PDE-aware Optimizer for Physics-informed Neural Networks

Hardik Shukla, Manurag Khullar, Vismay Churiwala

ENM 5320: AI4Science

Abstract

Physics-Informed Neural Networks (PINNs) have emerged as a powerful framework for solving partial differential equations (PDEs) by embedding physical constraints into the loss function. However, standard optimizers such as Adam often struggle to balance competing loss terms, particularly in stiff or ill-conditioned systems. In this work, we propose a PDE-aware optimizer that adapts parameter updates based on the variance of per-sample PDE residual gradients. This method addresses gradient misalignment without incurring the heavy computational costs of second-order optimizers such as SOAP. We benchmark the PDE-aware optimizer against Adam and SOAP on 1D Burgers', Allen–Cahn and Korteweg–de Vries(KdV) equations. Across both PDEs, the PDE-aware optimizer achieves smoother convergence and lower absolute errors, particularly in regions with sharp gradients. Our results demonstrate the effectiveness of PDE residual-aware adaptivity in enhancing stability in PINNs training. While promising, further scaling on larger architectures and hardware accelerators remains an important direction for future research.

1 Introduction

A significant advancement in scientific machine learning is physics-informed machine learning, which integrates physical laws and domain-specific constraints directly into the learning process. This integration can be achieved in various ways, including modifications to key components such as the model architecture, loss functions, optimisation algorithms, and hyperparameter tuning [2, 5]. A particularly effective strategy within physics-informed machine learning is embedding physical principles such as conservation laws, symmetries, invariances, and equivariances into the machine learning framework. Among the most flexible and widely adopted approaches is the use of tailored loss functions. These loss functions encode the physical constraints as soft penalties, guiding the model to learn solutions that are not only data-driven but also consistent with known physical laws. This approach has led to the development of Physics-Informed Neural Networks (PINNs), which have been successfully applied to both forward and inverse problems involving partial differential equations (PDE).

While PINNs offer conceptual simplicity and implementation flexibility, training them effectively for stiff or ill-conditioned PDEs remains challenging. Traditional first-order optimization methods, such as Adam, often struggle with convergence or require extensive tuning. To address these issues, researchers have turned to second-order optimization techniques, which utilize curvature information to make more informed updates to the model parameters [8, 7]. Second-order optimizers such as Second-Order Adaptive Optimisation (SOAP) and Second-order Clipping Shampoo (SHAMPOO) have demonstrated superior convergence properties in training PINNs, particularly on complex PDE problems [8]. These methods leverage second-order information, such as approximations of the Hessian or the Fisher information matrix,

to scale the gradient updates more precisely along each parameter direction, resulting in improved training stability and accuracy. However, these benefits come at a significant cost.

The primary drawback of second-order methods lies in their high computational and memory overhead. Maintaining and updating curvature matrices—often of size proportional to the number of parameters squared—is resource-intensive, especially for deep networks or large-scale problems [3, 4]. Moreover, operations such as matrix inversion or eigendecomposition add to the computational complexity [6]. Despite these challenges, ongoing research aims to make second-order methods more tractable through matrix sketching, low-rank, and block-diagonal approximations [1, 9].

In this project, we aim to develop a PDE-aware optimizer that can:

- Align gradients from multiple loss terms (PDE residual, boundary, and initial conditions),
- Adaptively scale updates to prevent unstable PDE gradients from dominating, and
- Avoid expensive operations like Hessian inversion, making it practical for real-world problems.

2 Loss terms in PINNs

We consider a general formulation of a parabolic partial differential equation (PDE) given by

$$\mathbf{u}_t + \mathcal{D}[\mathbf{u}] = \mathbf{f}, \qquad (t, \mathbf{x}) \in [0, T] \times \Omega \subset R^{1+d}. \tag{1}$$

where Ω is a bounded subset of R^d with a sufficiently regular boundary denoted by $\partial\Omega$. Here, $\mathcal{D}[\cdot]$ represents a linear or nonlinear differential operator, and $\mathbf{u}(t, \mathbf{x})$ denotes the unknown solution to be approximated.

