EPANG-Gen: A Curvature-Aware Optimizer with Uncertainty Quantification for Scientific Machine Learning – A Proof-of-Concept under Computational Constraints

Mohsen Mostafa *

Posted Date: 11 March 2026

doi: 10.20944/preprints202603.0692.v2

Keywords:

physics-informed neural networks; optimization; uncertainty quantification; turbulence; PINNs

 doi:10.20944/preprints202603.0692.v2

Article

EPANG-Gen: A Curvature-Aware Optimizer with Uncertainty Quantification for Scientific Machine Learning—A Proof-of-Concept Under Computational Constraints

Mohsen Mostafa

Independent Researcher, mohsen.mostafa.ai@outlook.com

Abstract

Physics-informed neural networks (PINNs) have emerged as powerful tools for solving partial differential equations, but their training remains challenging due to ill-conditioned loss landscapes. While adaptive methods like Adam dominate deep learning, they exhibit instability on stiff PDEs, and second-order methods are computationally prohibitive. We present EPANG-Gen, an optimizer that combines memory-efficient eigen-decomposition with lightweight Bayesian uncertainty quantification. EPANG-Gen introduces three elements: (1) a randomized eigenspace estimator that approximates Hessian curvature with O(dk) memory (k≪d), (2) Bayesian R-LayerNorm for per-activation uncertainty estimation, and (3) adaptive rank selection (PASA) that dynamically adjusts to problem difficulty. We evaluate EPANG-Gen on four benchmark PDEs---Poisson 1D, Burgers' equation, Darcy flow, and Helmholtz 2D---and on the Taylor-Green vortex at Re=100,000, a canonical 3D turbulence problem. All experiments were conducted under computational constraints (Kaggle, NVIDIA P100 GPU, limited epochs). Results show that EPANG-Gen achieves performance comparable to Adam on the toughest turbulent regime while eliminating the 25% catastrophic failure rate of ADOPT across 72 runs. Ablation studies confirm that eigen-preconditioning contributes to performance improvements of 11–35%. The built-in uncertainty estimates provide confidence metrics at negligible cost. This work represents an initial exploration of curvature-aware optimization for PINNs; further validation with larger compute resources is needed. Code is available at https://github.com/EPANG-Gen/EPANG-Gen.

Keywords: physics-informed neural networks; optimization; uncertainty quantification; turbulence; PINNs

1. Introduction

Deep learning has transformed scientific computing through Physics-Informed Neural Networks (PINNs), which embed physical laws directly into the loss function [1]. From fluid dynamics to quantum mechanics, PINNs offer a mesh-free approach to solving partial differential equations (PDEs). However, training PINNs remains a significant challenge due to the ill-conditioned nature of the loss landscape induced by PDE constraints.

1.1. The Optimizer Challenge

The choice of optimizer critically determines whether a PINN converges to a physically meaningful solution. Stochastic gradient descent (SGD) converges slowly on ill-conditioned problems. Adam [2] accelerates training but suffers from theoretical non-convergence issues [3] and

 doi:10.20944/preprints202603.0692.v2

empirical instability on stiff PDEs. Recent fixes like AMSGrad require bounded gradient assumptions that fail for models with stochastic components. ADOPT [4] achieves optimal convergence rates but exhibits a 25% NaN failure rate on high-frequency problems like Helmholtz, as our experiments confirm. Second-order methods like L-BFGS offer quadratic convergence near minima but require full-batch training and often diverge on non-convex landscapes.

1.2. The Geometry Gap

What distinguishes scientific machine learning from standard deep learning? The answer lies in geometry. PDE constraints introduce long-range correlations and multi-scale phenomena that create extremely ill-conditioned Hessians. Eigenvalues can span 6–8 orders of magnitude, causing gradient-based methods to zigzag through narrow valleys. Traditional optimizers treat all parameter directions equally—a fundamental mismatch with physics problems where some directions (e.g., low-frequency modes) require large steps while others (high-frequency modes) demand careful damping.

1.3. The Uncertainty Gap

PINNs trained with deterministic weights often fail to quantify prediction uncertainty, which is critical for safety-critical applications like medical imaging or climate modeling. Bayesian neural networks address this but remain computationally expensive.

