_t R_t^{-1/2q})$. The case where $p = q = 2$ yields the standard Shampoo update.

Moreover, [86] explored the Hessian approximation perspective of Shampoo, demonstrating that the preconditioner in Shampoo is a Kronecker product approximation of the Gauss-Newton component of the layerwise Hessian (see Remark 1).

**Muon.** The Muon optimizer [62] was recently proposed for optimizing neural network weights representable as matrices. At iteration $t$, given current weight $\mathbf{W}_{t-1}$, momentum $\mu$, learning rate $\eta_t$, and objective $\mathcal{L}_t$, the update rule of the Muon optimizer is:

$$\mathbf{M}_t = \mu \mathbf{M}_{t-1} + \nabla \mathcal{L}_t \left(\mathbf{W}_{t-1}\right), \tag{E.4}$$

$$\mathbf{O}_t = \text{Newton-Schulz}\left(\mathbf{M}_t\right), \tag{E.5}$$

$$\mathbf{W}_t = \mathbf{W}_{t-1} - \eta_t \mathbf{O}_t. \tag{E.6}$$

Here, $\mathbf{M}_t$ is the momentum of gradient at iteration $t$, initialized as a zero matrix when $t = 0$. The Newton-Schulz iteration process is adopted to approximately compute $(\mathbf{M}_t \mathbf{M}_t^{\mathrm{T}})^{-1/2} \mathbf{M}_t$.

When preconditioner accumulation is removed, we can observe that the update simplifies to [62]:

$$W_{t+1} = W_t - \eta (G_t G_t^\top)^{-1/4} G_t (G_t^\top G_t)^{-1/4} \tag{E.7}$$

$$= W_t - \eta (U S^2 U^\top)^{-1/4} (U S V^\top)(V S^2 V^\top)^{-1/4} \tag{E.8}$$

$$= W_t - \eta (U S^{-1/2} U^\top)(U S V^\top)(V S^{-1/2} V^\top) \tag{E.9}$$

$$= W_t - \eta U S^{-1/2} S S^{-1/2} V^\top \tag{E.10}$$

$$= W_t - \eta U V^\top \tag{E.11}$$

From this derivation, Muon can be viewed as approximating the use of $H_{\text{Ada}}^{-1/2}$ as a preconditioner, with additional orthogonalization benefits.

# F Additional Lemma and Proof

**Lemma 4** ([86], Corollary 2)**.** *Under the assumption that the reshaping of the Hessian tensor $H_{GN}$ is rank-1,*

$$\hat{H}_{GN} = \left(\mathbb{E}\left[GG^\top\right] \otimes \mathbb{E}\left[G^\top G\right]\right) / \operatorname{Tr}\left(\mathbb{E}\left[GG^\top\right]\right).$$

## F.1 Proof of Proposition 1

*Proof.* When $n = 2$, we note

$$\begin{aligned}
\mathcal{A}(v_1, v_2) &= 2 \left\| \frac{\frac{v_1}{\|v_1\|} + \frac{v_2}{\|v_2\|}}{2} \right\|^2 - 1 \\
&= \frac{1}{2} \left( \left\| \frac{v_1}{\|v_1\|} \right\|^2 + 2 \frac{v_1 \cdot v_2}{\|v_1\| \|v_2\|} + \left| \frac{v_2}{\|v_2\|} \right|^2 \right) - 1 \\
&= \frac{1}{2}(1 + 2\cos(v_1, v_2) + 1) - 1 = \cos(v_1, v_2).
\end{aligned}$$

□

## F.2 Proof of Proposition 2

*Proof.* **(i) Single-step drop.** By $L$-smoothness (descent lemma),

$$\mathcal{L}(\theta_{t+1}) \le \mathcal{L}(\theta_t) + \langle g_t, \theta_{t+1} - \theta_t \rangle + \tfrac{L}{2} \|\theta_{t+1} - \theta_t\|^2.$$

With $\theta_{t+1} - \theta_t = -\eta h_t$,

$$\Delta_t \;=\; \mathcal{L}(\theta_t) - \mathcal{L}(\theta_{t+1}) \;\geq\; \eta\langle g_t, h_t\rangle - \tfrac{L\eta^2}{2}\|h_t\|^2.$$

Since $h_t = P_t g_t$ and $P_t^{-1} \succeq \frac{1}{M} I$, we have

$$\langle g_t, h_t\rangle = h_t^\top P_t^{-1} h_t \;\geq\; \tfrac{1}{M}\|h_t\|^2,$$

hence

$$\Delta_t \;\geq\; \Big(\tfrac{\eta}{M} - \tfrac{L\eta^2}{2}\Big)\|h_t\|^2.$$

Under the equal-norm assumption $\|h_t^i\| = \lambda$, let $u_t^i := h_t^i/\lambda$. By the chosen definition of $A_t^{\text{intra}}$,

$$\Big\|\sum_{i=1}^n u_t^i\Big\|^2 = \tfrac{n^2}{2}\big(A_t^{\text{intra}} + 1\big).$$

Since $h_t = \sum_i h_t^i = \lambda \sum_i u_t^i$ and $\sum_i \|h_t^i\|^2 = n\lambda^2$, we obtain the displayed equal-norm formulas and the refined bound.

**(ii) Two-step cumulative drop.** Apply $L$-smoothness with $\theta = \theta_{t-1}$, $\phi = \theta_{t+1}$:

$$\mathcal{L}(\theta_{t+1}) \leq \mathcal{L}(\theta_{t-1}) + \langle g_{t-1}, \theta_{t+1} - \theta_{t-1}\rangle + \tfrac{L}{2}\|\theta_{t+1} - \theta_{t-1}\|^2.$$

Since $\theta_{t+1} - \theta_{t-1} = -\eta(h_{t-1} + h_t)$, we get

$$\Delta_{t-1} + \Delta_t \;\geq\; \eta\langle g_{t-1}, h_{t-1} + h_t\rangle - \tfrac{L\eta^2}{2}\|h_{t-1} + h_t\|^2.$$

For the linear term, using $P_{t-1}^{-1} \succeq \frac{1}{M} I$,

$$\langle g_{t-1}, h_{t-1}\rangle = h_{t-1}^\top P_{t-1}^{-1} h_{t-1} \;\geq\; \tfrac{1}{M}\|h_{t-1}\|^2,$$

and

$$\langle g_{t-1}, h_t\rangle = h_{t-1}^\top P_{t-1}^{-1} h_t \;\geq\; \tfrac{1}{M}\langle h_{t-1}, h_t\rangle.$$

Let $a := \|h_{t-1}\|$, $b := \|h_t\|$, and $A_t^{\text{inter}} := \frac{\langle h_{t-1}, h_t\rangle}{ab}$. Then

$$\eta\langle g_{t-1}, h_{t-1} + h_t\rangle \;\geq\; \tfrac{\eta}{M}\big(a^2 + ab\, A_t^{\text{inter}}\big).$$

For the quadratic term, $\|h_{t-1} + h_t\|^2 = a^2 + 2ab\, A_t^{\text{inter}} + b^2$. Putting the bounds together,

$$\Delta_{t-1} + \Delta_t \;\geq\; \tfrac{\eta}{M}\big(a^2 + ab\, A_t^{\text{inter}}\big) - \tfrac{L\eta^2}{2}\big(a^2 + 2ab\, A_t^{\text{inter}} + b^2\big),$$

which rearranges to the stated inequality. The coefficient of $A_t^{\text{inter}}$ equals $\frac{\eta}{M}(1 - L\eta M) \geq 0$ for $\eta \leq 1/(LM)$, hence monotonicity. □

# G Experimental Details

## G.1 Architectures

This section outlines the network architectures employed in our work, along with the enhancements introduced to improve their performance.

**Modified MLP.** The modified MLP architecture is proposed by [21], which has been extensively used in the literature [35, 73, 88, 54, 89, 90] due to its improved capability in learning complex PDE solutions. The network processes input coordinates through two parallel encoders:

$$\mathbf{U} = \sigma\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right), \quad \mathbf{V} = \sigma\left(\mathbf{W}_2\mathbf{x} + \mathbf{b}_2\right). \tag{G.1}$$

Then, for $l = 1, 2, \ldots, L$, the forward pass is defined as:

$$\mathbf{f}^{(l)}(\mathbf{x}) = \mathbf{W}^{(l)} \cdot \mathbf{g}^{(l-1)}(\mathbf{x}) + \mathbf{b}^{(l)}, \tag{G.2}$$

$$\mathbf{g}^{(l)}(\mathbf{x}) = \sigma\left(\mathbf{f}_\theta^{(l)}(\mathbf{x})\right) \odot \mathbf{U} + \left(1 - \sigma\left(\mathbf{f}_\theta^{(l)}(\mathbf{x})\right)\right) \odot \mathbf{V} \tag{G.3}$$

The final network output is given by

$$\mathbf{f}_\theta(\mathbf{x}) = \mathbf{W}^{(L+1)} \cdot \mathbf{g}^{(L)}(\mathbf{x}) + \mathbf{b}^{(L+1)}. \tag{G.4}$$

where $\sigma$ is a nonlinear activation function, $\odot$ denotes element-wise multiplication, and the trainable parameters are:

$$\theta = \left\{ \mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \left( \mathbf{W}^{(l)}, \mathbf{b}^{(l)} \right)_{l=1}^{L+1} \right\}. \tag{G.5}$$

This architecture extends the standard MLP by incorporating dual input encoders and merging their features through point-wise multiplication at each hidden layer. While computationally more demanding, this modification demonstrates superior performance in minimizing PDE residuals compared to standard MLPs.

