Self-Monitoring Warmup: Data-Driven Confidence Rectification for PINN Optimizers

Giangiacomo Mercatali [1] **Jie Chen** [2] **Alexandros Kalousis** [1]

Abstract

Adaptive optimizers such as Adam, RAdam, and SOAP decide how aggressively to trust their adaptive learning rate from a fixed rule: a closed-form factor, an iteration-count schedule, or a distributional assumption on gradients. In physics-informed neural networks (PINNs), where gradients couple multi-scale IC, BC, and PDE-residual terms through differential operators, these assumptions fit poorly with the observed training dynamics: gradients are heterogeneous across parameters, and Adam- and SOAP-style updates exhibit early-training gradient distortion. We argue that the optimizer should set its rectification from the empirical statistics of its own second-moment estimates, rather than from the iteration count, and propose *Adaptive Confidence Rectification* (ACR): a data-driven warmup in which each parameter's rectification is set by the empirical volatility of its second-moment estimate, turned into a bounded per-parameter confidence. ACR makes no assumption on the gradient distribution and acts independently on each parameter. We show ACR keeps the variance its rectification adds to the effective learning rate bounded—vanishing as confidence rises—and strictly contracts the step under low confidence. The method is general and can be combined with any adaptive optimizer; we demonstrate it on SOAP, Sophia, and AdaHessian, and in the first-order setting against RAdam, where it consistently improves performance across four PDEs (Allen–Cahn, Wave, KdV, Burgers) and three model scales. Code is available at `https://github.com/gmerca/warmup_pinn`.

[1]HES-SO Geneva, Switzerland [2]MIT-IBM Watson AI Lab, USA. Correspondence to: <giangiacomo.mercatali@hesge.ch>.

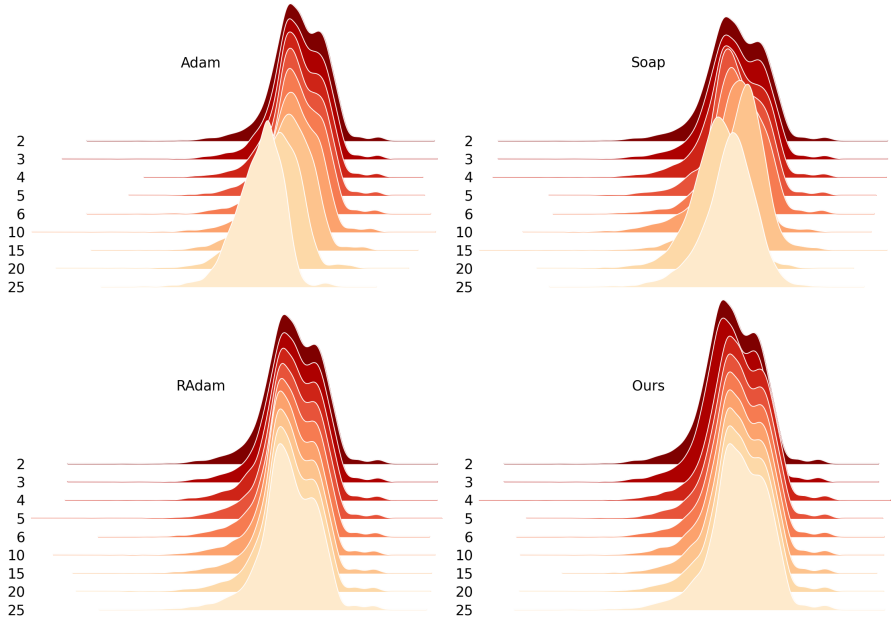
Proceedings of the AI4Physics Workshop at the 43rd International Conference on Machine Learning (AI4Physics@ICML 2026), Seoul, South Korea. 2026. Copyright 2026 by the author(s).

1. Introduction

Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019) have become the dominant computational tool for solving forward and inverse PDE problems across fluid mechanics (Cai et al., 2021; Mao et al., 2020), power systems (Misyris et al., 2020), and broader scientific modelling (Coscia et al., 2023), yet their training is notoriously difficult from an optimisation standpoint (Wang et al., 2023; Bonfanti et al., 2024; Rathore et al., 2024). Two well-documented sources of difficulty are the ill-conditioning that arises from coupling PDE-defined and backpropagation-defined derivatives (Wang et al., 2022; Ryck et al., 2024; Müller & Zeinhofer, 2024), and the multi-task structure of the loss, which combines initial-condition (IC), boundary-condition (BC), and PDE-residual terms whose gradient scales are typically misaligned by orders of magnitude (Wang et al., 2021; 2024; 2023). Existing remedies act on the loss landscape itself — by rebalancing the loss terms (Wang et al., 2021; 2022), by resolving gradient conflicts across them (Liu et al., 2024b; Hwang & Lim, 2024; Yao et al., 2023), by introducing curvature information (Wang et al., 2025; Hao et al., 2024; Jnini et al., 2024), or by optimising in function space (Müller & Zeinhofer, 2023; Schwencke & Furtlehner, 2025; Müller & Zeinhofer, 2024). A complementary, largely unstudied source of difficulty is the early-training instability of the *adaptive optimizer state* that drives all these methods, which is exactly what learning-rate warmup is designed to address.

Warmup as an under-studied lever in PINNs. Learning-rate warmup (Liu et al., 2020; Alimisis et al., 2025) stabilises adaptive optimisers such as Adam (Kingma & Ba, 2015) during the first iterations, when the second-moment estimate is built from too few samples and produces high-variance updates (Liu et al., 2020). While warmup has been extensively studied in supervised learning (Liu et al., 2025; Alimisis et al., 2025; Kalra & Barkeshli, 2024), particularly for transformers (Kosson et al., 2024), its role in PINN training has not been systematically investigated. We focus on the most principled instance of this family, RAdam (Liu et al., 2020), which derives an analytical warmup factor by rescaling the adaptive learning rate toward its asymptotic variance, and which has been broadly successful in other

Figure 1. Histograms of log-scale absolute gradients over the first 25 training iterations of a PINN solving the Navier–Stokes PDE, under four optimisers (top-left: Adam; top-right: SOAP; bottom-left: RAdam; bottom-right: ACR). Adam and SOAP rapidly distort the gradient distribution, collapsing the multimodal structure into a unimodal one; RAdam and ACR preserve it.



ML settings (Song et al., 2023; Luo et al., 2023; He et al., 2020; Katharopoulos et al., 2020; Yasunaga et al., 2021).

Warmup matters in PINNs. Figure 1 shows how the per-parameter gradient distribution evolves over the first 25 iterations of training a PINN on the Navier–Stokes equation under four optimisers. Both Adam (Kingma & Ba, 2015) and SOAP (Vyas et al., 2025) — the latter recently reported as state of the art for PINNs (Wang et al., 2025) — rapidly distort the gradient distribution, collapsing a multimodal shape into a unimodal one within the first 25 steps; in contrast, RAdam and ACR preserve the original structure. Gradient distortion of this kind was first observed by Liu et al. (2020) but its consequences for PINN training dynamics have not been studied. This motivates the rest of the paper: a careful look at RAdam in the PINN regime, and a method that fixes its main limitations there.

RAdam approach to warmup. Figure 1 motivates us to further study, understand, and improve RAdam in the context of PINNs. At a high level, RAdam treats the early phase of training as a low-evidence regime: with only a handful of gradient samples seen so far, the running estimate of the second moment is unreliable, which inflates the variance of the adaptive learning rate and destabilises updates. RAdam quantifies how much evidence the optimiser has accumulated and uses this quantity to shrink the adaptive step. When the evidence is too low, it falls back to plain momentum updates; as more gradients are observed, it gradually trusts the adaptive rule and recovers Adam-like behaviour. The effect is a built-in, hyperparameter-free warmup that replaces the linear schedules used in practice.

Limitations of RAdam. This recipe rests on a strong assumption: that gradients are well behaved, roughly symmetric and Gaussian, so the amount of evidence available can be inferred purely from how many steps have elapsed. In PINNs this assumption is routinely violated. Because IC, BC, and PDE-residual losses live on different scales and are composed with PDE differential operators, the resulting gradients are heavy-tailed, skewed, or multi-modal, and their reliability varies sharply across layers and parameters. RAdam is blind to this heterogeneity in two important ways. First, its rectification factor is a single scalar applied uniformly to every parameter, even though some coordinates have already produced stable curvature estimates while others remain noisy. Second, it depends only on the iteration count, so it cannot react to the actual variability of the second-moment estimates it is meant to correct, and the moment at which it transitions out of warmup is fixed in advance.

Our method. We propose *Adaptive Confidence Rectification* (ACR), illustrated in Figure 2, which replaces RAdam's iteration-based, distribution-dependent rule with an empirical, parameter-wise measure of how trustworthy each second-moment estimate currently is. ACR tracks how much each coordinate's curvature estimate fluctuates over recent steps and turns this into a confidence signal in $[0, 1]$: high-confidence coordinates recover RAdam's coefficient,

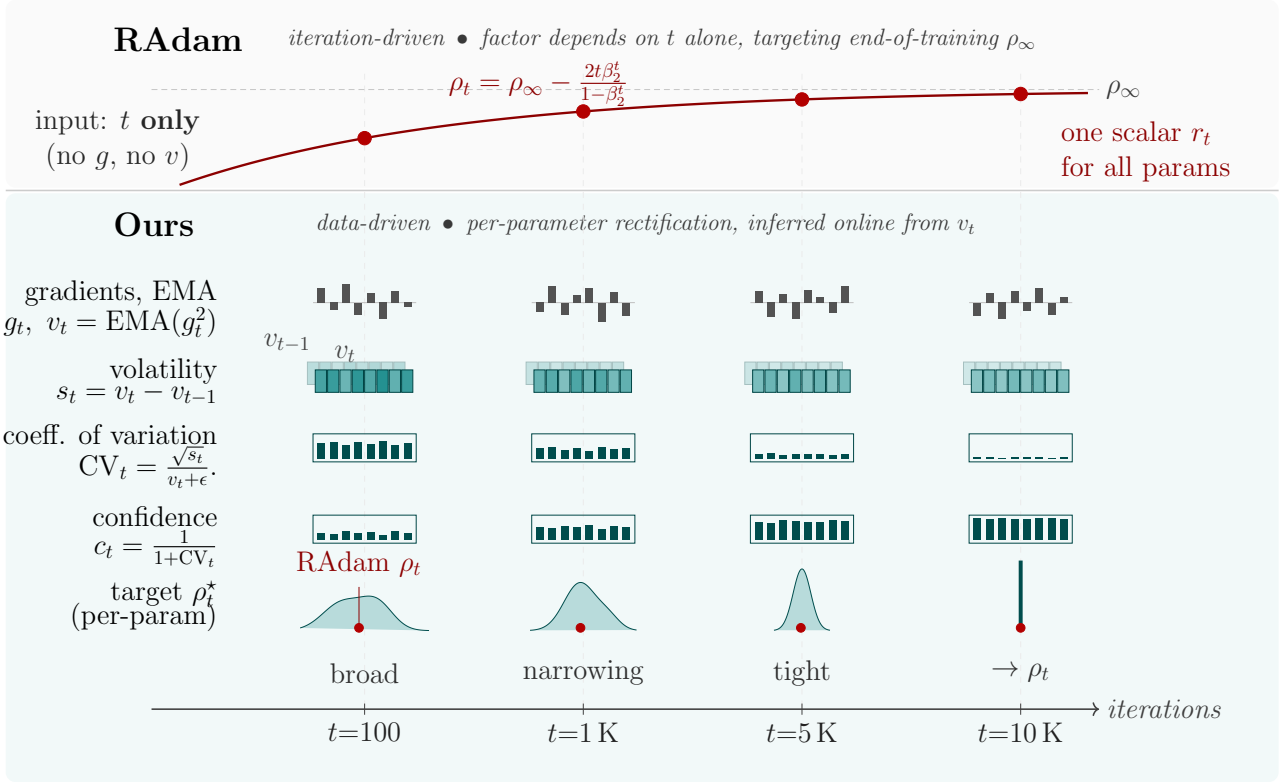


Figure 2. Overview of ACR compared to RAdam. RAdam (top) applies a single, time-dependent rectification factor derived from a Gaussian assumption on gradients: every parameter sees the same correction at a given step, and the warmup horizon is fixed in advance. ACR (bottom) reads, for each coordinate, the recent fluctuations of the second-moment estimate v and turns them into a coefficient of variation CV and a confidence $\mathrm{conf} \in [0, 1]$. The confidence interpolates the rectification between RAdam's coefficient (when the estimate is stable) and a conservative anchor that damps the adaptive step (when the estimate is noisy), so the correction adapts both per-parameter and over training time.