The PDE is supplemented with the following initial and boundary conditions:

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

$$\mathcal{B}[\mathbf{u}](t, \mathbf{x}) = 0, \qquad t \in [0, T],\ \mathbf{x} \in \partial\Omega. \tag{2b}$$

Here, $\mathbf{f}$ and $\mathbf{g}$ are given functions with appropriate smoothness, and $\mathcal{B}[\cdot]$ is a boundary operator which may represent Dirichlet, Neumann, Robin, or periodic boundary conditions. To approximate the solution $\mathbf{u}(t, \mathbf{x})$, we use a deep neural network $u_\theta(t, \mathbf{x})$ parameterized by θ, which includes all trainable parameters such as weights and biases. When a smooth activation function is employed, the neural network u_θ yields a differentiable approximation that can be evaluated at any point $(t, \mathbf{x})$. Moreover, automatic differentiation enables efficient computation of the necessary derivatives with respect to both the input variables and network parameters.

Using this framework, we define the residuals for the PDE, initial condition, and boundary condition as follows:

$$\mathcal{R}_{\text{int}}[u_\theta](t, \mathbf{x}) = \frac{\partial u_\theta}{\partial t}(t, \mathbf{x}) + \mathcal{D}[u_\theta](t, \mathbf{x}) - f(\mathbf{x}), \quad (t, \mathbf{x}) \in [0, T] \times \Omega, \tag{3}$$

$$\mathcal{R}_{\text{bc}}[u_\theta](t, \mathbf{x}) = \mathcal{B}[u_\theta](t, \mathbf{x}), \quad (t, \mathbf{x}) \in [0, T] \times \partial\Omega, \tag{4}$$

$$\mathcal{R}_{\text{ic}}[u_\theta](\mathbf{x}) = u_\theta(0, \mathbf{x}) - g(\mathbf{x}), \quad \mathbf{x} \in \Omega. \tag{5}$$

The physics-informed neural network (PINN) is then trained by minimising the following composite empirical loss function, which aggregates the residuals from the initial condition, boundary condition, and PDE:

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{N_{\text{ic}}} \sum_{i=1}^{N_{\text{ic}}} \left| \mathcal{R}_{\text{ic}}[u_\theta](\mathbf{x}_{\text{ic}}^i) \right|^2}_{\mathcal{L}_{\text{ic}}(\theta)} + \underbrace{\frac{1}{N_{\text{bc}}} \sum_{i=1}^{N_{\text{bc}}} \left| \mathcal{R}_{\text{bc}}[u_\theta](t_{\text{bc}}^i, \mathbf{x}_{\text{bc}}^i) \right|^2}_{\mathcal{L}_{\text{bc}}(\theta)} + \underbrace{\frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{R}_{\text{int}}[u_\theta](t_r^i, \mathbf{x}_r^i) \right|^2}_{\mathcal{L}_r(\theta)}. \tag{6}$$

3 Gradient Misalignment in PINNs

With multiple loss terms, a critical challenge in training PINNs is the conflict among gradients arising from them. This gradient misalignment significantly hampers training efficiency and convergence. The issue is especially severe in multi-physics problems or stiff PDE systems where multiple constraints such as the PDE residual, boundary conditions, and initial conditions must be satisfied simultaneously.

The two distinct types of gradient conflicts [8]:

- **Type I:** This occurs when gradients from different loss terms are directionally aligned but have vastly different magnitudes. In this case, smaller gradients may be overshadowed by dominant ones, leading the optimizer to focus excessively on a subset of constraints while neglecting others.
- **Type II:** Here, the gradients have comparable magnitudes but point in op-

posing directions. This destructive interference prevents meaningful progress toward satisfying all physical constraints and results in erratic optimisation trajectories.

Both types of conflicts degrade training performance. Type I leads to loss term dominance, where only a few constraints are enforced, while Type II causes oscillations or divergence in parameter updates, particularly during the early stages of training.

Traditional gradient-based optimisation methods, such as Adam, typically assume a unified descent direction. However, in PINNs, different physics-based loss components may push the parameter updates in conflicting directions. Without accounting for these misalignments, the optimizer will likely satisfy some constraints while ignoring others, suffer from poor convergence rates, and fail to find meaningful solutions.

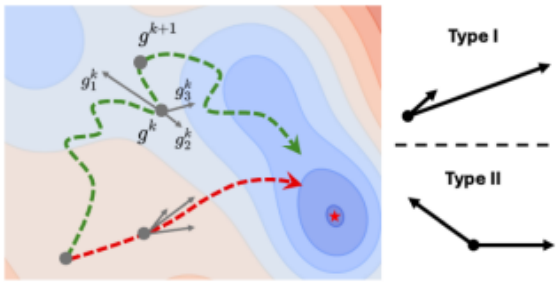


Figure 1: Type I: The irregular green trajectory illustrates how the optimisation struggles when facing two types of gradient conflicts. Type II: The red trajectory shows how appropriate preconditioning through Second-order information could mitigate these conflicts by aligning gradients both within and between optimisation steps.