**PirateNet.** PirateNet is proposed by [27], which aims to enable stable and efficient training of deep PINN models. The architecture first transforms input coordinates $\mathbf{x}$ into a high-dimensional feature space using random Fourier features [91]:

$$\Phi(\mathbf{x}) = \begin{bmatrix} \cos(\mathbf{B}\mathbf{x}) \\ \sin(\mathbf{B}\mathbf{x}) \end{bmatrix},$$

where $\mathbf{B} \in \mathbb{R}^{m \times d}$ has entries sampled i.i.d. from $\mathcal{N}(0, s^2)$ with user-specified $s > 0$. This embedding mitigates spectral bias in PINNs by improving the eigenfunction frequency of the Neural Tangent Kernel, enabling better learning of high-frequency components and multiscale features [30].

The embedded coordinates are processed through two dense layers that act as gates:

$$\mathbf{U} = \sigma(\mathbf{W}_1 \Phi(\mathbf{x}) + \mathbf{b}_1), \quad \mathbf{V} = \sigma(\mathbf{W}_2 \Phi(\mathbf{x}) + \mathbf{b}_2),$$

where $\sigma$ is a point-wise activation function. This gating mechanism is essentially the same as in modified MLP.

Let $\mathbf{x}^{(1)} = \Phi(\mathbf{x})$ and $\mathbf{x}^{(l)}$ be the input to the $l$-th block ($1 \leq l \leq L$). Each block performs:

$$\mathbf{f}^{(l)} = \sigma\left(\mathbf{W}_1^{(l)} \mathbf{x}^{(l)} + \mathbf{b}_1^{(l)}\right), \tag{G.6}$$

$$\mathbf{z}_1^{(l)} = \mathbf{f}^{(l)} \odot \mathbf{U} + (1 - \mathbf{f}^{(l)}) \odot \mathbf{V}, \tag{G.7}$$

$$\mathbf{g}^{(l)} = \sigma\left(\mathbf{W}_2^{(l)} \mathbf{z}_1^{(l)} + \mathbf{b}_2^{(l)}\right), \tag{G.8}$$

$$\mathbf{z}_2^{(l)} = \mathbf{g}^{(l)} \odot \mathbf{U} + (1 - \mathbf{g}^{(l)}) \odot \mathbf{V}, \tag{G.9}$$

$$\mathbf{h}^{(l)} = \sigma\left(\mathbf{W}_3^{(l)} \mathbf{z}_2^{(l)} + \mathbf{b}_3^{(l)}\right), \tag{G.10}$$

$$\mathbf{x}^{(l+1)} = \alpha^{(l)} \mathbf{h}^{(l)} + (1 - \alpha^{(l)}) \mathbf{x}^{(l)}, \tag{G.11}$$

Each block comprises three dense layers with dual gating operations and an adaptive residual connection. The trainable $\alpha^{(l)}$ parameters control block nonlinearity: $\alpha^{(l)} = 0$ yields an identity mapping, while $\alpha^{(l)} = 1$ produces fully nonlinear transformation.

The final output of a PirateNet of $L$ residual blocks is given by

$$\mathbf{u}_\theta = \mathbf{W}^{(L+1)} \mathbf{x}^{(L)}. \tag{G.12}$$

Importantly, we initialize $\alpha^{(l)} = 0$, making the initial output a linear combination of first-layer embeddings. This initialization strategy mitigates training difficulties in deep networks by starting with effectively shallow architecture and gradually increasing depth through learned $\alpha$ values. Additionally, the linear structure at initialization enables direct integration of prior solution data through least squares fitting:

$$\min_{\mathbf{W}} \|\mathbf{W}\Phi - \mathbf{Y}\|_2^2, \tag{G.13}$$

where $\mathbf{Y}$ represents available measurements. This approach provides an optimal initial guess based on various data sources, including experimental measurements, boundary conditions, or linearized PDE solutions.

**Exact imposition of periodic boundary conditions.** We adopt the approach of [70] to enforce periodic boundary conditions as hard constraints, improving both training convergence and accuracy. Consider a one-dimensional periodic function with period $P$ satisfying:

$$u^{(l)}(a) = u^{(l)}(a + P), \quad l = 0, 1, 2, \ldots. \tag{G.14}$$

We construct a Fourier feature embedding:

$$\mathbf{v}(x) = (\cos(\omega x), \sin(\omega x)), \tag{G.15}$$

where $\omega = \frac{2\pi}{L}$. Any network $u_\theta(v(x))$ using this embedding inherently satisfies the periodic boundary condition.

The same idea can be directly extended to higher-dimensional domains. For two-dimensional domains, the periodicity constraints are:

$$\frac{\partial^l}{\partial x^l} u(a, y) = \frac{\partial^l}{\partial x^l} u(a + P_x, y), \quad y \in [b, b + P_y], \tag{G.16}$$

$$\frac{\partial^l}{\partial y^l} u(x, a) = \frac{\partial^l}{\partial y^l} u(x, b + P_y), \quad x \in [a, a + P_x], \tag{G.17}$$

for $l = 0, 1, 2, \ldots$, where $P_x$ and $P_y$ are the periods in the $x$ and $y$ directions, respectively. Similarly, these constraints are encoded using the embedding:

$$\mathbf{v}(x, y) = [\cos(\omega_x x), \sin(\omega_x x), \cos(\omega_y y), \sin(\omega_y y)] \tag{G.18}$$

with $w_x = \frac{2\pi}{P_x}, w_y = \frac{2\pi}{P_y}$.

For time-dependent problems, we concatenate time coordinates $t$ with spatial embeddings: $u_\theta([t, \mathbf{v}(x)])$ or $u_\theta([t, \mathbf{v}(x, y)])$.

**Random weight factorization.** We implement random weight factorization (RWF) [69] to enhance PINN performance. RWF decomposes each neuron's weight vector as:

$$\mathbf{w}^{(k,l)} = s^{(k,l)} \cdot \mathbf{v}^{(k,l)}, \tag{G.19}$$

where $k = 1, \ldots, d_l$, $l = 1, \ldots, L + 1$, $\mathbf{w}^{(k,l)} \in \mathbb{R}^{d_{l-1}}$ is the $k$-th row of weight matrix $\mathbf{W}^{(l)}$, $s^{(k,l)} \in \mathbb{R}$ is a trainable scale factor, and $\mathbf{v}^{(k,l)} \in \mathbb{R}^{d_{i-1}}$. This factorization can be expressed in matrix form as:

$$\mathbf{W}^{(l)} = \operatorname{diag}\left(\mathbf{s}^{(l)}\right) \cdot \mathbf{V}^{(l)}, \quad l = 1, 2, \ldots, L + 1 \tag{G.20}$$

with $\mathbf{s}^{(l)} \in \mathbb{R}^{d_t}$.

Implementation involves: (1) initializing MLP parameters using the Glorot scheme [92], (2) initializing scale vectors $\exp(s)$ where $s \sim \mathcal{N}(\mu, \sigma \mathrm{I})$, (3) factorizing each weight matrix as $\mathbf{W} = \operatorname{diag}(\exp(\mathbf{s})) \cdot \mathbf{V}$, and (4) optimizing parameters $\mathbf{s}, \mathbf{V}$ directly. We employ exponential parameterization following Weight Normalization [93] to ensure non-zero scale factors across varied magnitudes. We recommend $\mu = 0.5$ or 1 and $\sigma = 0.1$, as these values consistently improve convergence and accuracy while avoiding the instability of larger values or the diminished effect of smaller ones.

### G.2 Training pipeline

This section details the methodologies and strategies used to train PINN models.