while low-confidence ones are pulled toward a conservative anchor that damps the adaptive step. Consequently, in early training the confidence is uniformly low and ACR behaves like a strong, automatic warmup; as the second-moment estimates settle, confidence rises and ACR smoothly recovers RAdam's asymptotic rule. The rectification therefore adapts both across parameters and across training stages, without invoking any assumption on the shape of the gradient distribution. The full mechanism is detailed in Section 4. Empirically, the per-parameter confidence and the conservative anchor are non-redundant by design: the data-driven per-parameter signal supplies the gain over RAdam, but without an anchor to bound it the same signal actively destabilises training. Only the anchored signal yields the lowest error with low run-to-run variance (Table 4), so ACR's central mechanism is anchor-gated per-parameter adaptation rather than either part alone.

Our contributions are threefold:

Methodological: We propose *Adaptive Confidence Rectification* (ACR), a general framework for dynamically correcting second-moment gradient estimates based on an empirical measure of statistical reliability. By explicitly accounting for uncertainty in variance estimation, ACR enables principled, data-driven modulation of adaptive learning rates (Section 4). We instantiate this framework for several state-of-the-art adaptive optimizers.

Theoretical: We present the first formal analysis of confidence-based rectification in adaptive optimization. Theorem 4.1 bounds the variance that ACR's confidence-driven rectification adds to the effective learning rate—a bound that vanishes as confidence rises—while Proposition 4.2 establishes local contraction toward a conservative regime under uncertainty. Together, these results characterize ACR as an automatic variance controller that stabilizes training while preserving asymptotic efficiency.

Empirical: We perform a comprehensive evaluation of ACR on a broad suite of PDE-based physics-informed neural network (PINN) benchmarks, including Allen–Cahn, Wave, Korteweg–de Vries, Burgers equations, across multiple architectures, model scales, and adaptive optimizers. Across all experimental settings, ACR consistently yields substantial and robust performance improvements over RAdam and non-rectified baselines.

2. Related Works

RAdam. RAdam (Liu et al., 2020) traces the early-training instability of Adam (Kingma & Ba, 2015) to the high variance of its adaptive learning rate under a small effective sample size, and corrects it with a closed-form factor that pulls the adaptive step toward its asymptotic variance, disabling the adaptive term and reverting to momentum-only updates while the variance is intractable. This yields a principled, hyperparameter-free warmup, but the correction is derived under i.i.d. Gaussian gradients and applied as a single, time-dependent scalar to all parameters. Our method, ACR, retains this reliability-aware view while replacing the rule with an empirical, parameter-wise estimate of second-moment stability, giving more flexible adaptation across training regimes.

Optimization methods for PINNs. Some works leverage the multi-loss aspect of PINNs: Yao et al. (2023) suggest applying separate Adam optimizers to each component of the PINN loss function, such as the boundary and initial conditions. Liu et al. (2024b) leverage multitask learning to address gradient imbalances between initial/boundary conditions and data terms. Hwang & Lim (2024) propose constraining gradients to lie within a dual cone region to improve training dynamics. Some works focus on second-order methods for PINNs: Rathore et al. (2024) propose a sequential optimization strategy for PINNs, combining Adam, L-BFGS, and a Nyström-accelerated Newton conjugate gradient method (NNCG). Hao et al. (2024) introduce a Gauss-Newton method tailored for solving PDEs with neural networks. Müller & Zeinhofer (2023; 2024) propose the use of the Energy Natural Gradient (ENG) method that is equivalent to a newton method in function space. Dangel et al. (2024) adapt the ENG method with a layerwise structure. Schwencke & Furtlehner (2025) explore optimization in the function space induced by the neural tangent kernel (NTK). Jnini & Vella (2024); Jnini et al. (2024) develop Hessian-free and Gauss–Newton methods for PINNs. Differently, we explore a parallel direction of investigating a principled method to warm up the LR, which can be applied to any optimization method. We provide an extended discussion of related work in Appendix G.

3. Background

PINN Setup. The PINN training framework (Raissi et al., 2019) considers a general PDE $\mathbf{u}_t + \mathcal{N}[\mathbf{u}] = 0$ on $t \in [0, T]$, $\mathbf{x} \in \Omega$, subject to the initial condition $\mathbf{u}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x})$ and boundary condition $\mathcal{B}[\mathbf{u}] = 0$ on $\partial\Omega$. The solution is approximated by a neural network $\mathbf{u}_\theta(t, \mathbf{x})$ trained on residual points $\{(t_r^i, \mathbf{x}_r^i)\}_{i=1}^{N_r}$, and the resulting stochastic gradient mixes three competing terms:

$$\begin{cases} \mathcal{L}_{\text{IC}}(\theta) = \frac{1}{N_0} \sum_{i=1}^{N_0} \left| \mathbf{u}_\theta(0, \mathbf{x}_0^i) - \mathbf{g}(\mathbf{x}_0^i) \right|^2 \\ \mathcal{L}_{\text{BC}}(\theta) = \frac{1}{N_b} \sum_{i=1}^{N_b} \left| \mathcal{B}[\mathbf{u}_\theta](t_b^i, \mathbf{x}_b^i) \right|^2 \\ \mathcal{L}_{\text{PDE}}(\theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{R}[\mathbf{u}_\theta](t_r^i, \mathbf{x}_r^i) \right|^2 \\ \mathbf{g} = \nabla_\theta [\lambda_1 \mathcal{L}_{\text{IC}} + \lambda_2 \mathcal{L}_{\text{BC}} + \lambda_3 \mathcal{L}_{\text{PDE}}] \,. \end{cases} \tag{PINN}$$

PINN training involves multiple competing loss terms and heterogeneous gradient scales, leading to noisy and poorly conditioned updates, especially in early training. In this paper we explore ideas to fix these issues via LR warmup methods, including RAdam, that we overview next.

RAdam. Liu et al. (2020) identify a critical limitation of the Adam optimizer during the initial training phase: the second-moment estimate $\mathbf{v}^{(t)} = \beta_2 \mathbf{v}^{(t-1)} + (1 - \beta_2) \mathbf{g}^{(t)2}$ suffers from high variance, which hampers convergence. RAdam introduces a variance rectification mechanism that stabilizes this adaptive term, leading to improved convergence behavior and greater robustness during initialization:

$$\begin{cases} \mathbf{m}^{(t)} = \beta_1 \mathbf{m}^{(t-1)} + (1 - \beta_1)\, \mathbf{g}^{(t)}, \\ \hat{\mathbf{m}}^{(t)} = \mathbf{m}^{(t)} / (1 - \beta_1^t) \\ \mathbf{v}^{(t)} = \beta_2 \mathbf{v}^{(t-1)} + (1 - \beta_2)\, \mathbf{g}^{(t)2}, \\ \hat{\mathbf{v}}^{(t)} = \mathbf{v}^{(t)} / (1 - \beta_2^t) \\ \rho_t = \rho_\infty - 2t\beta_2^t / (1 - \beta_2^t) \\ r_t = \sqrt{\dfrac{(\rho_t - 4)(\rho_t - 2)\rho_\infty}{(\rho_\infty - 4)(\rho_\infty - 2)\rho_t}} \\ \boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta \left(r_t \dfrac{\hat{\mathbf{m}}^{(t)}}{\sqrt{\hat{\mathbf{v}}^{(t)}} + \epsilon} + \lambda\, \boldsymbol{\theta}^{(t-1)} \right). \end{cases} \tag{RAdam}$$

By interpolating between SGD and Adam, RAdam improves early-stage stability. However, its rectification relies on distributional assumptions and a fixed variance target, which may not align well with the evolving statistical properties encountered during training. These considerations motivate a data-driven rectification mechanism that directly reflects the empirical reliability of second-moment estimates. In the next section, we introduce Adaptive Confidence Rectification, which generalizes RAdam by dynamically adjusting variance correction based on observed statistical stability.

4. Method

4.1. Adaptive Confidence Rectification (ACR).

We present ACR, an optimiser that targets the high-variance second-moment regime which destabilises PINN training in its early phase. Our method replaces RAdam's fixed, time-only rectification target ρ_t with a per-parameter, data-driven target computed from the empirical volatility of the second-moment estimate $\mathbf{v}^{(t)}$ itself. This removes the two

design constraints inherited from RAdam — a single scalar applied to every parameter, and a schedule that depends only on t — while recovering RAdam exactly once the optimiser stabilises. Concretely, ACR tracks per-parameter how much $\mathbf{v}^{(t)}$ fluctuates, converts that into a bounded confidence $\mathrm{conf}_t \in [0,1]$, and uses it to interpolate between RAdam's target ρ_t and a conservative anchor ρ_{safe} (Figure 2).

We re-use $\mathbf{m}^{(t)}, \hat{\mathbf{m}}^{(t)}, \mathbf{v}^{(t)}, \hat{\mathbf{v}}^{(t)}, \rho_t, \rho_\infty$ from (RAdam); the new ACR components are

$$\begin{cases} \mathbf{s}^{(t)} = \beta_s\, \mathbf{s}^{(t-1)} + (1-\beta_s)\big(\mathbf{v}^{(t)} - \mathbf{v}^{(t-1)}\big)^2 \\ CV_t = \sqrt{\mathbf{s}^{(t)}}/(\hat{\mathbf{v}}^{(t)} + \epsilon), \quad \mathrm{conf}_t = 1/(1+CV_t) \\ \rho_{\mathrm{target}} = \mathrm{conf}_t\, \rho_t + (1-\mathrm{conf}_t)\, \rho_{\mathrm{safe}} \\ r(\rho_{\mathrm{target}}) = \sqrt{\dfrac{(\rho_{\mathrm{target}}-4)(\rho_{\mathrm{target}}-2)\,\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\,\rho_{\mathrm{target}}}} \\ \eta_t^{\mathrm{ACR}} = \alpha_t\, r(\rho_{\mathrm{target}})/(\sqrt{\hat{\mathbf{v}}^{(t)}} + \epsilon). \end{cases} \tag{ACR}$$

$\mathbf{s}^{(t)}$ is an EMA of the per-step change in $\mathbf{v}^{(t)}$; CV_t and conf_t turn that volatility into a bounded reliability score; ρ_{target} is a per-parameter convex combination, and η_t^{ACR} is the resulting effective learning rate (here $\alpha_t \equiv \eta$, the global learning rate used in Algorithm 1). As $\mathrm{conf}_t \to 1$, $\rho_{\mathrm{target}} \to \rho_t$ and ACR recovers RAdam; as $\mathrm{conf}_t \to 0$, ρ_{target} anchors to ρ_{safe} and the step is damped. The full per-step procedure is given in Algorithm 1.