4 PDE-Aware Optimizer

To resolve the gradient misalignment issues, we introduce a variant of Adam whose second–moment accumulator is driven by per-sample physics-residual gradients. We

compute the variance of the per-sample PDE gradients and divide each momentum component by the square root of that variance. Weights whose residual gradients fluctuate strongly across collocation points, therefore, take smaller steps, while those with consistent gradients move farther, giving the optimizer a built-in sensitivity to stiff regions, sharp fronts, or other difficult parts of the solution. Because the first-moment term is the batch-average of those same residual gradients, the update direction is aligned with the dominant physics signal and is less disturbed by initial- or boundary-condition terms that may point elsewhere.

For a mini-batch of B collocation points $\{x_i\}_{i=1}^{B}$ we compute

$$\boldsymbol{g}_i \;=\; \nabla_{\boldsymbol{w}}\, R_{\mathrm{pde}}(x_i;\boldsymbol{w}_t), \qquad \overline{\boldsymbol{g}} \;=\; \frac{1}{B}\sum_{i=1}^{B}\boldsymbol{g}_i. \tag{7}$$

$$\boldsymbol{m}_t = \beta_1\,\boldsymbol{m}_{t-1} + (1-\beta_1)\,\overline{\boldsymbol{g}}, \tag{8}$$

$$\boldsymbol{v}_t = \beta_2\,\boldsymbol{v}_{t-1} + \frac{1-\beta_2}{B}\sum_{i=1}^{B}\boldsymbol{g}_i^{\odot 2}, \tag{9}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\,\frac{\boldsymbol{m}_t}{\sqrt{\boldsymbol{v}_t}+\epsilon}, \tag{10}$$

where $\odot$ denotes the element-wise square. Because each coordinate $m_t^{(j)}$ is scaled by $\big(\sqrt{v_t^{(j)}}+\epsilon\big)^{-1}$, the step size is *smaller* where the residual gradient varies strongly across the batch and *larger* where it is coherent, focusing adaptation on difficult physics regions rather than on the total loss.

Algorithm 1 PDE-Aware Optimizer for training PINNs

Require: initial parameters $\boldsymbol{w}_0$, learning rate η, decay rates β_1, β_2, batch size B, small constant ϵ

Ensure: trained parameters $\boldsymbol{w}_T$

$\boldsymbol{m}_{-1} \leftarrow \mathbf{0}$ ▷ first moment

$\boldsymbol{v}_{-1} \leftarrow \mathbf{0}$ ▷ second moment

for $t = 0$ **to** $T - 1$ **do**

  for all x_i in mini-batch **do**

    $\boldsymbol{g}_i \leftarrow \nabla_{\boldsymbol{w}} R_{\text{pde}}(x_i; \boldsymbol{w}_t)$

  end for

  $\overline{\boldsymbol{g}} \leftarrow \dfrac{1}{B} \displaystyle\sum_{i=1}^{B} \boldsymbol{g}_i$

  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\, \overline{\boldsymbol{g}}$

  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + \dfrac{1 - \beta_2}{B} \displaystyle\sum_{i=1}^{B} \boldsymbol{g}_i^{\odot 2}$

  $\tilde{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t \,/\, \left(\sqrt{\boldsymbol{v}_t} + \epsilon\right)$

  $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta\, \tilde{\boldsymbol{m}}_t$

end for

5 Experimental Setup

Following common practice in PINNs literature, we benchmark the proposed Physics-Aware Optimizer on three canonical 1-D PDEs: Burgers' convection–diffusion equation, the Allen–Cahn reaction–diffusion equation and the Korteweg–De Vries wave equation.