**Causal training.** Recent work by [73] shows that PINNs may violate temporal causality when solving time-dependent PDEs, as they tend to minimize residuals at later times before correctly solving earlier times. To address this, we introduce a causality-aware training approach. We partition the temporal domain into $M$ equal segments and denote the PDE residual loss within the $i$-th segment as $L_r^i$. The modified residual loss becomes:

$$\mathcal{L}_r(\theta) = \frac{1}{M} \sum_{i=1}^{M} w_i \mathcal{L}_r^i(\theta). \tag{G.21}$$

We compute the temporal weights as

$$w_i = \exp\left(-\epsilon \sum_{k=1}^{i-1} \mathcal{L}_r^k(\theta)\right), \text{ for } i = 2, 3, \ldots, M. \tag{G.22}$$

Then,

$$\mathcal{L}_r(\theta) = \frac{1}{M} \sum_{i=1}^{M} \exp\left(-\epsilon \sum_{k=1}^{i-1} \mathcal{L}_r^k(\theta)\right) \mathcal{L}_r^i(\theta). \tag{G.23}$$

The weight $w_i$ decreases exponentially with the cumulative residual loss from previous time steps. This ensures that $\mathcal{L}_r^i(\theta)$ is minimized only after previous residuals $\{\mathcal{L}_r^k(\theta)\}_{k=1}^{i-1}$ become sufficiently small, enforcing temporal causality in the optimization process.

The causality parameter $\epsilon$ requires careful tuning: small values may insufficiently enforce causality, while large values can create optimization difficulties by requiring extremely small early-time residuals before later times are considered. We recommend selecting a moderate $\epsilon$ that allows all temporal weights to converge to 1 by training completion, reducing it if necessary.

**Learning rate annealing.** Another key challenge in training PINNs is balancing loss components, as they often exhibit multi-scale behaviors, resulting in unbalanced gradients and training failures.

We implement a self-adaptive learning rate annealing algorithm [21] that automatically balances the weighted loss:

$$\mathcal{L}(\theta) = \lambda_{ic}\mathcal{L}_{ic}(\theta) + \lambda_{bc}\mathcal{L}_{bc}(\theta) + \lambda_r\mathcal{L}_r(\theta), \tag{G.24}$$

The global weights are dynamically computed to equalize the gradient norms of each loss component:

$$\hat{\lambda}_{ic} = \frac{\|\nabla_\theta \mathcal{L}_{ic}(\theta)\| + \|\nabla_\theta \mathcal{L}_{bc}(\theta)\| + \|\nabla_\theta \mathcal{L}_r(\theta)\|}{\|\nabla_\theta \mathcal{L}_{ic}(\theta)\|}, \tag{G.25}$$

$$\hat{\lambda}_{bc} = \frac{\|\nabla_\theta \mathcal{L}_{ic}(\theta)\| + \|\nabla_\theta \mathcal{L}_{bc}(\theta)\| + \|\nabla_\theta \mathcal{L}_r(\theta)\|}{\|\nabla_\theta \mathcal{L}_{bc}(\theta)\|}, \tag{G.26}$$

$$\hat{\lambda}_r = \frac{\|\nabla_\theta \mathcal{L}_{ic}(\theta)\| + \|\nabla_\theta \mathcal{L}_{bc}(\theta)\| + \|\nabla_\theta \mathcal{L}_r(\theta)\|}{\|\nabla_\theta \mathcal{L}_r(\theta)\|}, \tag{G.27}$$

where $\|\cdot\|$ denotes the $L^2$ norm. Then we obtain

$$\|\hat{\lambda}_{ic}\nabla_\theta \mathcal{L}_{ic}(\theta)\| = \|\hat{\lambda}_{bc}\nabla_\theta \mathcal{L}_{ic}(\theta)\| = \|\hat{\lambda}_r\nabla_\theta \mathcal{L}_{ic}(\theta)\| = \|\nabla_\theta \mathcal{L}_{ic}(\theta)\| + \|\nabla_\theta \mathcal{L}_{bc}(\theta)\| + \|\nabla_\theta \mathcal{L}_r(\theta)\|. \tag{G.28}$$

This formulation equalizes the gradient norms of weighted losses, preventing bias toward any particular term during training. The weights are updated as running averages of their previous values, stabilizing stochastic gradient descent. These updates occur at user-specified intervals (typically every 100-1000 iterations), incurring minimal computational overhead.

**Curriculum training and time-marching.** Despite the improvements described above, PINNs still face challenges in complex domains requiring high accuracy, such as chaotic systems like high Reynolds number Navier-Stokes equations where error accumulation can cause trajectory divergence. We address these challenges using curriculum training [42], which decomposes the optimization into more manageable sub-tasks.

An effective approach we employ is the curriculum training strategy introduced by [42]. The core idea involves decomposing the entire optimization task for PINNs into a sequence of more manageable sub-tasks. In this work, we mainly focus on integrating this strategy into our training pipeline for solving time-dependent PDEs and singular perturbation problems.

For time-dependent PDEs, we implement temporal domain decomposition: the time domain is divided into smaller intervals. After the first window, initial conditions for subsequent windows are set using predictions from the final step of the previous window. This approach reduces the difficulty of the optimization task of learning full system dynamics, though at an increased computational cost due to per-window model retraining.

Table 5: Parameter settings and numerical configurations for generating the reference solution across PDE benchmarks.

| **PDE** | Parameter | Package | Resolution |
|---|---|---|---|
| **Wave** | $c = 4$ | N/A | $200 \times 128$ |
| **Burgers** | $\nu = 0.01\ \pi$ | Chebfun | $200 \times 512$ |
| **AC** | $\epsilon = 10^{-4}, a = 5$ | Chebfun | $200 \times 512$ |
| **KdV** | $\eta = 1, \mu = 0.022$ | Chebfun | $200 \times 512$ |
| **KS** | $\alpha = 100/16, \beta = 100/16^2, \gamma = 100/16^4$ | Chebfun | $250 \times 512$ |
| **GS** | $\epsilon_1 = 0.2, \epsilon_2 = 0.1, b_1 = 40, b2 = 100, c_1 = c_2 = 1,000$ | Chebfun | $100 \times 200 \times 200$ |
| **GL** | $\epsilon = 0.004, \mu = 10, \gamma = 10 + 15i$ | Chebfun | $100 \times 200 \times 200$ |
| **LDC** | Re=$5\times10^3$ | IncompressibleNavierStokes | $128 \times 128$ |
| **KF** | Re=$10^4$ | IncompressibleNavierStokes | $50 \times 512 \times 512$ |
| **RT** | Ra=$10^6$, Pr $= 0.71$ | IncompressibleNavierStokes | $40 \times 100 \times 200$ |

While we also partition the temporal domain to compute causal weights within each window, this differs from the time-marching strategy. Both techniques promote learning solutions sequentially along the time axis to respect causality, but causal weighting complements rather than replaces time-marching, as causality violations may still occur within individual time windows.

### G.3 Data Generation

We generate our reference dataset using two numerical packages: Chebfun [94] in MATLAB and IncompressibleNavierStokes [95] in Julia. The data generation process employs a time step of $dt = 10^{-4}$, followed by temporal downsampling to construct the final dataset. Table 5 summarizes the PDE parameters and dataset details.

### G.4 Hyper-parameters

Unless otherwise specified, we adopt the following hyperparameter configuration for our experiments.

**Architecture.** We employ PirateNet [27] as our backbone architecture, configured with three residual blocks (9 layers in total), a hidden dimension of 256, and `Tanh` activation functions. All weight matrices are initialized using random weight factorization (RWF) [69] with parameters $\mu = 1.0$ and $\sigma = 0.1$. When applicable, we strictly enforce exact periodic boundary conditions following [70].

**Training Protocol.** We train our models using mini-batch gradient descent with 8,192 randomly sampled collocation points per iteration. The learning rate schedule comprises an initial linear warm-up phase over the first 5,000 steps (from 0 to $10^{-3}$), followed by exponential decay with a factor of 0.9. To enhance training stability and convergence, we implement learning rate annealing for loss balancing [21, 72], updating the loss weights every 1,000 iterations with a moving average. For time-dependent PDEs, we apply causal training [73, 72] with a causal tolerance of 1.0. Additionally, we leverage time-marching and curriculum learning strategies [42] for particularly challenging benchmarks.

**Optimizers.** We evaluate several optimizers with the following configurations:

- **Adam:** We use the Adam optimizer [71] with standard hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, which has become the de facto standard for training PINNs due to its robust performance and computational efficiency.
- **SOAP [1]:** Based on our ablation studies presented in Figure 4, we select $\beta_1 = 0.99$ and $\beta_2 = 0.999$ for SOAP, which yield optimal performance across our experimental tasks.
- **Muon [62]:** For Muon, we adopt momentum hyperparameters matching those of Adam: $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Notably, we employ more accurate Newton-Schulz coefficients $(2, -1.5, 0.5)$ and implement 10 Newton-Schulz matrix iterations to ensure convergence of the orthogonalization procedure.