4.2. Analysis of our method.

Theorem 4.1 (ACR Rectification-Variance Bound)**.** *Under standard regularity assumptions (detailed in Appendix B.1), the variance that the confidence-driven rectification contributes to the ACR effective learning rate η_t^{ACR} is bounded by*

$$\mathrm{Var}[\eta_t^{\mathrm{ACR}}] \le \Delta_t,$$
$$\Delta_t = \frac{\alpha_t^2 L_r^2}{(\sqrt{\hat{\mathbf{v}}^{(t)}} + \epsilon)^2}\,(\rho_t - \rho_{\mathrm{safe}})^2\,\mathrm{Var}[\mathrm{conf}_t], \tag{1}$$

where L_r is the Lipschitz constant of the rectification function $r(\cdot)$. The bound satisfies $\Delta_t \ge 0$ and $\Delta_t \to 0$ as $\mathrm{conf}_t \to 1$, recovering RAdam; moreover, whenever $\rho_t > \rho_{\mathrm{safe}}$, the ACR step is pointwise no larger than RAdam's (Proposition 4.2).

Proof sketch. Writing $\eta_t^{\mathrm{ACR}} = \eta_0\, r(\rho_{\mathrm{target}})$ with $\eta_0 = \alpha_t/(\sqrt{\hat{\mathbf{v}}^{(t)}}+\epsilon)$, the Lipschitz continuity of r and the identity $\rho_{\mathrm{target}} - \rho_t = (1-\mathrm{conf}_t)(\rho_{\mathrm{safe}} - \rho_t)$ give $\mathrm{Var}[r(\rho_{\mathrm{target}})] \le L_r^2(\rho_t - \rho_{\mathrm{safe}})^2\,\mathrm{Var}[\mathrm{conf}_t]$, hence $\mathrm{Var}[\eta_t^{\mathrm{ACR}}] \le \Delta_t$, which vanishes as $\mathrm{conf}_t \to 1$. The full proof is in Appendix B.1.

The bound Δ_t is large precisely when the second-moment estimate is unstable (so $\mathrm{Var}[\mathrm{conf}_t]$ is large) and vanishes once conf_t saturates near 1, recovering RAdam asymptotically. ACR therefore acts as an implicit variance controller that is active during the early-training regime and inactive once the optimiser settles.

The same construction also provides a strict step-size guarantee under low confidence.

Proposition 4.2 (Local Stability via Confidence Contraction)**.** *For any iteration t with $\rho_t > \rho_{\mathrm{safe}}$, the ACR rectification coefficient satisfies*

$$r(\rho_{\mathrm{target}}) \le r(\rho_t) - \mathcal{C}\,(1 - \mathrm{conf}_t), \tag{2}$$

for some $\mathcal{C} > 0$ depending on the local curvature of $r(\cdot)$.

Proof sketch. $r(\rho)$ is monotonically increasing on $[4,\infty)$, and ρ_{target} is a convex combination $\mathrm{conf}_t\rho_t + (1-\mathrm{conf}_t)\rho_{\mathrm{safe}}$ with $\rho_t > \rho_{\mathrm{safe}}$, so $\rho_{\mathrm{target}} \le \rho_t$ and $r(\rho_{\mathrm{target}}) \le r(\rho_t)$. Linearising r around ρ_t gives a reduction proportional to $(1-\mathrm{conf}_t)(\rho_t - \rho_{\mathrm{safe}})$. Full proof in Appendix B.2.

In words, whenever the second-moment estimate is uncertain ($\mathrm{conf}_t < 1$), the effective step is strictly smaller than RAdam's, preventing the optimiser from acting on unreliable curvature; when $\mathrm{conf}_t \to 1$, both methods coincide.

4.3. Algorithm.

Algorithm 1 summarizes the per-step procedure: lines 1–5 are standard RAdam, lines 6–9 are the ACR additions (blue) that turn the volatility of $\mathbf{v}^{(t)}$ into a confidence and an adaptive target ρ_{target}, and lines 10–11 compute the rectified coefficient and apply the update. The same mechanism plugs into other adaptive optimizers; pseudocode for SOAP, Sophia, AdaHessian, and the SOAP+RAdam baseline is in Appendix F.

Algorithm 1 Adaptive Confidence Rectification (ACR). Differences with RAdam are in blue.

1: $\mathbf{m}^{(t)} \leftarrow \beta_1\, \mathbf{m}^{(t-1)} + (1-\beta_1)\, \mathbf{g}^{(t)}$,
2: $\hat{\mathbf{m}}^{(t)} \leftarrow \mathbf{m}^{(t)}/(1-\beta_1^t)$
3: $\mathbf{v}^{(t)} \leftarrow \beta_2\, \mathbf{v}^{(t-1)} + (1-\beta_2)\, \mathbf{g}^{(t)2}$,
4: $\hat{\mathbf{v}}^{(t)} \leftarrow \mathbf{v}^{(t)}/(1-\beta_2^t)$
5: $\rho_t \leftarrow \rho_\infty - 2t\beta_2^t/(1-\beta_2^t), \quad \rho_\infty = 2/(1-\beta_2) - 1$
6: $\mathbf{s}^{(t)} \leftarrow \beta_s\, \mathbf{s}^{(t-1)} + (1-\beta_s)(\mathbf{v}^{(t)} - \mathbf{v}^{(t-1)})^2$
7: $CV_t \leftarrow \sqrt{\mathbf{s}^{(t)}}/(\hat{\mathbf{v}}^{(t)} + \epsilon)$
8: $\mathrm{conf}_t \leftarrow 1/(1+CV_t) \in [0,1]$
9: $\rho_{\mathrm{target}} \leftarrow \mathrm{conf}_t\, \rho_t + (1-\mathrm{conf}_t)\, \rho_{\mathrm{safe}}$
10: $r_t^{\mathrm{ACR}} \leftarrow \sqrt{\dfrac{(\rho_{\mathrm{target}}-4)(\rho_{\mathrm{target}}-2)\,\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho_{\mathrm{target}}}}$
11: $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} - \eta\left(r_t^{\mathrm{ACR}}\, \dfrac{\hat{\mathbf{m}}^{(t)}}{\sqrt{\hat{\mathbf{v}}^{(t)}} + \epsilon} + \lambda_{\mathrm{wd}}\, \boldsymbol{\theta}^{(t-1)}\right)$

Table 1. L2 results over multiple model sizes with MLP architecture.

Dataset	70K parameters		135K parameters		200K parameters	
	SOAP+RAdam	Ours	SOAP+RAdam	Ours	SOAP+RAdam	Ours
Allen Cahn ($\times 10^{-6}$)	41.0±8.6	22.1±3.7	5.46±0.99	4.79±0.65	5.27±1.32	3.48±0.05
Wave ($\times 10^{-6}$)	4.70±0.60	2.71±0.21	3.40±0.26	2.65±0.13	3.66±0.21	2.56±0.24
KdV ($\times 10^{-4}$)	5.32±0.53	4.08±0.25	4.29±0.18	3.28±0.01	3.48±0.16	3.38±0.26
Burgers ($\times 10^{-5}$)	8.55±0.49	5.96±0.66	6.60±0.47	4.55±0.24	4.90±0.03	4.11±0.02

Table 2. L^2 error across SOAP, AdaHessian, and Sophia, comparing ACR (Ours) against two baselines per optimizer: the non-rectified optimizer (Non-rect) and its RAdam-rectified variant (RAdam). ACR improves on both in all cases; note that RAdam rectification is not consistently better than no rectification (e.g. AdaHessian on KdV and Burgers).

Dataset	SOAP			AdaHessian ($\times 10^{1}$)			Sophia ($\times 10^{2}$)		
	Non-rect	RAdam	Ours	Non-rect	RAdam	Ours	Non-rect	RAdam	Ours
Allen Cahn ($\times 10^{-6}$)	42.3±4.00	41.0±8.6	22.1±3.7	8.6±0.58	7.84±0.78	6.58±0.54	3.83±0.81	4.11±0.50	2.64±0.25
Wave ($\times 10^{-6}$)	4.98±0.56	4.70±0.60	2.71±0.21	4.4±0.21	4.64±0.27	3.16±0.07	7.58±1.19	8.19±0.82	3.87±1.36
KdV ($\times 10^{-4}$)	5.71±0.45	5.32±0.53	4.08±0.25	3.61±0.40	9.19±1.26	1.78±0.24	8.78±1.17	7.44±0.14	6.63±0.06
Burgers ($\times 10^{-5}$)	8.42±0.18	8.55±0.49	5.96±0.66	2.89±0.86	3.85±1.51	1.25±0.14	10.8±0.10	9.00±0.40	7.56±0.62

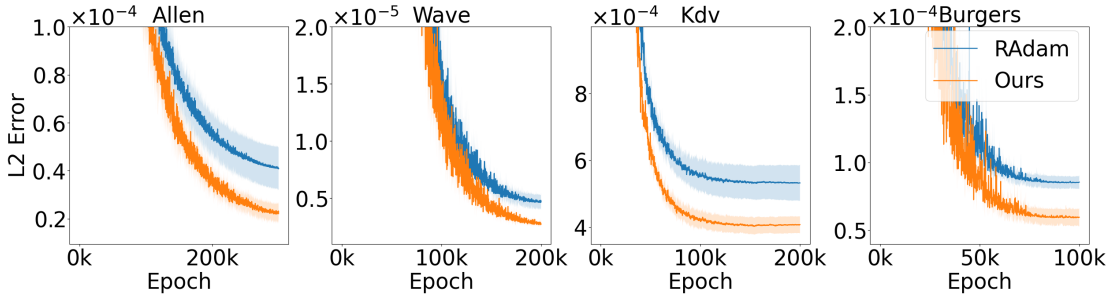


Figure 3. L^2-error trajectories over training for SOAP+RAdam (baseline) vs. SOAP+ACR on Allen–Cahn, Wave, KdV, and Burgers.

5. Experiments

We evaluate ACR on four canonical PDE benchmarks—Allen–Cahn, Wave, KdV, and Burgers—covering accuracy and convergence, hyperparameter sensitivity, and robustness across model sizes, architectures, and optimizers, with an ablation isolating each algorithmic component. Our primary baseline is RAdam's rectification applied within SOAP (Vyas et al., 2025) (the current state of the art for PINN optimization (Wang et al., 2025)). All reported numbers are mean±std over five independent seeds. Full PDE definitions and data-generation details are provided in Appendix D, and the network architectures (MLP, PirateNet, ResNet) in Appendix E.