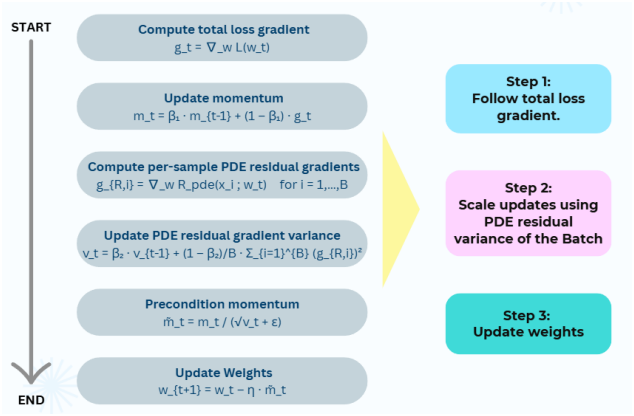


Figure 2: Flow chart of the **PDE-Aware Optimizer**. The first moment $\boldsymbol{m}_t$ is built from batch-averaged PDE-residual gradients; the second moment $\boldsymbol{v}_t$ tracks their element-wise variance, so the pre-conditioned step $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\, \boldsymbol{m}_t/(\sqrt{\boldsymbol{v}_t} + \epsilon)$ automatically shrinks learning rates in stiff regions and enlarges them where the residual is smooth.

PDE benchmarks: **Burgers** , **Allen–Cahn** and **Korteweg–De Vries** on the common domain $(x, t) \in [-1, 1] \times [0, 1]$. The interior residual minimised by the PINN is

$$\mathcal{R}(x, t; \boldsymbol{w}) = \begin{cases} \partial_t u_\theta + u_\theta\, \partial_x u_\theta - \nu\, \partial_{xx} u_\theta, & \text{Burgers,} \\ \partial_t u_\theta - \epsilon\, \partial_{xx} u_\theta - f(u_\theta), & \text{Allen–Cahn,} \\ \partial_t u_\theta + u_\theta\, \partial_x u_\theta + \mu\, \partial_{xxx} u_\theta, & \text{Korteweg–de Vries (KdV).} \end{cases}$$

Optimizer benchmark: Standard Adam, the second-order SOAP, and PDE-aware Optimizer

Model: A single MLP: input $(x, t) \in R^2$; hidden 3×64 `tanh` layers; output $u_\theta(x, t) \in R$

Collocation sampling. We generate training data as follows:

- **Interior (PDE residual):** $N_{\text{int}} = 10{,}000$ collocation points sampled uniformly over the spatio-temporal domain $(x, t) \in [x_{\min}, x_{\max}] \times [t_{\min}, t_{\max}]$.
- **Initial condition:** $N_{\text{ic}} = 1{,}000$ points sampled along $t = t_{\min}$ with $x \in [x_{\min}, x_{\max}]$ evenly spaced.
- **Boundary condition:** $N_{\text{bc}} = 1{,}000$ points sampled with $t \in [t_{\min}, t_{\max}]$, and $x = x_{\min}$ or $x = x_{\max}$ fixed on either boundary.

Training: **Batch size** $B = 1{,}024$; **epochs** $T = 10{,}000$.

Evaluation Metric: Relative L^2 error on a 400×400 grid:

$$\text{Rel-}L^2 = \left(\frac{\sum (u_\theta - u_{\text{exact}})^2}{\sum u_{\text{exact}}^2} \right)^{1/2}.$$

Hyper-parameter tuning: We carried out a full $2 \times 2 \times 2$ grid search over the learning-rate $\eta \in \{10^{-3}, 10^{-4}\}$ and moments $\beta_1 \in \{0.9, 0.99\}$, $\beta_2 \in \{0.99, 0.999\}$ for our PDE-aware optimizer on the 1-D Burgers equation. We chose to begin at the default values of the Adam optimizer and vary the hyperparameters from there. The rationale behind the choice for β_2 was to increase $1 - \beta_2$ since that term would determine how much the corresponding residual term would be weighted in v_t. Figure 3 visualizes the validation loss surface for the eight configurations. The best setting—$\eta = 10^{-3}$, $\beta_1 = 0.99$, $\beta_2 = 0.99$—achieved an average validation loss of 4.32×10^{-2}, with the two closest neighbors ($\eta = 10^{-3}, \beta_1 = 0.99, \beta_2 = 0.999$ and $\eta = 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.99$) trailing by less than 3×10^{-4}.