Table 6: *Hyperparameter configurations for benchmark PDEs.* Hyperparameter settings used to reproduce our experimental results. The backbone architecture is PirateNet, where Depth indicates the number of adaptive residual blocks, and Width denotes the number of neurons per hidden layer. RFF and RWF represent Random Fourier Features and Random Weight Factorization, respectively.

| **Parameter** | **Wave** | **Burgers** | **AC** | **KdV** | **KS** | **GS** | **GL** | **LDC** | **KF** | **RT** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Architecture** | | | | | | | | | | |
| Depth | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 3 | 3 |
| Width | 256 | 256 | 256 | 256 | 256 | 256 | 256 | 256 | 384 | 384 |
| Activation | Tanh | Tanh | Tanh | Tanh | Tanh | Swish | Swish | Tanh | Tanh | Tanh |
| RFF scale | 10.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 10.0 | 2.0 | 2.0 |
| RWF | $\mu$=1.0, $\sigma$=0.1 | | | | | | | | | |
| **Learning rate schedule** | | | | | | | | | | |
| Initial learning rate | $10^{-3}$ | | | | | | | | | |
| Decay rate | 0.9 | | | | | | | | | |
| Decay steps | $2 \times 10^3$ | $2 \times 10^3$ | $5 \times 10^3$ | $2 \times 10^3$ | $2 \times 10^3$ | $2 \times 10^3$ | $2 \times 10^3$ | $2 \times 10^3$ | $2 \times 10^3$ | $2 \times 10^3$ |
| Warmup steps | $5 \times 10^3$ | | | | | | | | | |
| **Training** | | | | | | | | | | |
| Iters (per time window) | $10^5$ | $10^5$ | $3{\times}10^5$ | $10^5$ | $10^5$ | $10^5$ | $10^5$ | $2{\times}10^5$ | $2{\times}10^4$ | $10^5$ |
| Batch size | 8,192 | | | | | | | | | |
| # Time windows | 1 | 1 | 1 | 1 | 10 | 10 | 5 | N / A | 25 | 4 |
| **Weighting Scheme** | Grad Norm | | | | | | | | | |
| **Causal weighting** | | | | | | | | | | |
| Tolerance | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | N / A | 1.0 | 1.0 |
| # Chunks | 16 | 16 | 16 | 16 | 16 | 16 | 16 | N / A | 16 | 16 |

- **Kron [63]:** For Kronecker-factored optimization, we use the same momentum hyperparameters as Adam: $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

The complete set of hyperparameters is detailed in Table 6, largely following the configurations established in [72] and [27]. The decay step is tailored for each benchmark to ensure the learning rate reaches a sufficiently small value (approximately $10^{-7}$) by the end of training. The number of time windows is determined empirically based on problem complexity, with fine-tuning guided by the loss convergence behavior observed during preliminary experiments.

### G.5 Computational Cost

Our implementation is based on JAX-PI [72] and we conducted all experiments on a single NVIDIA A6000 GPU, with detailed runtime benchmarks reported in Table 7.

### G.6 Benchmarks

**Wave equation.** We consider a one-dimensional wave equation in the domain $\Omega = [0, 1] \times [0, 1]$ taking the form

$$
\begin{aligned}
&u_{tt}(x,t) - 4u_{xx}(x,t) = 0, \quad (x,t) \in (0,1) \times (0,1),\\
&u(0,t) = u(1,t) = 0, \quad t \in [0,1],\\
&u(x,0) = \sin(\pi x) + \frac{1}{2}\sin(4\pi x), \quad x \in [0,1],\\
&u_t(x,0) = 0, \quad x \in [0,1].
\end{aligned}
$$

where $u$ represents the wave amplitude, and $c$ is the wave propagation speed, determined by the medium's physical properties.

By d'Alembert's formula, the solution $u(x,t)$ is given by

$$
u(x,t) = \sin(\pi x)\cos(2\pi t) + \frac{1}{2}\sin(4\pi x)\cos(8\pi t).
$$

Table 7: Computational runtime (in hours) comparison of different methods across various PDEs. All experiments are performed on an Nvidia A6000 GPU, reporting the total training time needed to achieve convergence using PINNs with Adam and SOAP, respectively

| **Benchmark** | **Adam** | **SOAP** |
|---|---|---|
| Wave | 2.80 | 4.35 |
| Burgers | 1.18 | 4.05 |
| Allen-Cahn | 1.48 | 5.83 |
| Korteweg–De Vries | 1.61 | 3.90 |
| Kuramoto-Sivashinsky | 19.51 | 34.16 |
| Grey-Scott | 19.52 | 40.01 |
| Ginzburg-Landau | 15.98 | 23.75 |
| Lid-driven cavity ($\mathrm{Re} = 5 \times 10^3$) | 5.67 | 8.25 |
| Kolmogorov flow ($\mathrm{Re} = 10^4$) | 9.56 | 11.00 |
| Rayleigh-Taylor instability ($\mathrm{Pr} = 0.71, \mathrm{Ra} = 10^6$) | 20.23 | 21.73 |

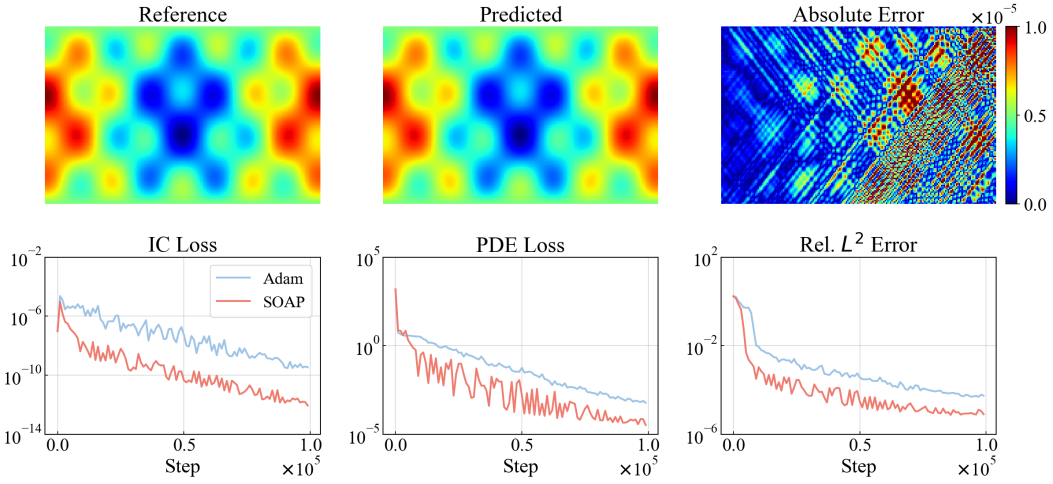


Figure 6: *Wave equation.* Top: Comparison between the reference solution and the model predictions. Bottom: Training loss and test error trajectories for the Adam and SOAP optimizers.

**Burgers equation.** The 1D Burgers equation is defined as:

$$u_t + uu_x = \nu u_{xx},$$

where $u$ represents the velocity field, and $\nu$ is the kinematic viscosity coefficient controlling the diffusion strength. Here we set $(x, t) \in \Omega = [-1, 1] \times [0, 1]$, with initial and boundary conditions:

$$u(x, 0) = -\sin(\pi x),$$
$$u(-1, t) = u(1, t) = 0,$$

and viscosity parameter $\nu = 0.01/\pi$.

**Allen-Cahn equation.** We investigate the one-dimensional Allen-Cahn equation with periodic boundary conditions:

$$u_t - 0.0001u_{xx} + 5u^3 - 5u = 0\,, \quad t \in [0, 1]\,,\ x \in [-1, 1]\,,$$
$$u(0, x) = x^2 \cos(\pi x)\,,$$
$$u(t, -1) = u(t, 1)\,, \quad u_x(t, -1) = u_x(t, 1)\,.$$

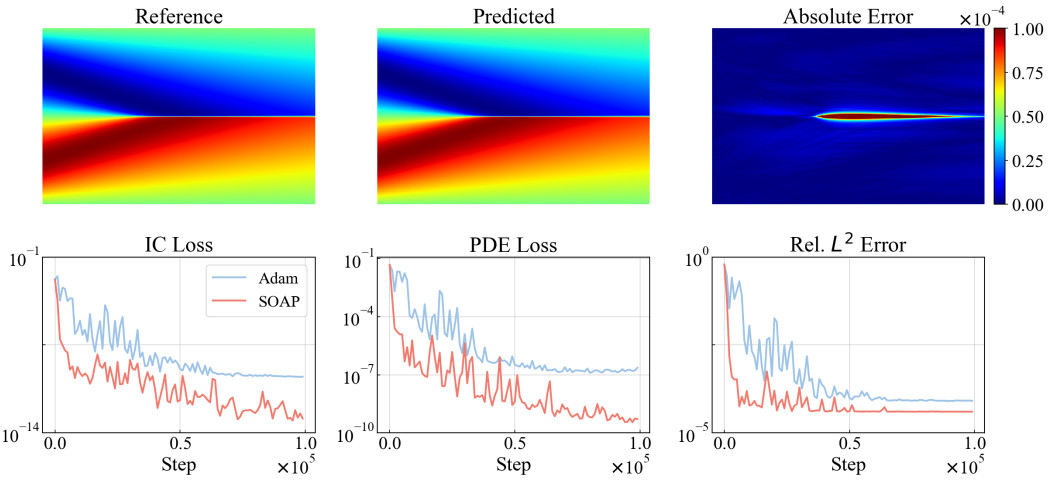


Figure 7: *Burgers' equation.* Top: Comparison between the reference solution and model predictions. Bottom: Training loss and test error trajectories for the Adam and SOAP optimizers.