L2 error across PDE benchmarks. Figure 3 presents the L^2 error trajectories of our method and RAdam under the SOAP framework across four PDE benchmarks. Across all settings, our approach achieves consistently lower errors and exhibits faster convergence behavior. Final L^2 errors at convergence are reported in Table 1. On all evaluated datasets, our method yields lower final errors than the baseline SOAP+RAdam. These results suggest that the proposed algorithmic modifications lead to stable and effective performance across diverse PDE problems, highlighting their practical applicability and robustness.

Comparison across multiple optimizers. We apply ACR to three adaptive methods — SOAP (Vyas et al., 2025) (Algorithm 2), AdaHessian (Yao et al., 2021) (Algorithm 4), and Sophia (Liu et al., 2024a) (Algorithm 5) — and compare it against two baselines for each: the non-rectified optimizer and its RAdam-rectified variant (Opt+RAdam). Table 2 reports the L^2 error. ACR consistently outperforms both baselines in all considered cases; notably, RAdam rectification is not consistently better than no rectification (e.g. AdaHessian on KdV and Burgers), whereas ACR improves on both. For each optimizer we swept the learning rate over the same grid used for SOAP+RAdam (Table 7) and report the per-optimizer best for each baseline; ACR variants reuse the same per-optimizer LR without further tuning.

Hyperparameter Search for the Baseline. To ensure that ACR's improvements are not attributable to suboptimal baseline tuning, we conducted controlled sweeps over the optimizer parameters β_1, β_2, and the learning rate for the base-

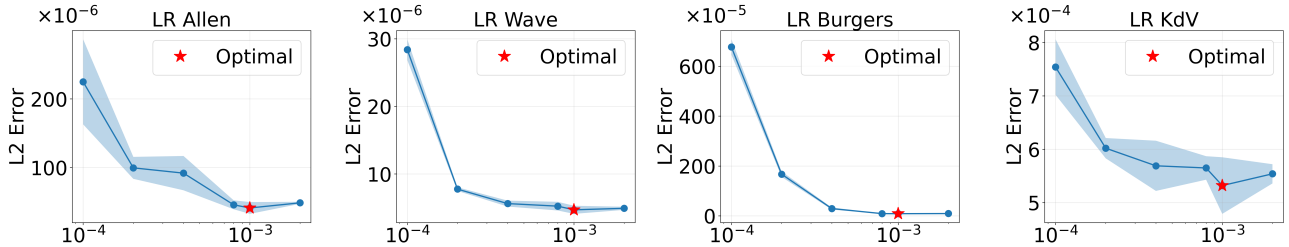


Figure 4. Baseline (SOAP+RAdam) learning-rate sweep on the Wave PDE.

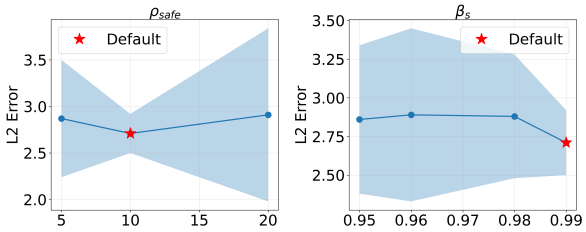


Figure 5. ACR hyperparameter sensitivity on the Wave PDE: ρ_{safe} (left), β_s (right).

Table 3. L2 error for the ACR hyperparameter sweep on the Wave PDE.

Parameter	Values	L2 error $\times 10^{-6}$
ρ_{safe}	5	2.87 ± 0.63
	10	$\mathbf{2.71 \pm 0.21}$
	20	2.91 ± 0.93
β_s	0.95	2.86 ± 0.48
	0.96	2.89 ± 0.56
	0.98	2.88 ± 0.40
	0.99	$\mathbf{2.71 \pm 0.21}$

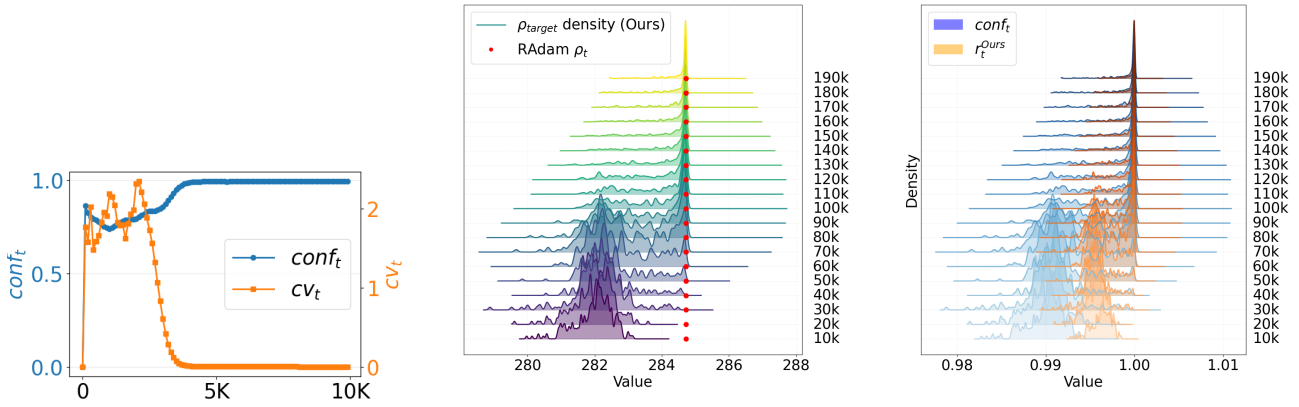
line model (SOAP+RAdam), using a fixed training budget and identical initialization. When sweeping one parameter, all others were held at their default values to isolate individual effects. The selected defaults, $\beta_1 = 0.99$ and $\beta_2 = 0.993$, achieved the strongest performance in the baseline sweep and were therefore used in all subsequent experiments without further tuning. For the learning rate, we performed a broad search spanning two orders of magnitude and including commonly used values. We observe that $LR = 0.001$ consistently yields the lowest average L^2 error across four PDE benchmarks. Accordingly, all baseline results reported in this work use this empirically optimal learning rate. Moreover, the baseline SOAP+RAdam achieves competitive state-of-the-art performance, improving the results reported by (Wang et al., 2025). These findings confirm that the baseline operates near its best achievable configuration.

Hyperparameter Sensitivity of ACR. To assess the robustness of ACR, we perform a controlled hyperparameter sweep on the Wave PDE benchmark. Each parameter is varied independently while keeping the remaining ones fixed at their default values: $\rho_{\text{safe}} = 10, \quad \beta_s = 0.99$. This experiment isolates the contribution of each hyperparameter to stability and reconstruction quality. As shown in Table 3 and visualised in Figure 5, ACR exhibits a broad optimal basin: moderate perturbations in individual parameters do not produce catastrophic degradation, indicating that the method is not overly sensitive to tuning. Figure 5 plots the L^2 error as each parameter is swept independently, with the safe fallback scale ρ_{safe} shown on the left and the variance smoothing factor β_s on the right; in both panels the error stays within a narrow band around the default configuration. Notably, the safe fallback scale ρ_{safe} shows the clearest influence on performance, reflecting its role in uncertainty buffering. Meanwhile, the variance smoothing factor β_s remains stable near its default value, confirming that variance tracking benefits from conservative temporal filtering.

Ablation: the anchor gates the data-driven signal. ACR couples a data-driven, per-parameter confidence signal with a conservative anchor that bounds it. The two form a single mechanism rather than independent features: the anchor acts only through the confidence the signal produces, so we ablate the mechanism by removing the anchor while retaining the data-driven signal, on the Wave PDE benchmark over 5 seeds (Table 4). Two findings emerge. *First*, replacing RAdam's single, iteration-counted scalar rectification with ACR's per-parameter, data-driven one is the source of the gain: full ACR reaches $2.71 \pm 0.21 \times 10^{-6}$, a 42% reduction over SOAP+RAdam (4.70 ± 0.60) with markedly lower run-to-run variability (std 0.21 vs. 0.60). *Second*, this signal is usable only once anchored: with the anchor removed the per-parameter target is no longer bounded, and the same signal becomes actively destabilising (15.8 ± 0.45, more than $3\times$ worse than SOAP+RAdam). The two are therefore non-redundant by construction — the data-driven rectification supplies the performance, while the anchor makes it safe to use.

Table 4. Ablation on the Wave PDE: removing the conservative anchor that bounds ACR's data-driven per-parameter rectification.

Configuration	Data-driven	Anchor	L2 Err. $\times 10^{-6}$
SOAP+RAdam (baseline)	✗	✗	4.70 ± 0.60
ACR (no anchor)	✓	✗	15.8 ± 0.45
ACR (full)	✓	✓	$\mathbf{2.71 \pm 0.21}$



(a) Confidence & volatility

(b) Target distribution

(c) Confidence-gated blend

Figure 6. Dynamics of ACR on the Wave PDE. Panel a: evolution of conf_t (blue) and CV_t (orange) over training — conf_t rises while CV_t falls and flattens after warmup, so the per-parameter reliability estimate becomes well-defined and stable. Panel b: per-parameter ρ_{target} distribution (rainbow KDEs) versus RAdam's scalar ρ_t (red dots) across 200K steps — RAdam's single rectification ignores a wide distribution early in training and aligns with it only at convergence. Panel c: KDE comparison of conf_t (blue) and the ACR rectification factor r_t^{ACR} (orange), illustrating confidence-gated blending: the distributions decouple early (ACR anchors to the safe fallback) and converge as confidence rises.

Dynamics of our method. In Figure 6 we test that: *(i)* the per-parameter target ρ_{target} is genuinely heterogeneous early in training, so RAdam's single scalar is mis-specified (Figure 6b); and *(ii)* the confidence signal is itself well-defined enough to detect this heterogeneity and gate the step accordingly (Figures 6a and 6c).

First, Figure 6a shows that the confidence signal is itself well-behaved: the empirical signal conf_t rises steadily while the coefficient of variation CV_t decreases and stabilises after the initial warmup, so ACR's per-parameter reliability estimate becomes well-defined and remains stable throughout training.

Second, we investigate the target distribution. Figure 6b exposes the limitation of RAdam that ACR is designed to address: while RAdam enforces a single scalar rectification ρ_t (red dots), the actual per-parameter target ρ_{target} exhibits a broad, heterogeneous distribution, especially in early training; as training progresses the distribution gradually concentrates and aligns with ρ_t, indicating that uniform rectification becomes appropriate only after stabilization.

Third, Figure 6c visualizes ACR's confidence-gated blending: when confidence is low, the rectification factor r_t^{ACR} decouples from conf_t and concentrates near the conservative anchor; as confidence rises, the two distributions converge, showing that ACR has handed control over to RAdam's adaptive rule.

Additional results. We further validate ACR through a comprehensive set of experiments deferred to Appendix C. ACR's robustness holds across (i) PINN architectures (Table 5; MLP, ResNet, PirateNet), (ii) the first-order setting without SOAP (Table 6), where ACR consistently outperforms RAdam in every setting, and (iii) pointwise absolute-error maps against SOAP+RAdam across all four PDE benchmarks (Figure 7).

Limitations. Our empirical study covers canonical 1D/2D PDE benchmarks (Allen–Cahn, Wave, KdV, Burgers); generalisation of ACR to higher-dimensional PDEs, to inverse problems, and to domains beyond PINNs remains future work.