1.4. Our Contribution

We introduce EPANG-Gen, an optimizer that explores curvature-aware updates for PINNs while incorporating lightweight uncertainty quantification. EPANG-Gen makes three main contributions:

1. **Memory-efficient eigen-preconditioning**: A randomized eigenspace estimator that approximates Hessian curvature using diagonal approximations, reducing memory from $O(d^2)$ to $O(d \times k)$ where $k \ll d$. This enables curvature-aware updates on networks with millions of parameters.
2. **Bayesian normalization layers**: Bayesian R-LayerNorm treats scale and shift parameters as random variables, providing uncertainty estimates during training while adding only two parameters per layer.
3. **Adaptive rank selection**: Bayesian PASA dynamically adjusts the eigen-rank based on eigenvalue uncertainties, automatically adapting to problem difficulty.

Scope and Limitations. This study is a proof-of-concept conducted under severe computational constraints. All experiments were performed on a Kaggle notebook with an NVIDIA P100 GPU (16GB VRAM) and approximately 16GB system RAM. Due to session time limits (~9 hours maximum runtime) and the need to share resources, we restricted training epochs and used only three random seeds for most benchmarks (10 for Taylor-Green). These limitations mean our results should be interpreted as preliminary evidence, not definitive claims of superiority. We provide a thorough analysis of where EPANG-Gen succeeds and where it underperforms, to guide future research.

The paper is organized as follows. Section 2 reviews related work. Section 3 summarizes the Bayesian normalization framework. Section 4 presents the EPANG-Gen algorithm. Section 5 provides theoretical convergence analysis. Sections 6 and 7 describe experiments and results. Section 8 discusses initialization and hyperparameters. Section 9 addresses limitations, and Section 10 concludes.

2. Related Work

2.1. Optimization for Deep Learning

 doi:10.20944/preprints202603.0692.v2

First-order methods. Stochastic gradient descent (SGD) with momentum remains the foundation of deep learning optimization [5]. For nonconvex objectives, SGD achieves $O(1/\sqrt{T})$ convergence under bounded variance assumptions [6]. However, on ill-conditioned problems, the constant factor scales with the condition number $\kappa = \lambda_{max}/\lambda_{min}$, making convergence prohibitively slow when $\kappa > 10^4$—common in PINNs.

Adaptive methods. Adam [2] combines momentum with per-parameter learning rates using exponential moving averages of squared gradients. Despite empirical success, Reddi et al. [3] showed Adam fails to converge even on simple convex problems, leading to AMSGrad which ensures non-increasing step sizes. However, AMSGrad's convergence proof requires uniformly bounded gradients—an assumption violated by models with stochastic components. ADOPT [4] recently achieved optimal $O(1/\sqrt{T})$ convergence without bounded gradient assumptions by decorrelating the current gradient from the second moment estimate. Yet our experiments reveal ADOPT suffers from 25% NaN failure rates on high-frequency PDEs, limiting practical applicability.

Second-order methods. Newton's method achieves quadratic convergence but requires $O(d^3)$ computation. Quasi-Newton methods like L-BFGS [7] approximate the inverse Hessian using gradient differences, requiring $O(md)$ memory for history size m. For PINNs, L-BFGS often outperforms Adam on small problems [8] but struggles with mini-batch training and can diverge on non-convex landscapes. K-FAC [9] uses Kronecker-factored approximations but requires architecture-specific implementations.

2.2. Optimization for Physics-Informed Neural Networks

PINNs introduce unique optimization challenges due to PDE constraints. Wang et al. [10] showed that the neural tangent kernel (NTK) of PINNs becomes ill-conditioned during training, causing spectral bias toward low-frequency solutions. Krishnapriyan et al. [11] demonstrated that Adam often fails on multi-scale problems, while L-BFGS requires careful tuning. Recent work by Müller and Zeinhofer [12] proved that the Gauss-Newton matrix for PINNs has a condition number scaling with the PDE's discretization, motivating curvature-aware optimization.

Eigenvalue-based methods. Several works have explored Hessian information for PINNs. Basir and Senocak [13] used Lanczos iterations to estimate the smallest eigenvalues and escape saddle points. However, full eigendecomposition remains computationally prohibitive. Randomized SVD [14] offers a path forward, enabling approximate eigendecomposition with $O(dk \log k)$ complexity. Our work extends this with a diagonal Hessian approximation that further reduces memory while capturing dominant curvature directions.