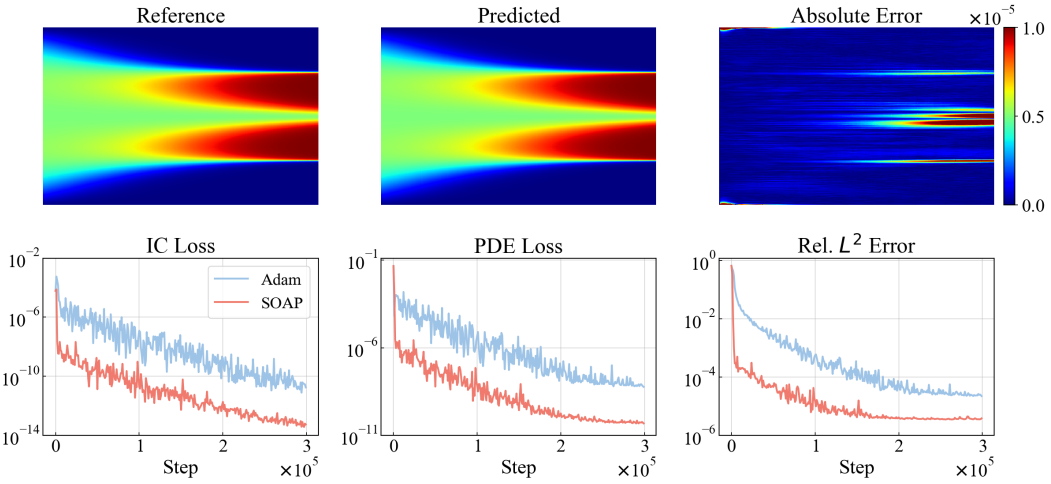


Figure 8: *Allen-Cahn equation.* Top: Comparison between the reference solution and model predictions. Bottom: Training loss and test error trajectories for the Adam and SOAP optimizers.

where $u$ represents the order parameter (e.g., concentration difference between two phases), $\epsilon$ controls the interfacial width, $a$ is the reaction rate coefficient, and the term $(u - u^3)$ drives the phase separation.

It is worth noting that this benchmark has been extensively used to validate the effectiveness of PINNs methodologies. In Table 8, we compare the test errors across different PINNs advancements, demonstrating that our approach achieves state-of-the-art performance with an improvement of up to one order of magnitude in accuracy.

**Korteweg–De Vries equation.** The one-dimensional KdV equation is expressed as follows:

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

where $u$ represents the wave amplitude or water surface elevation, and $\eta$ governs the strength of the nonlinearity, while $\mu$ controls the dispersion level. Under the KdV dynamics, this initial wave evolves into a series of solitary-type waves.

Table 8: *Allen-Cahn equation:* Relative $L^2$ test errors obtained by different PINNs variants.

| **Method** | **Relative $L^2$ error** |
|---|---|
| Original formulation of Raissi *et al.* [60] | $4.98 \times 10^{-1}$ |
| Adaptive time sampling [41] | $2.33 \times 10^{-2}$ |
| Self-attention [96] | $2.10 \times 10^{-2}$ |
| Time marching [97] | $1.68 \times 10^{-2}$ |
| Causal training [73] | $1.39 \times 10^{-4}$ |
| Dirac delta function causal training [98] | $6.29 \times 10^{-5}$ |
| JAX-PI [72] | $5.37 \times 10^{-5}$ |
| RBA-PINNs [88] | $4.55 \times 10^{-5}$ |
| PirateNet [27] | $2.24 \times 10^{-5}$ |
| BRDR-PINNs [54] | $1.45 \times 10^{-5}$ |
| **Ours** | $\mathbf{3.48 \times 10^{-6}}$ |

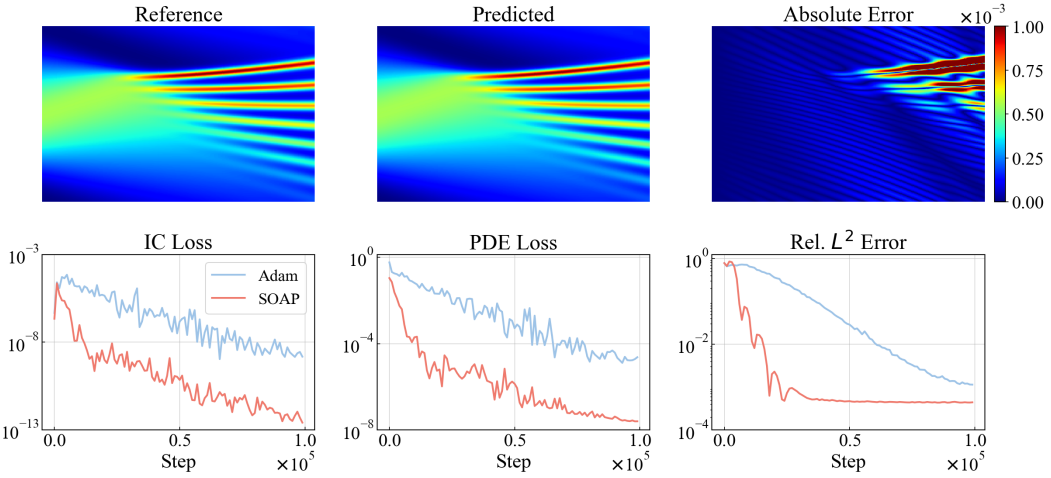


Figure 9: *Korteweg–De Vries equation.* Top: Comparison between the reference solution and model predictions. Bottom: Training loss and test error trajectories for the Adam and SOAP optimizers.

For our study, we adopt the classical parameters of the KdV equation, setting $\eta = 1$ and $\mu = 0.022$ [99].

**Kuramoto-Sivashinsky equation.** The one-dimensional equation takes the form:

$$u_t + \alpha u u_x + \beta u_{xx} + \gamma u_{xxxx} = 0, \quad t \in [0, T], x \in [0, 2\pi],$$
$$u(0, x) = u_0(x),$$

where $u$ represents the height of a thin film or flame front. This equation arises in various physical contexts, including flame front propagation, thin film flows, and plasma instabilities.

In this example, we take $T = 0.8$, $\alpha = 100/16, \beta = 100/16^2, \gamma = 100/16^4$ and $u_0(x) = \cos(x)(1 + \sin(x))$.

**Grey-Scott equation.** The system is described by the following coupled PDEs:

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

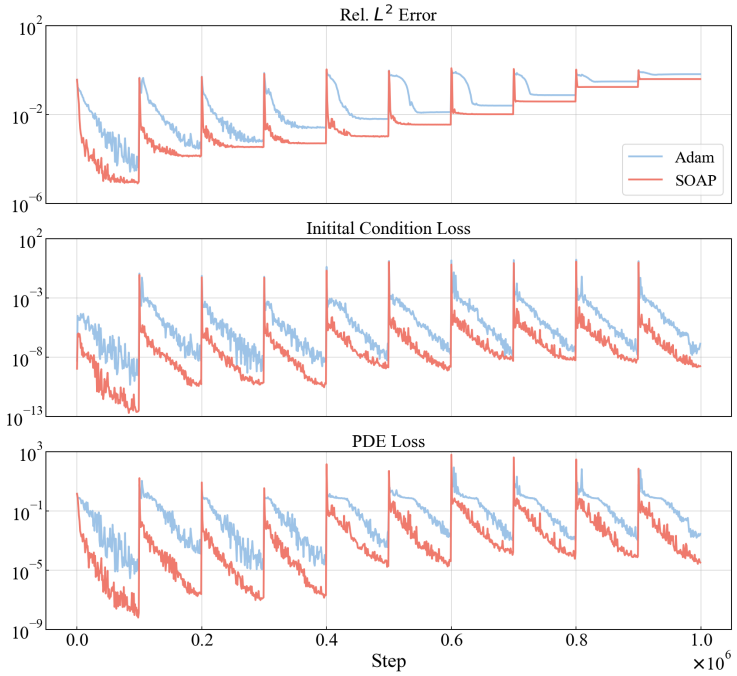


Figure 10: *Kuramoto-Sivashinsky equation.* Training loss and test error trajectories for the Adam and SOAP optimizers.