6. Conclusion

We addressed the early-training instability of adaptive optimisers in PINNs, where RAdam's warmup — a single scalar rectification on an iteration-counted schedule, derived under an i.i.d.-Gaussian-gradient assumption — fits the observed dynamics poorly.

Our *Adaptive Confidence Rectification* (ACR) replaces that rule with a per-parameter, data-driven rectification read from the empirical volatility of each second-moment estimate and gated by a conservative anchor; it makes no assumption on the gradient distribution and layers onto any adaptive optimiser. *Theoretically*, ACR keeps the variance its rectification adds to the effective learning rate bounded and strictly contracts the step under low confidence, intervening when the second-moment estimate is unreliable and recovering RAdam once it stabilises. *Empirically*, ACR consistently lowers L^2 error and tightens run-to-run variance relative to RAdam across four PDEs, three model scales, three architectures, and four adaptive optimisers (SOAP, AdaHessian, Sophia, and first-order RAdam). In summary, ACR recasts warmup as a *self-monitoring* process: instead of following a time-based schedule, our method sets the rectification from the empirical statistics.

Acknowledgements

We acknowledge the financial support of the Swiss National Science Foundation (SNF). G. Mercatali has been supported by SNF project IZLJZ2_214000, Interpretable Condition Monitoring.

References

Alimisis, F., Islamov, R., and Lucchi, A. Why do we need warm-up? a theoretical perspective. *arXiv preprint arXiv:2510.03164*, 2025.

Bonfanti, A., Bruno, G., and Cipriani, C. The challenges of the nonlinear regime for physics-informed neural networks. *Advances in Neural Information Processing Systems*, 37:41852–41881, 2024.

Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

Coscia, D., Ivagnes, A., Demo, N., and Rozza, G. Physics-informed neural networks for advanced modeling. *Journal of Open Source Software*, 8(87):5352, 2023.

Dangel, F., Müller, J., and Zeinhofer, M. Kronecker-factored approximate curvature for physics-informed neural networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=jrNlWfor7q.

Gupta, V., Koren, T., and Singer, Y. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Hao, W., Hong, Q., and Jin, X. Gauss newton method for solving variational problems of pdes with neural network discretizaitons. *Journal of scientific computing*, 100(1): 17, 2024.

He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

Hwang, Y. and Lim, D. Dual cone gradient descent for training physics-informed neural networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=gvtCR7dHJ3.

Jnini, A. and Vella, F. Hessian-free natural gradient descent for physics informed machine learning, 2024. URL https://openreview.net/forum?id=Oqk1Ui6m0n.

Jnini, A., Vella, F., and Zeinhofer, M. Gauss-newton natural gradient descent for physics-informed computational fluid dynamics. *arXiv preprint arXiv:2402.10680*, 2024.

Kalra, D. S. and Barkeshli, M. Why warmup the learning rate? underlying mechanisms and improvements. *Advances in Neural Information Processing Systems*, 37: 111760–111801, 2024.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6980.

Kosson, A., Messmer, B., and Jaggi, M. Analyzing & reducing the need for learning rate warmup in gpt training. *Advances in Neural Information Processing Systems*, 37: 2914–2942, 2024.

Liu, H., Li, Z., Hall, D. L. W., Liang, P., and Ma, T. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=3xHDeA8Noi.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgz2aEKDr.

Liu, Q., Chu, M., and Thuerey, N. Config: Towards conflict-free training of physics informed neural networks. *arXiv preprint arXiv:2408.11104*, 2024b.

Liu, Y., Ge, Y., Pan, R., Kang, A., and Zhang, T. Theoretical analysis on how learning rate warmup accelerates convergence. *arXiv preprint arXiv:2509.07972*, 2025.

Luo, S., Tan, Y., Huang, L., Li, J., and Zhao, H. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

Mao, Z., Jagtap, A. D., and Karniadakis, G. E. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

Misyris, G. S., Venzke, A., and Chatzivasileiadis, S. Physics-informed neural networks for power systems. In *2020 IEEE power & energy society general meeting (PESGM)*, pp. 1–5. IEEE, 2020.

Müller, J. and Zeinhofer, M. Achieving high accuracy with pinns via energy natural gradient descent. In *International Conference on Machine Learning*, pp. 25471–25485. PMLR, 2023.

Müller, J. and Zeinhofer, M. Position: Optimization in sciml should employ the function space geometry. In *International Conference on Machine Learning*, pp. 36705–36722. PMLR, 2024.

Pagliardini, M., Ablin, P., and Grangier, D. The adEMAMix optimizer: Better, faster, older. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=jj7b3p5kLY.

Raissi, M., Perdikaris, P., and Karniadakis, G. 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.

Rathore, P., Lei, W., Frangella, Z., Lu, L., and Udell, M. Challenges in training pinns: A loss landscape perspective. *arXiv preprint arXiv:2402.01868*, 2024.

Ryck, T. D., Bonnet, F., Mishra, S., and de Bezenac, E. An operator preconditioning perspective on training in physics-informed machine learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=WWlxFtR5sV.

Schwencke, N. and Furtlehner, C. ANaGRAM: A natural gradient relative to adapted model for efficient PINNs learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=o1IiiNIoaA.

Sivan, H., Gabel, M., and Schuster, A. FOSI: Hybrid first and second order optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NvbeD9Ttkx.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252. PMLR, 2023.

Vyas, N., Morwani, D., Zhao, R., Shapira, I., Brandfonbrener, D., Janson, L., and Kakade, S. M. SOAP: Improving and stabilizing shampoo using adam for language modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IDxZhXrpNf.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

Wang, S., Yu, X., and Perdikaris, P. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Wang, S., Sankaran, S., Wang, H., and Perdikaris, P. An expert's guide to training physics-informed neural networks. *arXiv preprint arXiv:2308.08468*, 2023.

Wang, S., Li, B., Chen, Y., and Perdikaris, P. Piratenets: Physics-informed deep learning with residual adaptive networks. *arXiv preprint arXiv:2402.00326*, 2024.

Wang, S., Bhartari, A. K., Li, B., and Perdikaris, P. Gradient alignment in physics-informed neural networks: A second-order optimization perspective. *arXiv preprint arXiv:2502.00604*, 2025.

Yao, J., Su, C., Hao, Z., Liu, S., Su, H., and Zhu, J. Multi-adam: Parameter-wise scale-invariant optimizer for multi-scale training of physics-informed neural networks. In *International Conference on Machine Learning*, pp. 39702–39721. PMLR, 2023.

Yao, Z., Gholami, A., Shen, S., Mustafa, M., Keutzer, K., and Mahoney, M. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10665–10673, 2021.

Yasunaga, M., Ren, H., Bosselut, A., Liang, P., and Leskovec, J. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 535–546, 2021.

Contents

A Broader Impacts **12**

B Proofs **12**

B.1 Proof of Theorem 4.1 12

B.2 Proof of Proposition 4.2 13

C Additional experimental results **15**

C.1 Absolute Error Analysis of Predicted Solutions. 15

C.2 Robustness across PINN architectures. 15

C.3 Validation of ACR on First-Order Optimizers. 15

C.4 Hyperparameter search for LR of the baseline. 16

D Datasets **17**

D.1 Data Generation. 17

E Model architectures **18**

E.1 MLP. 18

E.2 PirateNet. 18

E.3 ResNet. 18

F Algorithms **20**

F.1 Rectified SOAP with ACR. 20

F.2 Rectified SOAP with RAdam. 21

F.3 Rectified AdaHessian with ACR 22

F.4 Rectified Sophia with ACR. 23

G Additional related works **24**

A. Broader Impacts

This work contributes a training-side improvement to physics-informed neural networks, a class of models used in scientific computing and engineering simulation. More reliable and lower-variance PINN training can reduce the compute cost of building surrogate models for PDE-governed systems, with downstream benefits in domains such as fluid dynamics, materials, and climate modelling. As foundational research on optimisation, the contribution has no direct path to harmful applications, and we are not aware of negative societal impacts specific to this work beyond the general considerations applicable to numerical machine learning.

B. Proofs

B.1. Proof of Theorem 4.1

Proof of Theorem 4.1 (Rectification-Variance Bound) from Section 4.

Notation and Definitions.

Let $g_t \in \mathbb{R}^d$ denote the gradient at step t. The optimizer state and effective learning rates are defined as follows:

1. **Second Moment:**

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}. \tag{3}$$

2. **Rectification Lengths:** RAdam uses a deterministic length ρ_t. ACR introduces an adaptive target:

$$\rho_{\text{target}} = \text{conf}_t \rho_t + (1-\text{conf}_t)\rho_{\text{safe}}, \tag{4}$$

   where $\text{conf}_t = (1+CV_t)^{-1}$ depends on the coefficient of variation of $\hat{v}_t$.

3. **Effective Learning Rates:**

$$\eta_t^{\text{RAdam}} = \frac{\alpha_t\, r(\rho_t)}{\sqrt{\hat{v}_t}+\epsilon}, \tag{5}$$

$$\eta_t^{\text{ACR}} = \frac{\alpha_t\, r(\rho_{\text{target}})}{\sqrt{\hat{v}_t}+\epsilon}. \tag{6}$$

Assumptions.

We impose the following regularity conditions for the analysis:

(A1) ρ_t is deterministic (dependent only on time step t). Stochasticity arises from gradient estimates affecting $\hat{v}_t$ and conf_t.

(A2) The rectification function $r(\rho)$ is Lipschitz continuous on $[\rho_{\min}, \rho_{\max}]$ with constant $L_r > 0$.

(A3) The inverse denominator is well-behaved: $\text{Var}[\hat{v}_t] < \infty$ and $\sqrt{\hat{v}_t}+\epsilon > 0$.

Proof of Variance Bound.