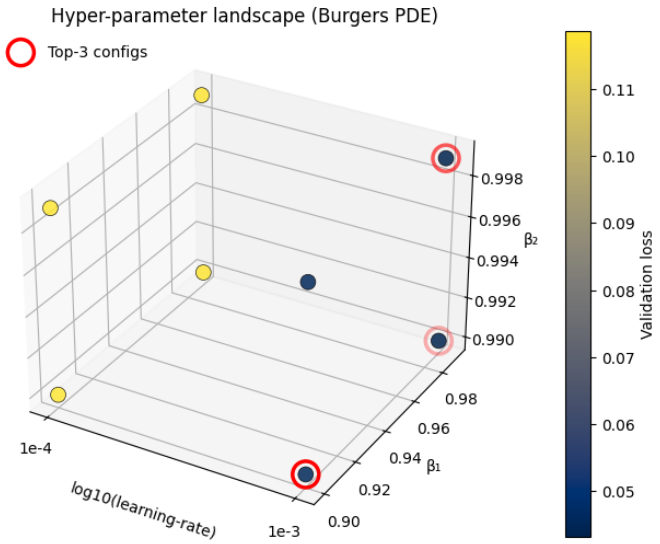
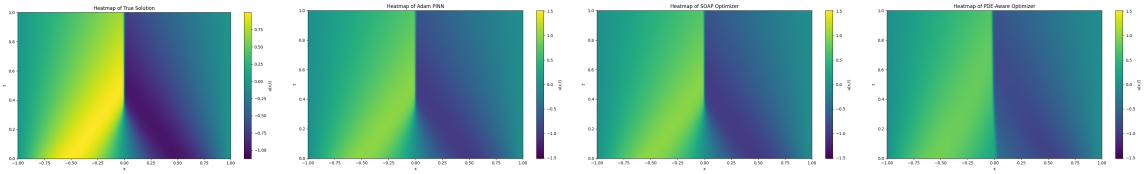


Figure 3: Validation-loss landscape over the (η, β_1, β_2) grid for Burgers PDE.

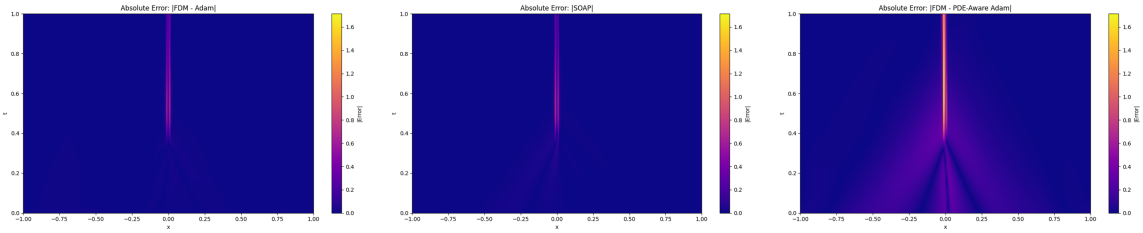
6 Results

6.1 1D Burgers PDE



(a) True (b) Adam (c) SOAP (d) PDE-aware

Figure 4: Heatmap comparison of Adam, SOAP, and PDE-aware optimizers on the Burgers' equation



(a) Adam (b) SOAP (c) PDE-aware

Figure 5: Absolute error ($|u_{\text{PINN}} - u_{\text{FDM}}|$) for Burgers' equation across different optimizers

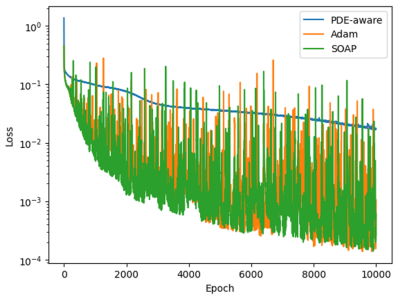
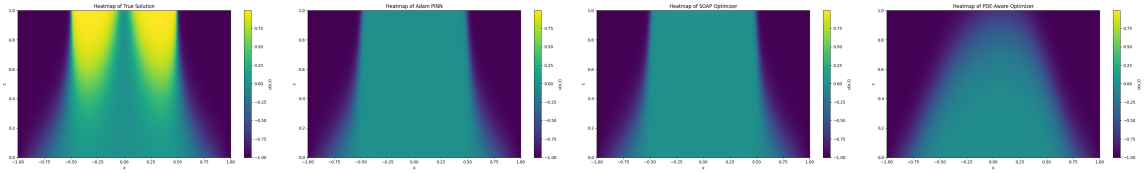