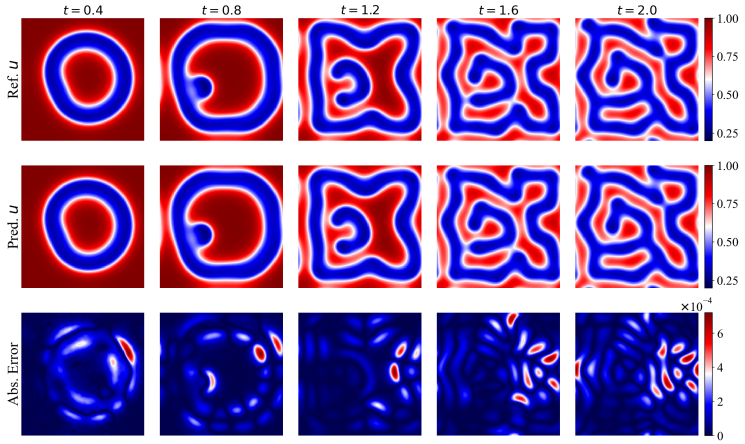


Figure 11: *Grey-Scott equation.* Comparison between reference solution and model predictions.

With periodic boundary conditions, the initial conditions are:

$$u_0(x, y) = 1 - \exp(-10((x + 0.05)^2 + (y + 0.02)^2)),$$
$$v_0(x, y) = 1 - \exp(-10((x - 0.05)^2 + (y - 0.02)^2)).$$

where $u$ and $v$ represent activator and inhibitor concentrations respectively, $\varepsilon_1$ and $\varepsilon_2$ are diffusion coefficients, and $(b_1, b_2, c_1, c_2)$ control reaction kinetics. This system generates diverse spatial patterns including spots and stripes.

We set parameters $\epsilon_1 = 0.2$, $\epsilon_2 = 0.1$, $b_1 = 40$, $b_2 = 100$, and $c_1 = c_2 = 1,000$, which generates characteristic pattern formations. Due to the similar behavior of $u$ and $v$, we report only the relative $L^2$ error of $u$ in Table 3.

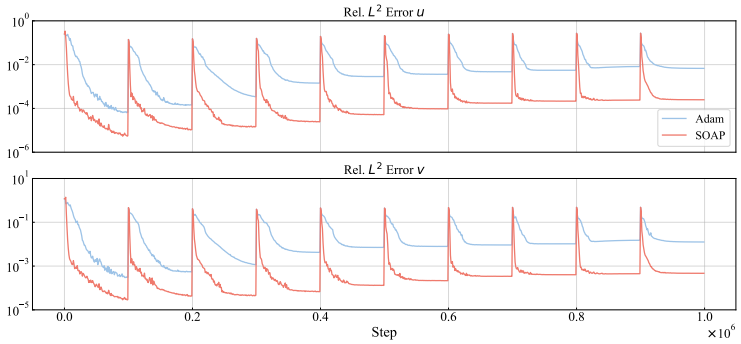


Figure 12: *Grey-Scott equation.* Test error trajectories for the Adam and SOAP optimizers.

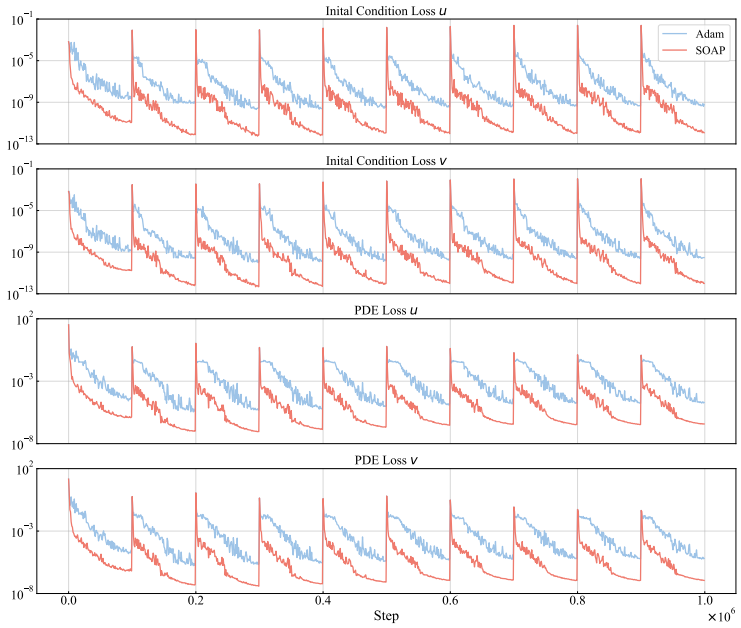


Figure 13: *Grey-Scott equation.* Training loss and test error trajectories for the Adam and SOAP optimizers.

**Ginzburg-Landau equation.** The complex Ginzburg-Landau equation in 2D takes the form

$$\frac{\partial A}{\partial t} = \epsilon \Delta A + \mu A - \gamma A|A|^2\,, \quad t \in (0,1)\,,\ (x,y) \in (-1,1)^2\,,$$

with periodic boundary conditions, an initial condition

$$A_0(x,y) = (10y + 10ix)\exp\left(-0.01(2500x^2 + 2500y^2)\right)\,,$$

where $A$ is the complex amplitude representing the envelope of oscillations, $\epsilon$ represents the diffusion coefficient, $\mu$ is the linear growth rate, and $\gamma$ controls the nonlinear saturation. For this example, we set $\epsilon = 0.004$, $\mu = 10$ and $\gamma = 10 + 15i$.

By denoting $A = u + iv$, we can decompose the equation into real and imaginary components, resulting in the following system of PDEs,

$$\frac{\partial u}{\partial t} = \epsilon \Delta u + \mu(u - (u - 1.5v)(u^2 + v^2))\,,$$
$$\frac{\partial v}{\partial t} = \epsilon \Delta v + \mu(v - (v + 1.5u)(u^2 + v^2))\,.$$

Given the coupled dynamics of $u$ and $v$, we present the relative $L^2$ error of $u$ in Table 3.

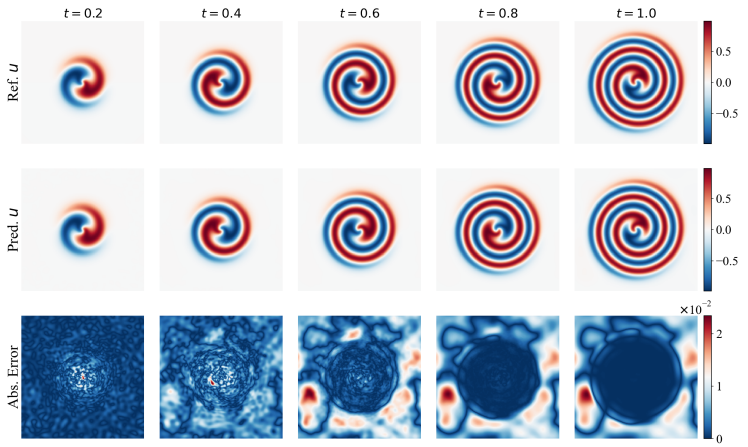


Figure 14: *Ginzburg-Landau equation.* Comparison between the reference solution and model predictions.

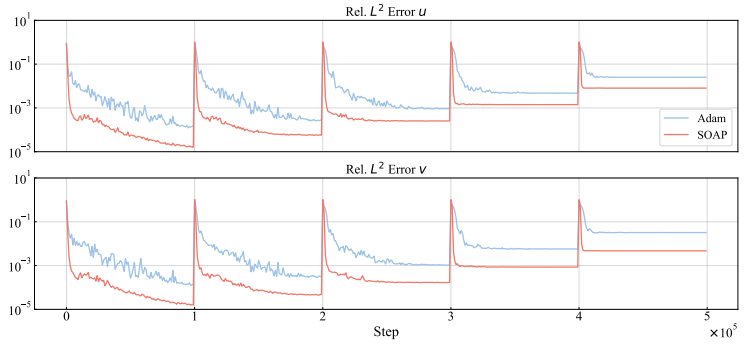


Figure 15: *Ginzburg-Landau equation.* Test error trajectories for the Adam and SOAP optimizers.