We define the base adaptive scaling factor as $\eta_0 = \frac{\alpha_t}{\sqrt{\hat{v}_t}+\epsilon}$. We can rewrite the effective learning rate for ACR as:

$$\eta_t^{\text{ACR}} = \eta_0\, r(\rho_{\text{target}}). \tag{7}$$

The variance of the product of random variables can be approximated using the decomposition $\text{Var}[XY] \approx \mathbb{E}[X]^2\text{Var}[Y] + \mathbb{E}[Y]^2\text{Var}[X]$ (assuming weak correlation). Applying this to η_t^{ACR}:

$$\text{Var}[\eta_t^{\text{ACR}}] \approx \mathbb{E}[\eta_0]^2 \text{Var}\big[r(\rho_{\text{target}})\big]. \tag{8}$$

Hence, the variance of η_t^{ACR} is controlled by the variance of the rectification term $r(\rho_{\mathrm{target}})$:

$$\mathrm{Var}[\eta_t^{\mathrm{ACR}}] \leq \eta_0^2 \mathrm{Var}[r(\rho_{\mathrm{target}})]. \tag{9}$$

Bounding $\mathrm{Var}[r(\rho_{\mathrm{target}})]$**.** Using the Lipschitz condition (A2):

$$|r(\rho_{\mathrm{target}}) - r(\rho_t)| \leq L_r |\rho_{\mathrm{target}} - \rho_t|. \tag{10}$$

From the definition of ρ_{target}, we have:

$$\rho_{\mathrm{target}} - \rho_t = (1 - \mathrm{conf}_t)(\rho_{\mathrm{safe}} - \rho_t). \tag{11}$$

Taking the variance of both sides (noting that ρ_t and ρ_{safe} are constants at step t):

$$\mathrm{Var}[\rho_{\mathrm{target}}] = (\rho_t - \rho_{\mathrm{safe}})^2 \mathrm{Var}[\mathrm{conf}_t]. \tag{12}$$

Therefore, the variance of the rectified term is bounded by:

$$\mathrm{Var}[r(\rho_{\mathrm{target}})] \leq L_r^2 (\rho_t - \rho_{\mathrm{safe}})^2 \mathrm{Var}[\mathrm{conf}_t]. \tag{13}$$

Final bound. Combining (9) and (13), the variance contributed by the confidence-driven rectification is bounded by

$$\mathrm{Var}[\eta_t^{\mathrm{ACR}}] \leq \eta_0^2 L_r^2 (\rho_t - \rho_{\mathrm{safe}})^2 \,\mathrm{Var}[\mathrm{conf}_t] = \frac{\alpha_t^2 L_r^2}{(\sqrt{\hat{v}_t} + \epsilon)^2} (\rho_t - \rho_{\mathrm{safe}})^2 \,\mathrm{Var}[\mathrm{conf}_t] = \Delta_t. \tag{14}$$

The bound is non-negative and vanishes as $\mathrm{conf}_t \to 1$ (so $\rho_{\mathrm{target}} \to \rho_t$ and ACR recovers RAdam); by Proposition 4.2, whenever $\rho_t > \rho_{\mathrm{safe}}$ ACR additionally satisfies $r(\rho_{\mathrm{target}}) \leq r(\rho_t)$, i.e. its step is pointwise no larger than RAdam's. □

B.2. Proof of Proposition 4.2

Restatement of Proposition. *Let ρ_t be the RAdam variance tractability parameter at step t, and let ρ_{safe} be the conservative fallback parameter. For any iteration t such that $\rho_t > \rho_{\mathrm{safe}}$, the ACR rectification coefficient $r(\rho_{\mathrm{target}})$ satisfies:*

$$r(\rho_{\mathrm{target}}) \leq r(\rho_t) - \mathcal{C}_t \cdot (1 - \mathrm{conf}_t)$$

where $\mathcal{C}_t > 0$ depends on the derivative of the rectification function.

Proof. Let the rectification function $r(\rho)$ be defined as in the standard RAdam algorithm:

$$r(\rho) = \sqrt{\frac{(\rho - 4)(\rho - 2)\rho_\infty}{(\rho_\infty - 4)(\rho_\infty - 2)\rho}}. \tag{15}$$

It is established in the RAdam literature that $r(\rho)$ is a monotonically increasing function for $\rho \in (4, \infty)$. Consequently, its derivative satisfies $r'(\rho) > 0$ in this domain.

In the ACR algorithm, the target tractability parameter ρ_{target} is defined as a convex combination of the current RAdam parameter ρ_t and the safe fallback ρ_{safe}, weighted by the confidence score $\mathrm{conf}_t \in [0, 1]$:

$$\rho_{\mathrm{target}} = \mathrm{conf}_t \cdot \rho_t + (1 - \mathrm{conf}_t) \cdot \rho_{\mathrm{safe}}. \tag{16}$$

We examine the regime where the standard RAdam schedule has grown aggressive, such that $\rho_t > \rho_{\mathrm{safe}}$. Under this condition, for any imperfect confidence ($\mathrm{conf}_t < 1$), we have $\rho_{\mathrm{target}} < \rho_t$. Due to the monotonicity of $r(\cdot)$, it immediately follows that the ACR step size is strictly bounded:

$$r(\rho_{\mathrm{target}}) < r(\rho_t). \tag{17}$$

To quantify this contraction, we apply the Mean Value Theorem to $r(\cdot)$ on the interval $[\rho_{\mathrm{target}}, \rho_t]$. There exists some $\xi \in (\rho_{\mathrm{target}}, \rho_t)$ such that:

$$r(\rho_t) - r(\rho_{\mathrm{target}}) = r'(\xi)(\rho_t - \rho_{\mathrm{target}}). \tag{18}$$

Substituting the definition of ρ_{target} into the difference term:

$$\rho_t - \rho_{\text{target}} = \rho_t - [\text{conf}_t \rho_t + (1 - \text{conf}_t)\rho_{\text{safe}}] \tag{19}$$

$$= (1 - \text{conf}_t)(\rho_t - \rho_{\text{safe}}). \tag{20}$$

Substituting Eq. (8) back into Eq. (7) yields:

$$r(\rho_{\text{target}}) = r(\rho_t) - r'(\xi)(\rho_t - \rho_{\text{safe}})(1 - \text{conf}_t). \tag{21}$$

Since $\rho_t > \rho_{\text{safe}}$ and $r'(\xi) > 0$, we define the constant $\mathcal{C}_t = r'(\xi)(\rho_t - \rho_{\text{safe}})$. This constant is strictly positive for fixed t, yielding the final bound:

$$r(\rho_{\text{target}}) = r(\rho_t) - \mathcal{C}_t(1 - \text{conf}_t). \tag{22}$$

This confirms that the effective step size of ACR is contracted relative to RAdam in direct proportion to the lack of confidence $(1 - \text{conf}_t)$. □

C. Additional experimental results

C.1. Absolute Error Analysis of Predicted Solutions.

We evaluate the pointwise absolute error between the predicted and ground-truth fields (Wang et al., 2025; Müller & Zeinhofer, 2023). Figure 7 presents absolute error maps computed using the final model parameters for each of the four PDE benchmarks. Our method SOAP+ACR consistently shows darker maps against SOAP+RAdam, indicating lower errors.

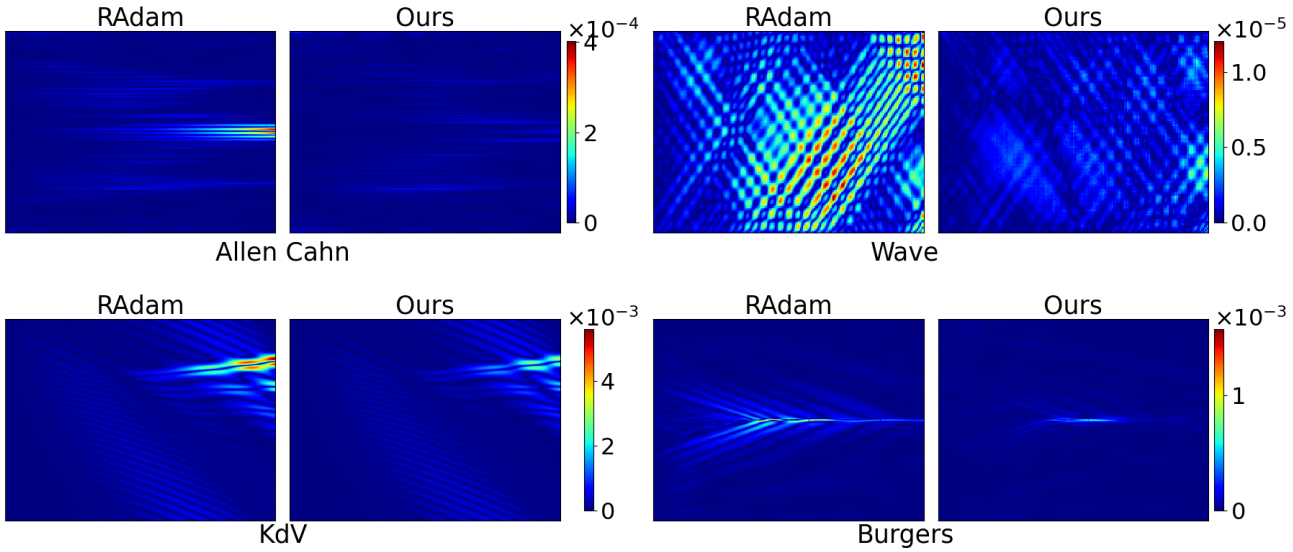


Figure 7. Per-coordinate absolute-error maps of SOAP+RAdam vs. SOAP+ACR at convergence, on Allen–Cahn, Wave, KdV, and Burgers.

C.2. Robustness across PINN architectures.

We consider PINN architecture methods, including MLP (Wang et al., 2025), ResNet (Wang et al., 2024), PirateNet (Wang et al., 2024). We demonstrate for the SOAP + rectification (Algorithm 2), over such architectures. Note that PirateNet and ResNet are significantly larger compared to MLP. In Table 5 we show the L2 error results. Our method consistently outperforms RAdam rectification method in all cases.

Table 5. L2 results over multiple PINN architectures.

Dataset	**MLP**		**ResNet**		**PirateNet**	
	SOAP+RAdam	Ours	SOAP+RAdam	Ours	SOAP+RAdam	Ours
Allen Cahn ($\times 10^{-6}$)	41.0±8.6	22.1±3.7	6.25±0.61	4.40±0.05	4.59±0.18	4.02±0.05
Wave ($\times 10^{-6}$)	4.70±0.60	2.71±0.21	3.58±0.54	2.52±0.02	2.32±0.89	1.73±0.13
KdV ($\times 10^{-4}$)	5.32±0.53	4.08±0.25	3.09±0.08	2.68±0.19	3.74±0.07	3.10±0.18
Burgers ($\times 10^{-5}$)	8.55±0.49	5.96±0.66	7.94±1.19	2.96±0.18	6.55±0.02	5.14±0.10

C.3. Validation of ACR on First-Order Optimizers.

Table 6 presents the performance of our method in the first-order setting. We compare RAdam against our proposed approach. Across all datasets, ACR consistently yields improved results.

Table 6. L2 error in the first-order setting (RAdam vs. ACR, without SOAP).

Model	**Wave** ($\times 10^{-5}$)	**Allen** ($\times 10^{-4}$)	**KdV** ($\times 10^{-3}$)	**Burgers** ($\times 10^{-3}$)
RAdam	4.68 ± 0.12	10.40 ± 0.80	2.86 ± 0.05	3.19 ± 0.04
ACR	3.50 ± 0.14	8.30 ± 0.35	2.04 ± 0.01	1.27 ± 0.01

C.4. Hyperparameter search for LR of the baseline.

A key concern is that the improvements of ACR might stem from an unfavorable learning rate choice for the baseline, rather than from the mechanism itself. To rule this out, we conducted a dedicated learning rate sweep for the baseline optimizer (SOAP+RAdam) without ACR. This isolates the baseline's optimal configuration and ensures a fair comparison.

Table 7 reports the results of a broad learning rate search across four PDE benchmarks. The sweep spans two orders of magnitude and includes commonly used values in prior work. We observe a consistent optimum at **LR = 0.001**, which yields the lowest average L2 error across tasks. All baseline comparisons in the paper therefore use this empirically determined optimal LR.

This result confirms that the baseline operates near its best achievable performance with respect to LR, and thus the improvements introduced by ACR cannot be attributed to a poorly tuned or mismatched baseline learning rate.

Table 7. Learning Rate Sweep for Baseline SOAP+RAdam.