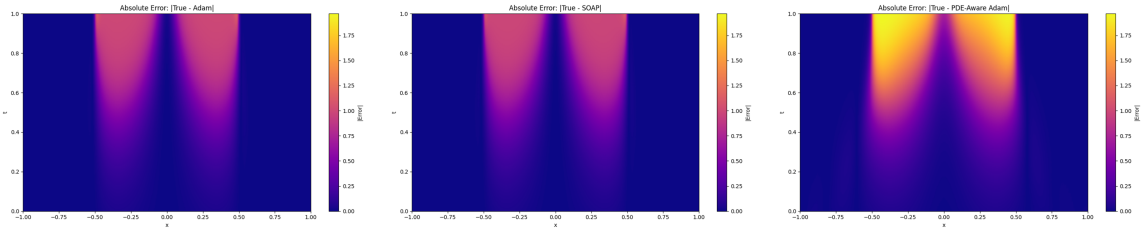
Figure 6: Training loss over epochs for Burgers' equation. PDE-aware optimizer shows smooth convergence, while Adam and SOAP exhibit faster but more oscillatory behavior.

6.2 1D Allen–Cahn PDE



(a) True (b) Adam (c) SOAP (d) PDE-aware

Figure 7: Heatmap comparison of Adam, SOAP, and PDE-aware optimizers on the Allen–Cahn equation



(a) Adam (b) SOAP (c) PDE-aware

Figure 8: Absolute error ($|u_{\text{PINN}} - u_{\text{FDM}}|$) for Allen–Cahn equation across different optimizers

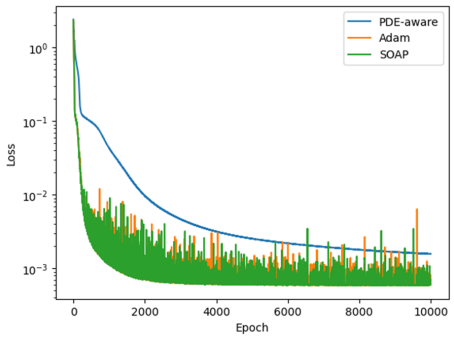
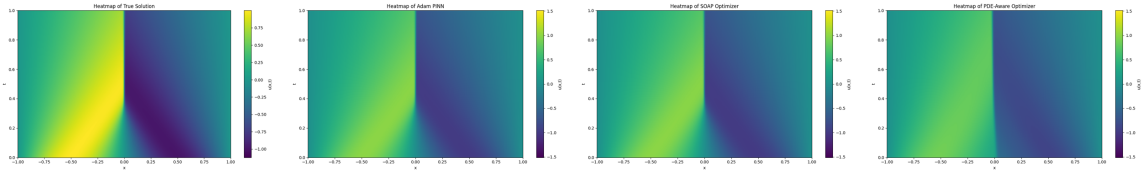


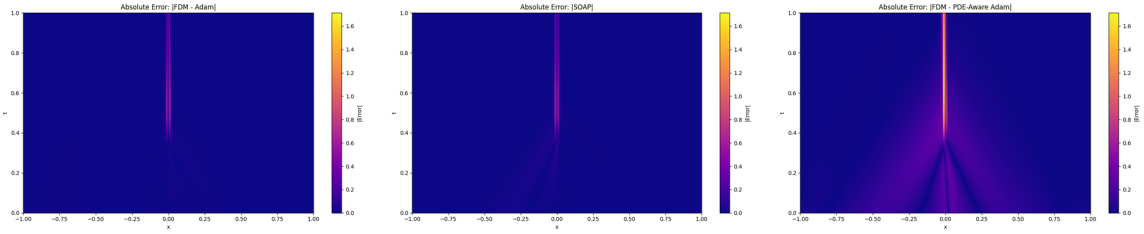
Figure 9: Training loss over epochs for Allen–Cahn equation. PDE-aware optimizer converges smoothly, while Adam and SOAP show faster but noisier behavior.

6.3 1D Korteweg–de Vries (KdV) PDE



(a) True (b) Adam (c) SOAP (d) PDE-aware

Figure 10: Heatmap comparison of Adam, SOAP, and PDE-aware optimizers on the KdV equation



(a) Adam (b) SOAP (c) PDE-aware

Figure 11: Absolute error ($|u_{\text{PINN}} - u_{\text{FDM}}|$) for KdV equation across different optimizers

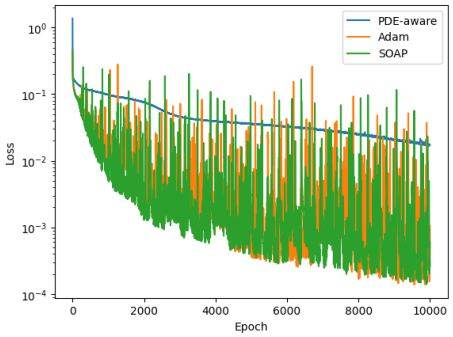


Figure 12: Training loss over epochs for KdV equation. PDE-aware optimizer shows smoother convergence, while Adam and SOAP demonstrate faster but more oscillatory descent.