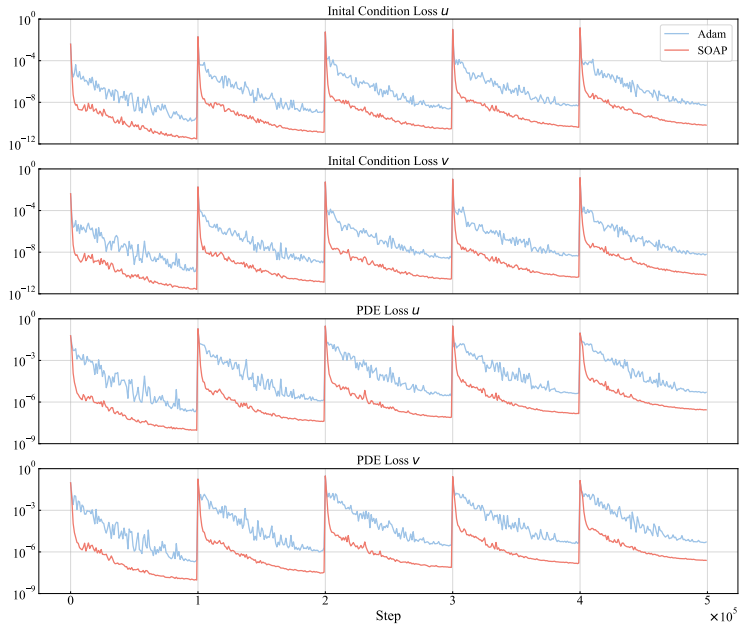


Figure 16: *Ginzburg-Landau equation.* Training loss trajectories for the Adam and SOAP optimizers.

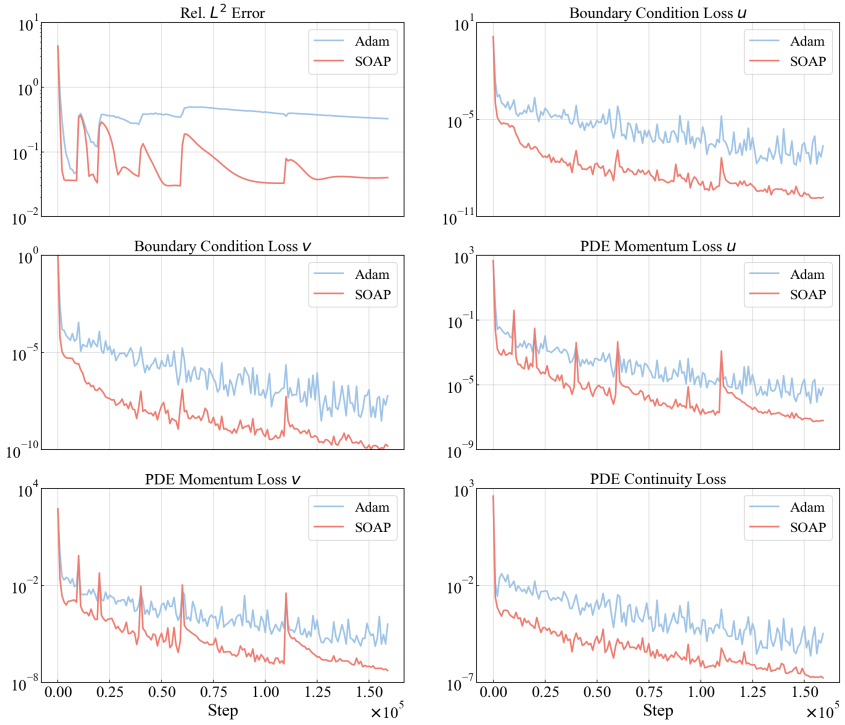


Figure 17: *Lid-driven Cavity.* Training loss and test error trajectories for the Adam and SOAP optimizers.

**Lid-driven Cavity.** We study the incompressible Navier-Stokes equations in non-dimensional form for a two-dimensional domain:

$$\begin{aligned} \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - \frac{1}{Re} \Delta \mathbf{u} = 0\,, \quad (x, y) \in (0, 1)^2\,, \\ \nabla \cdot \mathbf{u} = 0\,, \quad (x, y) \in (0, 1)^2\,, \end{aligned}$$

where $\mathbf{u} = (u, v)$ represents the steady-state velocity field, $p$ is the pressure field, and $Re$ is the Reynolds number which characterizes the ratio of inertial to viscous forces. This system models the equilibrium state of the flow, which is driven by the top boundary moving at a constant velocity while the other walls are stationary, leading to the formation of characteristic vortical structures whose complexity increases with the Reynolds number.

To ensure continuity at the corner boundaries, we implement a smoothed top-lid boundary condition:

$$u(x, y) = 1 - \frac{\cosh\left(C_0(x - 0.5)\right)}{\cosh\left(0.5 C_0\right)}\,, \quad v(x, y) = 0\,, \tag{G.29}$$

where $x \in [0, 1], y = 1, C_0 = 50$. For the other three walls, we enforce a no-slip boundary condition. Our goal is to obtain the velocity and pressure field corresponding to a Reynolds number of $5,000$. The accuracy of our method is evaluated using the velocity magnitude $\sqrt{u^2 + v^2}$, with results presented in Table 3.

**Kolmogorov flow.** We study the two-dimensional Kolmogorov flow governed by the incompressible Navier-Stokes equations:

$$\begin{aligned} \mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re} \Delta \mathbf{u} + \mathbf{f}, \\ \nabla \cdot \mathbf{u} = 0, \end{aligned}$$

on the unit square domain $(x, y) \in [0, 1]^2$.

Here $\mathbf{u} = (u, v)$ represents the time-varying velocity field, and $\mathbf{f}$ denotes the external forcing term that maintains the flow structure. The system evolves from a random initial state and develops

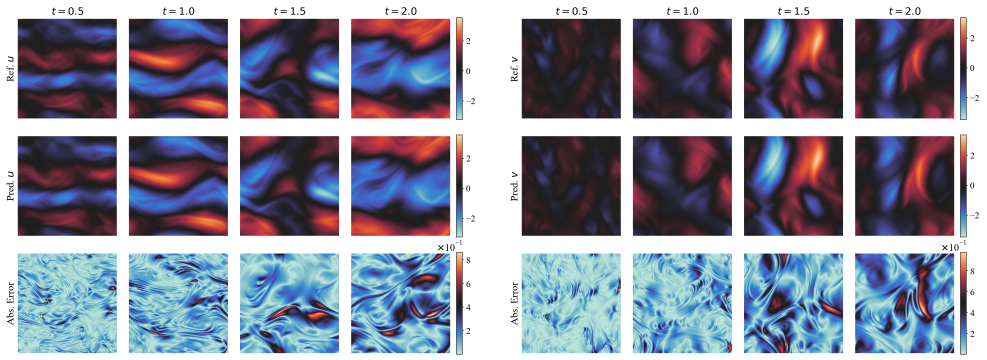


Figure 18: *Kolmogorov flow.* Comparison between reference solution and model predictions.

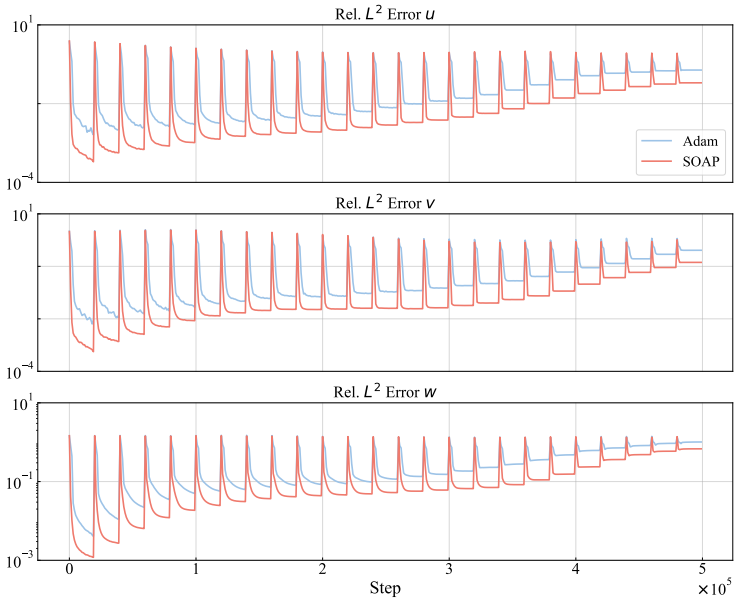


Figure 19: *Kolmogorov flow.* Test error trajectories for the Adam and SOAP optimizers.

characteristic flow patterns, where energy transfers between different spatial scales through nonlinear interactions and viscous dissipation.

For our study, the system is driven by a sinusoidal forcing $\mathbf{f} = (2\sin(4\pi y), 0)$. The numerical experiment initializes with a random initial condition and evolves until $T = 2$. The model's performance is quantified by the relative $L^2$ error of vorticity (Table 3).