LR	**Allen** $\times 10^{-6}$	**Wave** $\times 10^{-6}$	**Burgers** $\times 10^{-5}$	**KdV** $\times 10^{-4}$
0.0001	225 ± 62	28.39 ± 1.53	677 ± 31	7.54 ± 0.52
0.0002	99.5 ± 16	7.76 ± 0.27	167 ± 11	6.02 ± 0.19
0.0004	91.8 ± 25	5.62 ± 0.44	29.0 ± 4.2	5.69 ± 0.47
0.0008	45.6 ± 6.5	5.24 ± 0.64	8.83 ± 1.01	5.65 ± 0.22
0.001	$\mathbf{41.0 \pm 8.6}$	$\mathbf{4.70 \pm 0.60}$	$\mathbf{8.55 \pm 0.49}$	$\mathbf{5.32 \pm 0.53}$
0.002	48.5 ± 1.5	4.92 ± 0.23	9.08 ± 0.76	5.54 ± 0.18

D. Datasets

D.1. Data Generation.

The reference datasets were generated using the numerical package Chebfun in MATLAB. The data was initially simulated with a fine time resolution of $dt = 10^{-4}$, and then temporally downsampled to produce the final dataset used for analysis. Table 8 summarizes the parameters of the partial differential equations (PDEs) and key details of the generated datasets.

Table 8. Parameter and numerical configurations for generating the PDE solutions.

PDE	Parameter	Package	Resolution
Allen Cahn	$\epsilon = 10^{-4}, a = 5$	Chebfun	200×512
Wave	$c = 4$	N/A	200×128
KdV	$\eta = 1, \mu = 0.022$	Chebfun	200×512
Burgers	$\nu = 0.01\,\pi$	Chebfun	200×512

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

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

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

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

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

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 η governs the strength of the nonlinearity, while μ controls the dispersion level. Under the KdV dynamics, this initial wave evolves into a series of solitary-type waves.

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

$$u_t + u u_x = \nu u_{xx},$$

where u represents the velocity field, and ν 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:

$$\begin{aligned}
u(x,0) &= -\sin(\pi x),\\
u(-1,t) &= u(1,t) = 0,
\end{aligned}$$

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

E. Model architectures

In our experiments, we use MLP, ResNet, PirateNet architectures from (Wang et al., 2025).

E.1. MLP.

When comparing multiple datasets, i.e., in Figure 3, we use the MLP with one layer. When comparing over multiple model sizes, i.e., in Table 1, we compare the MLP with 1, 3 and 4 layers.

The used architecture is defined as follows. We use a multilayer perceptron (MLP) defined as a composition of L fully connected layers with hidden dimension d_h, where $L = 4$ and $d_h = 256$ by default. The input $x \in \mathbb{R}^n$ is optionally encoded using a positional embedding function $E(\cdot)$, which may include periodic features or Fourier embeddings, resulting in an encoded input $\tilde{x} = E(x)$. Each hidden layer applies a linear transformation followed by a non-linear activation function $\sigma(\cdot)$, typically $\tanh$:

$$h^{(0)} = \tilde{x}, \quad h^{(l)} = \sigma(W^{(l)} h^{(l-1)} + b^{(l)}), \quad \text{for } l = 1, \ldots, L,$$

where $W^{(l)} \in \mathbb{R}^{d_h \times d_h}$ and $b^{(l)} \in \mathbb{R}^{d_h}$ are the weights and biases of layer l, possibly reparameterized as specified. The final output y is computed through a final learned linear layer:

$$y = W^{(L+1)} h^{(L)} + b^{(L+1)}.$$

E.2. PirateNet.

The PirateNet is designed to address the initialization problem in neural networks. It features a specialized forward pass with residual blocks and gate-like functions derived from two encoding maps, U and V, based on an embedding function $\Phi(x)$. The forward pass for each residual block l is defined as follows:

$$f^{(l)} = \sigma(W_1^{(l)} x^{(l)} + b_1^{(l)}), \tag{23}$$

$$z_1^{(l)} = f^{(l)} \odot U + (1 - f^{(l)}) \odot V, \tag{24}$$

$$g^{(l)} = \sigma(W_2^{(l)} z_1^{(l)} + b_2^{(l)}), \tag{25}$$

$$z_2^{(l)} = g^{(l)} \odot U + (1 - g^{(l)}) \odot V, \tag{26}$$

$$h^{(l)} = \sigma(W_3^{(l)} z_2^{(l)} + b_3^{(l)}), \tag{27}$$

$$x^{(l+1)} = \alpha^{(l)} h^{(l)} + (1 - \alpha^{(l)}) x^{(l)}, \tag{28}$$

where $\odot$ denotes point-wise multiplication, and $\alpha^{(l)}$ is a trainable parameter.

The input coordinates x are first mapped to a high-dimensional feature space using a random Fourier embedding function $\Phi(x) = \begin{bmatrix} \cos(Bx) \\ \sin(Bx) \end{bmatrix}$, with B sampled from a Gaussian distribution. The encoded embeddings $\Phi(x)$ are then sent into two dense layers to create the gate functions $U = \sigma(W_1 \Phi(x) + b_1)$ and $V = \sigma(W_2 \Phi(x) + b_2)$.

The final output of a PirateNet of L residual blocks is given by $\mathbf{u}_\theta = W^{(L+1)} \mathbf{x}^{(L)}$.

E.3. ResNet.

The ResNet for PINNs is an architecture that incorporates residual connections to help mitigate the vanishing gradient problem in deep neural networks. This structure allows for the training of deeper models, which can better approximate complex solutions to partial differential equations (PDEs). The core of the architecture lies in its residual blocks, which add the input of the block to its output.

The forward pass for a single residual block is given by:

$$\mathbf{y}^{(l+1)} = \mathbf{y}^{(l)} + \sigma(W_2^{(l)}(\sigma(W_1^{(l)} \mathbf{y}^{(l)} + \mathbf{b}_1^{(l)})) + \mathbf{b}_2^{(l)}), \tag{29}$$

where $\mathbf{y}^{(l)}$ is the input to the block, σ is an activation function, $W_1^{(l)}$ and $W_2^{(l)}$ are the weight matrices, and $\mathbf{b}_1^{(l)}$ and $\mathbf{b}_2^{(l)}$ are the bias vectors.

For a PINN, the input is typically the coordinates (x, t). This input is passed through a series of residual blocks. The input to the first layer, $\mathbf{y}^{(0)}$, is the coordinate vector, i.e., $\mathbf{y}^{(0)} = [x, t]^T$. The output of the final layer, $\mathbf{y}^{(L)}$, is then mapped to the final solution $\mathbf{u}_\theta$ through a linear transformation:

$$\mathbf{u}_\theta(x, t) = W^{(L+1)}\mathbf{y}^{(L)} + \mathbf{b}^{(L+1)}. \tag{30}$$

This output $\mathbf{u}_\theta$ is then used to construct the physics-informed loss function, which consists of a data-driven part and a physics-informed part. The loss function is minimized to train the network.

F. Algorithms

We present several applications of our adaptive rectification method into other optimizers.

F.1. Rectified SOAP with ACR.

Algorithm 2 illustrates the application of our method to SOAP (Vyas et al., 2025).

Algorithm 2 Our Method: Adaptive Confidence Rectification for SOAP

Input: Data distribution D, initial parameters $\theta^{(0)}$, learning rate η, small constant ϵ, SOAP parameters $\beta_1, \beta_2, \lambda$, ACR parameters $\beta_s = 0.99$, $\rho_{\text{safe}} = 10$
Initialize $m^{(0)} \leftarrow \mathbf{0}$, $v^{(0)} \leftarrow \mathbf{0}$
Initialize $s^{(0)} \leftarrow 0$
Sample $x \sim D$
Compute gradient $g^{(t)} \leftarrow \nabla_\theta \mathcal{L}_{\theta^{(t-1)}}(x)$
Project into Eigenspace
$m^{(t)} \leftarrow \beta_1 m^{(t-1)} + (1-\beta_1) g^{(t)}$
$\hat{m}^{(t)} \leftarrow \frac{m^{(t)}}{1-\beta_1^t}$
$m'^{(t)} \leftarrow Q_L^\top \hat{m}^{(t)} Q_R$
$g'^{(t)} \leftarrow Q_L^\top g^{(t)} Q_R$
Adam-style Update
$v^{(t)} \leftarrow \beta_2 v^{(t-1)} + (1-\beta_2) {g'^{(t)}}^2$
$\hat{v}^{(t)} \leftarrow \frac{v^{(t)}}{1-\beta_2^t}$
$\rho_\infty \leftarrow \frac{2}{1-\beta_2} - 1$
$\rho_t \leftarrow \rho_\infty - \frac{2t\beta_2^t}{1-\beta_2^t}$
ACR: Adaptive Confidence Rectification
$s^{(t)} \leftarrow \beta_s s^{(t-1)} + (1-\beta_s)(v^{(t)} - v^{(t-1)})^2$
$CV_t \leftarrow \frac{\sqrt{s^{(t)}}}{\hat{v}^{(t)}+\epsilon}$
$\text{conf}_t \leftarrow \frac{1}{1+CV_t}$
$\rho_{\text{target}} \leftarrow \text{conf}_t \rho_t + (1-\text{conf}_t)\rho_{\text{safe}}$
$r_t^{\text{ACR}} \leftarrow \sqrt{\frac{(\rho_{\text{target}}-4)(\rho_{\text{target}}-2)\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho_{\text{target}}}}$
$N' \leftarrow r_t^{\text{ACR}} \frac{m'^{(t)}}{\sqrt{\hat{v}^{(t)}}+\epsilon}$
Project Back
$N \leftarrow Q_L N' Q_R^\top$
$\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta(N + \lambda\theta^{(t-1)})$
Update Preconditioners
$L \leftarrow \beta_2 L + (1-\beta_2) g g^\top$
$R \leftarrow \beta_2 R + (1-\beta_2) g^\top g$
if $t \bmod f = 0$ **then**
  $Q_L \leftarrow \text{eigenvec}(L, Q_L)$
  $Q_R \leftarrow \text{eigenvec}(R, Q_R)$
end if

F.2. Rectified SOAP with RAdam.

We provide a rectified version of the baseline SOAP optimizer by integrating the variance rectification mechanism from RAdam (Liu et al., 2020). The resulting algorithm, SOAP + RAdam, preserves the eigenspace preconditioning of SOAP while correcting for the variance of the adaptive learning rate during the early stages of training. The full procedure is summarized in Algorithm 3.

Algorithm 3 SOAP + RAdam Algorithm. Differences with SOAP are in blue.