7 Discussion

The comparative analysis of Adam, SOAP, and our proposed PDE-aware optimizer across the three benchmark PDEs Burgers, Allen-Cahn, and KdV highlights distinct trade-offs in optimizer behavior. Adam and SOAP demonstrate faster convergence in terms of training loss but exhibit pronounced oscillations during training. In contrast, the PDE-aware optimizer converges more gradually, yet consistently, with significantly smoother loss curves. Qualitatively, the PDE-aware optimizer produces solutions that closely track the true PDE behavior across both benchmarks, particularly in stiff or sharply varying regions. This is further supported by the absolute error plots, where the PDE-aware optimizer yields lower and more uniformly distributed errors compared to its counterparts. While SOAP shows competitive accuracy, it often suffers from instability near sharp gradients.

Overall, the PDE-aware optimizer offers a robust and interpretable alternative by directly adapting to residual gradient variance, which helps mitigate gradient misalignment and improves physical consistency in PINN training. Its performance is especially valuable when modeling stiff systems or when stability and smooth convergence are prioritized.

8 Limitations

Our current experiments were conducted using relatively small MLP architectures and limited GPU availability. While results on benchmark PDEs are promising, further validation on larger models and more complex multidimensional systems is needed. Future work should explore scalability and generalization performance using better GPUs and larger models.

9 Code Availability

All code and experiments used in this study are publicly available at: `https://github.com/vismaychuriwala/PDE-aware-optimzer-jax`.

References

[1] Vineet Gupta, Tomer Koren, and Yoram Singer. "Shampoo: Preconditioned Stochastic Tensor Optimization". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 1842–1850. URL: `https://proceedings.mlr.press/v80/gupta18a.html`.

[2] George Em Karniadakis et al. "Physics-informed machine learning". In: *Nature Reviews Physics* 3.6 (2021), pp. 422–440. DOI: `10.1038/s42254-021-00314-5`.

[3] James Martens. "Deep Learning via Hessian-Free Optimization". In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*. Omnipress, 2010, pp. 735–742. URL: `http://www.icml2010.org/papers/458.pdf`.

[4] James Martens and Roger Grosse. "Optimizing Neural Networks with Kronecker-Factored Approximate Curvature". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. Vol. 37. Proceedings of Machine Learning Research. 2015, pp. 2408–2417. DOI: `10.48550/arXiv.1503.05671`. URL: `https://arxiv.org/abs/1503.05671`.

[5] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. "Physics-informed neural networks: A deep-learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707. DOI: `10.1016/j.jcp.2018.10.045`.

[6] Farbod Roosta-Khorasani and Michael W. Mahoney. "Sub-Sampled Newton Methods I: Globally Convergent Algorithms". In: *arXiv preprint* (2016). DOI: 10.48550/arXiv.1601.04737. arXiv: 1601.04737. URL: https://arxiv.org/abs/1601.04737.

[7] Sifan Wang, Yujun Teng, and Paris Perdikaris. "Understanding and Mitigating Gradient Pathologies in Physics-Informed Neural Networks". In: *SIAM Journal on Scientific Computing* 43.5 (2021), A3058–A3081. DOI: 10.1137/20M1318043. URL: https://epubs.siam.org/doi/10.1137/20M1318043.

[8] Sifan Wang et al. "Gradient Alignment in Physics-Informed Neural Networks: A Second-Order Optimization Perspective". In: *arXiv preprint arXiv:2502.00604* (2025). Submitted 2 Feb 2025, 39 pages, 22 figures. DOI: 10.48550/arXiv.2502.00604. URL: https://arxiv.org/abs/2502.00604.

[9] Zhewei Yao et al. "ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning". In: *arXiv preprint* (2021). DOI: 10.48550/arXiv.2006.00719. arXiv: 2006.00719. URL: https://arxiv.org/abs/2006.00719.