**Rayleigh-Taylor instability.** We investigate a coupled flow-temperature system that models buoyancy-driven instability in a rectangular domain $(x, y) \in [0, 1] \times [0, 2]$:

$$\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \sqrt{\frac{Pr}{Ra}} \Delta \mathbf{u} + T\mathbf{e}_y, \tag{G.30}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{G.31}$$

$$T_t + \nabla \cdot (\mathbf{u}T) = \frac{1}{\sqrt{PrRa}} T_{tt} \tag{G.32}$$

where $T$ is the temperature field (acting as a density proxy through the Boussinesq approximation), Pr is the Prandtl number (ratio of momentum to thermal diffusivity), and Ra is the Rayleigh number (measuring buoyancy-driven flow strength). This system captures the characteristic mushroom-shaped plumes that develop as the heavier fluid penetrates into the lighter fluid below.

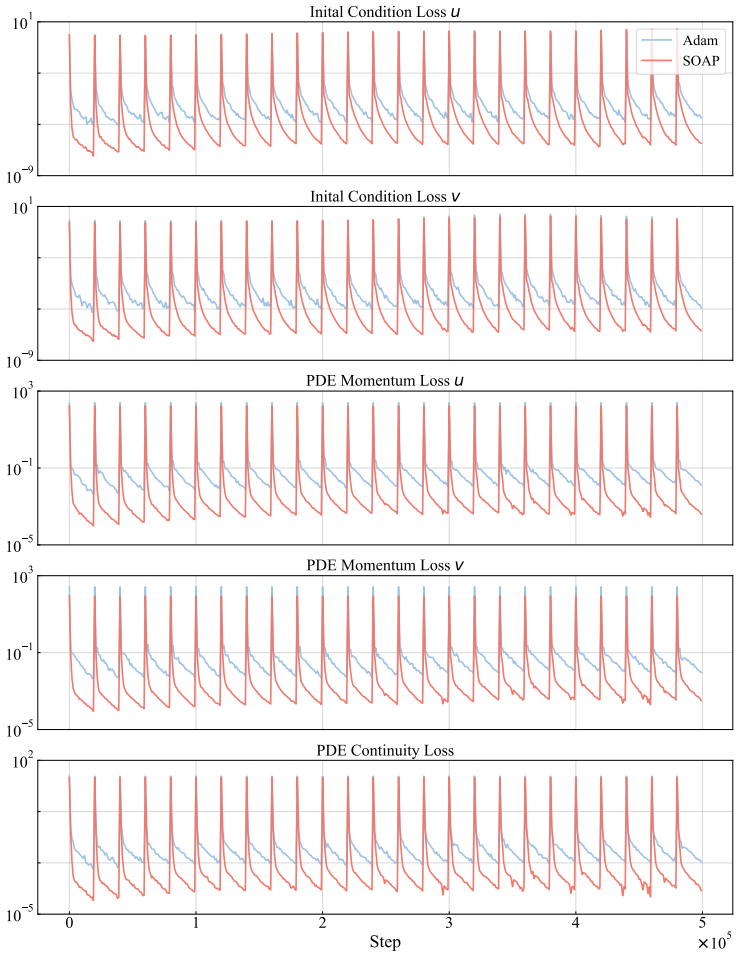


Figure 20: *Kolmogorov flow.* Training loss trajectories for the Adam and SOAP optimizers.

We set the Prandtl number $\mathrm{Pr} = 0.71$ and Rayleigh number $\mathrm{Ra} = 10^6$. The boundary conditions are periodic in the horizontal direction for both $\mathbf{u}$ and $T$, with Dirichlet conditions $\mathbf{u} = T = 0$ imposed on the top and bottom boundaries. The accuracy of our method is evaluated using the temperature field, with results presented in Table 3.

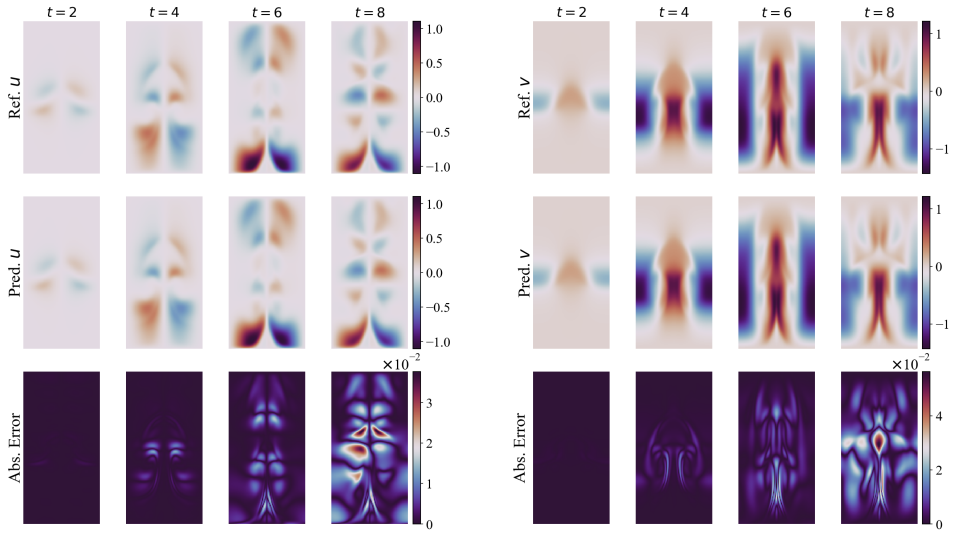


Figure 21: *Rayleigh-Taylor instability.* Comparison between reference solution and model predictions.

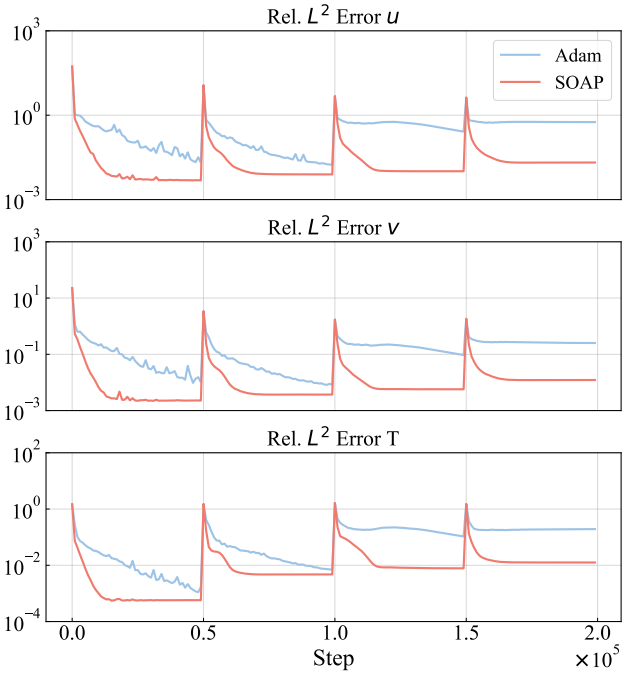


Figure 22: *Rayleigh-Taylor instability.* Test error trajectories for the Adam and SOAP optimizers.

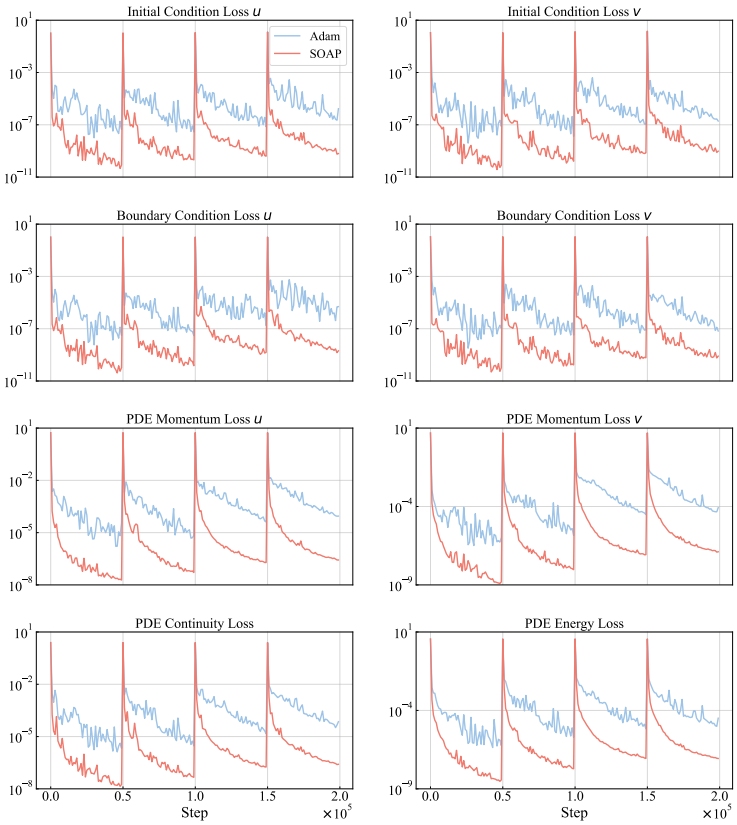


Figure 23: *Rayleigh-Taylor instability.* Training loss trajectories for the Adam and SOAP optimizers.