Input: Data distribution D, initial parameters $\theta^{(0)}$, learning rate η, small constant ϵ, SOAP parameters $\beta_1, \beta_2, \lambda$, update frequency f
Initialize $m^{(0)} \leftarrow \mathbf{0}$, $v^{(0)} \leftarrow \mathbf{0}$
Sample $x \sim D$
Compute gradient $g^{(t)} \leftarrow \nabla_\theta \mathcal{L}_{\theta^{(t-1)}}(x)$
$\rho_\infty \leftarrow \frac{2}{1-\beta_2} - 1$ {Max length of approximated SMA}
Project into Eigenspace
$m^{(t)} \leftarrow \beta_1 m^{(t-1)} + (1-\beta_1) g^{(t)}$
$\hat{m}^{(t)} \leftarrow \frac{m^{(t)}}{1-\beta_1^t}$
$m'^{(t)} \leftarrow Q_L^\top \hat{m}^{(t)} Q_R$
$g'^{(t)} \leftarrow Q_L^\top g^{(t)} Q_R$
Adam-style Update
$v^{(t)} \leftarrow \beta_2 v^{(t-1)} + (1-\beta_2) {g'^{(t)}}^2$
$\hat{v}^{(t)} \leftarrow \frac{v^{(t)}}{1-\beta_2^t}$
$\rho_t \leftarrow \rho_\infty - \frac{2t\beta_2^t}{1-\beta_2^t}$ {Length of approximated SMA}
if $\rho_t > 4$ **then**
  $r_t \leftarrow \sqrt{\frac{(\rho_t-4)(\rho_t-2)\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho_t}}$ {Variance rectification}
  $N' \leftarrow r_t \frac{m'^{(t)}}{\sqrt{\hat{v}^{(t)}}+\epsilon}$
else
  $N' \leftarrow \frac{m'^{(t)}}{\sqrt{\hat{v}^{(t)}}+\epsilon}$
end if
$N \leftarrow Q_L N' Q_R^\top$
$\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta \left(N + \lambda \theta^{(t-1)}\right)$
Update Preconditioners
$L \leftarrow \beta_2 L + (1-\beta_2) g g^\top$
$R \leftarrow \beta_2 R + (1-\beta_2) g^\top g$
if $t \bmod f = 0$ **then**
  $Q_L \leftarrow \text{eigenvec}(L, Q_L)$
  $Q_R \leftarrow \text{eigenvec}(R, Q_R)$
end if

F.3. Rectified AdaHessian with ACR

In Algorithm 4, we provide the pseudocode for applying our Adaptive Confidence Rectification (ACR) method to AdaHessian (Yao et al., 2021). This version of AdaHessian uses the Hutchinson estimator to approximate the Hessian diagonal, which is then used as the second-moment estimate. Our ACR method rectifies the update based on the statistical confidence of this diagonal approximation. The ACR framework is general and can be applied to a variety of optimizers, and we found it to be a powerful complement to second-order methods like AdaHessian.

Algorithm 4 Adaptive Confidence Rectification (ACR) for AdaHessian. Differences are in blue.

Input: Data distribution D, initial parameters $\theta^{(0)}$, learning rate η, small constant ϵ, AdaHessian parameters β_1, β_2, k, ACR parameters $\beta_s, \rho_{\text{safe}}$, weight decay λ_{wd}

Initialize $m^{(0)} \leftarrow \mathbf{0}$, $v^{(0)} \leftarrow \mathbf{0}$

Initialize $s^{(0)} \leftarrow \mathbf{0}$

Sample $x \sim D$

Compute gradient $g^{(t)} \leftarrow \nabla_\theta \mathcal{L}_{\theta^{(t-1)}}(x)$

$\text{diag}(h^{(t)}) \leftarrow \text{Hutchinson}\big(\nabla^2_\theta \mathcal{L}(x)\big)$ {Approximate Hessian diagonal}

$m^{(t)} \leftarrow \beta_1 m^{(t-1)} + (1-\beta_1) g^{(t)}$

$\hat{m}^{(t)} \leftarrow \frac{m^{(t)}}{1-\beta_1^t}$

$v^{(t)} \leftarrow \beta_2 v^{(t-1)} + (1-\beta_2)\text{diag}(h^{(t)})$

$\hat{v}^{(t)} \leftarrow \frac{v^{(t)}}{1-\beta_2^t}$

$\rho_\infty \leftarrow \frac{2}{1-\beta_2} - 1$

$\rho_t \leftarrow \rho_\infty - \frac{2t\beta_2^t}{1-\beta_2^t}$ {Length of approximated SMA}

ACR: Adaptive Confidence Rectification

$s^{(t)} \leftarrow \beta_s s^{(t-1)} + (1-\beta_s)\left(v^{(t)} - v^{(t-1)}\right)^2$

$CV_t \leftarrow \frac{\sqrt{s^{(t)}}}{\hat{v}^{(t)}+\epsilon}$

$\text{conf}_t \leftarrow \frac{1}{1+CV_t}$

$\rho_{\text{target}} \leftarrow \text{conf}_t \rho_t + (1-\text{conf}_t)\rho_{\text{safe}}$

$r_t^{\text{ACR}} \leftarrow \sqrt{\frac{(\rho_{\text{target}}-4)(\rho_{\text{target}}-2)\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho_{\text{target}}}}$

$\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta\left(r_t^{\text{ACR}} \frac{\hat{m}^{(t)}}{\left(\sqrt{\hat{v}^{(t)}}\right)^k + \epsilon} + \lambda_{\text{wd}}\, \theta^{(t-1)}\right)$

F.4. Rectified Sophia with ACR.

In Algorithm 5, we provide the pseudocode for applying our Adaptive Confidence Rectification (ACR) method to Sophia (Liu et al., 2024a). This version uses the Hutchinson estimator to approximate the Hessian diagonal, which is then used as the second-moment estimate. Our ACR method rectifies the update based on the statistical confidence of this diagonal approximation. The ACR framework is general and can be applied to a variety of optimizers, and we found it to be a powerful complement to second-order methods like Sophia.

Algorithm 5 Adaptive Confidence Rectification (ACR) for Sophia with Clipped Update

Input: Data distribution D, initial parameters $\theta^{(0)}$, learning rate η, small constant ϵ, Sophia parameters β_1, β_2, γ, ACR parameters $\beta_s, \rho_{\text{safe}}$, weight decay λ_{wd}
Initialize $m^{(0)} \leftarrow \mathbf{0}$, $v^{(0)} \leftarrow \mathbf{0}$
Initialize $s^{(0)} \leftarrow \mathbf{0}$
Sample $x \sim D$
Compute gradient $g^{(t)} \leftarrow \nabla_\theta \mathcal{L}_{\theta^{(t-1)}}(x)$
$\text{diag}(h^{(t)}) \leftarrow \text{Hutchinson}\big(\nabla_\theta^2 \mathcal{L}(x)\big)$ {Approximate Hessian diagonal}
$m^{(t)} \leftarrow \beta_1 m^{(t-1)} + (1-\beta_1) g^{(t)}$
$\hat{m}^{(t)} \leftarrow \frac{m^{(t)}}{1-\beta_1^t}$
$v^{(t)} \leftarrow \beta_2 v^{(t-1)} + (1-\beta_2)\,\text{diag}(h^{(t)})$
$\hat{v}^{(t)} \leftarrow \frac{v^{(t)}}{1-\beta_2^t}$
$\rho_\infty \leftarrow \frac{2}{1-\beta_2} - 1$
$\rho_t \leftarrow \rho_\infty - \frac{2t\beta_2^t}{1-\beta_2^t}$ {Length of approximated SMA}
ACR: Adaptive Confidence Rectification
$s^{(t)} \leftarrow \beta_s s^{(t-1)} + (1-\beta_s)\left(v^{(t)} - v^{(t-1)}\right)^2$
$CV_t \leftarrow \frac{\sqrt{s^{(t)}}}{\hat{v}^{(t)}+\epsilon}$
$\text{conf}_t \leftarrow \frac{1}{1+CV_t}$
$\rho_{\text{target}} \leftarrow \text{conf}_t \rho_t + (1-\text{conf}_t)\rho_{\text{safe}}$
$r_t^{\text{ACR}} \leftarrow \sqrt{\frac{(\rho_{\text{target}}-4)(\rho_{\text{target}}-2)\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho_{\text{target}}}}$
$\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta\,\text{clip}\left(r_t^{\text{ACR}} \frac{\hat{m}^{(t)}}{\gamma \hat{v}^{(t)}+\epsilon}\right)$

G. Additional related works

Adaptive momentum-based first order methods. Adam (Kingma & Ba, 2015) remains the most widely used momentum-based optimization algorithm, leveraging adaptive learning rates and exponential moving averages of gradients. RAdam (Liu et al., 2020) addresses the high variance of Adam's adaptive learning rate by introducing a rectification mechanism that stabilizes training in the early phases. AdEMAMix (Pagliardini et al., 2025) extends this family by incorporating a secondary exponential moving average of gradients, enabling the optimizer to retain information from older gradient directions. While our approach also builds upon momentum, it differs fundamentally: rather than modifying the momentum or curvature estimates, ACR rescales the adaptive learning rate from an empirical, per-parameter estimate of how reliable each second-moment estimate currently is, and can be combined with any of these optimizers.

Second-Order Optimization for LLMs. Sivan et al. (2024) propose decomposing the gradient into two orthogonal components, applying Newton's method to one and Adam to the other. Gupta et al. (2018) approximate the empirical Fisher matrix by maintaining two square preconditioners, which are combined with the gradient using a Kronecker product. Building on this, Vyas et al. (2025) further project the preconditioners into the eigenspace to improve efficiency. Liu et al. (2024a) approximate the diagonal of the Gauss-Newton matrix for scalable preconditioning. In contrast to these approaches, we propose rescaling the adaptive learning rate to mitigate its high variance during the early stages of training.

Functional PINN Optimizers. Müller & Zeinhofer (2023) propose the use of the Energy Natural Gradient (ENG) method that is equivalent to a newton method in function space. Dangel et al. (2024) adapt the ENG method with a layerwise structure. Schwencke & Furtlehner (2025) explore optimization in the function space induced by the neural tangent kernel (NTK), and Jnini & Vella (2024) develop a Hessian-free optimization method in functional space. Additionally, Jnini et al. (2024) propose a Gauss-Newton method tailored for functional optimization in PINNs. A comprehensive overview of functional optimization approaches in scientific machine learning is presented by Müller & Zeinhofer (2024). Unlike these functional methods, ACR does not reformulate the optimisation in function space; it operates directly on a standard optimizer's adaptive learning rate, and is compatible with minibatch training and layerwise application.

PINN Methods with Eigenvalues. Wang et al. (2022) leverage the eigenvalues of the Neural Tangent Kernel (NTK) to mitigate disparities in the convergence rates of different PINN losses. Schwencke & Furtlehner (2025) compute eigenvalues from concatenated domain and boundary features, while Jnini & Vella (2024) extract eigenvalues from the Gauss-Newton (GN) matrix. Wang et al. (2025) introduce SOAP, an approximation of the GN matrix that computes eigenpairs via the power iteration algorithm. Our method is complementary to these eigenvalue-based approaches: ACR does not itself compute or project onto eigenvectors, but rectifies the adaptive learning rate of the underlying optimizer (e.g. SOAP), benefiting from such preconditioning while remaining agnostic to it.

Other Optimizers for PINNs. Rathore et al. (2024) propose a sequential optimization strategy for PINNs, combining Adam, L-BFGS, and a Nyström-accelerated Newton conjugate gradient method (NNCG). Hao et al. (2024) introduce a Gauss-Newton method tailored for solving PDEs with neural networks. Yao et al. (2023) suggest applying separate Adam optimizers to each component of the PINN loss function, such as the boundary and initial conditions. Liu et al. (2024b) leverage multitask learning to address gradient imbalances between initial/boundary conditions and data terms. Similarly, Hwang & Lim (2024) propose constraining gradients to lie within a dual cone region to improve training dynamics.