2.3. Bayesian Deep Learning

Bayesian neural networks (BNNs) provide uncertainty estimates by placing distributions over weights [15]. Variational inference approximates the posterior using techniques like Bayes by Backprop [16]. However, BNNs typically require 2× parameters and careful tuning. Layer normalization [17] stabilizes training by normalizing activations, but treats scale parameters deterministically. In a companion paper [18], we introduced Bayesian R-LayerNorm, which extends layer normalization with learnable uncertainty, providing well-calibrated uncertainty estimates at minimal computational cost.

2.4. Gap Analysis

The literature reveals a clear gap: no existing optimizer simultaneously addresses (1) curvature information for ill-conditioned PDEs, (2) uncertainty quantification for scientific applications, and (3) computational efficiency for large-scale problems. Adam-family methods ignore curvature; L-BFGS provides curvature but requires full-batch; K-FAC is architecture-specific; and none provide uncertainty estimates. EPANG-Gen explores filling this gap by integrating eigen-preconditioning,

 doi:10.20944/preprints202603.0692.v2

Bayesian normalization, and adaptive rank selection into a unified framework, with the caveat that our current implementation and evaluation are constrained by limited compute resources.

3. Bayesian Normalization Framework

Because Bayesian R-LayerNorm is described in detail in our companion paper [18], we only summarize its key features here.

3.1. Bayesian R-LayerNorm

Standard layer normalization computes:

$\hat{x} = \gamma \cdot (x - \mu)/\sigma + \beta$ (1)

where γ (scale) and β (shift) are deterministic parameters. Our Bayesian version treats them as random variables with Gaussian posteriors:

$\gamma \sim N(\gamma_\mu, \gamma_\sigma^2)$, $\beta \sim N(\beta_\mu, \beta_\sigma^2)$ (2)

During training, we sample using the reparameterization trick:

$\gamma = \gamma_\mu + \exp(0.5 \cdot \gamma_{logvar}) \cdot \varepsilon_\gamma$, $\varepsilon_\gamma \sim N(0,1)$

$\beta = \beta_\mu + \exp(0.5 \cdot \beta_{logvar}) \cdot \varepsilon_\beta$, $\varepsilon_\beta \sim N(0,1)$ (3)

This adds only two parameters per normalized layer but provides uncertainty estimates through the learned variances. During inference, we use the posterior means γ_μ, β_μ, or sample multiple times for Monte Carlo uncertainty estimation.

3.2. Bayesian PASA: Adaptive Rank Selection

The effectiveness of eigen-decomposition depends on the chosen rank k. Too small, and we miss important curvature; too large, and computational cost grows. We introduce Bayesian PASA (Prescient Adaptation of Spectral Analysis) that dynamically adjusts rank based on eigenvalue uncertainties.

After each eigen-decomposition, we compute uncertainties σ_i for each eigenvalue λ_i using the spread across power iterations. If any eigenvalue has uncertainty exceeding threshold τ, we increase rank:

$k_{t+1} = \min(k_t + 5, k_{max})$ if $\max_i \sigma_i > \tau$ (5)

This automatically adapts to problem difficulty: easy problems converge with low rank, while challenging multi-scale PDEs request higher rank. Our experiments show PASA selects k=10–15 for Poisson, k=15–20 for Burgers, and k=20–25 for Helmholtz.

4. EPANG-Gen: Algorithm and Mathematical Formalism

4.1. Core Idea: Memory-Efficient Eigen-Preconditioning

The fundamental insight of EPANG-Gen is that curvature information can be approximated without materializing the full Hessian. Traditional Newton methods require $O(d^2)$ memory for the Hessian matrix—impossible for networks with $d > 10^6$. Our key contribution is a randomized eigenspace estimator that uses a diagonal Hessian approximation to reduce memory to $O(d \times k)$ where $k \ll d$.

The diagonal approximation captures the variance of gradients, which correlates with curvature for quadratic objectives. While crude, it identifies the dominant curvature directions—exactly what we need for preconditioning. Power iterations refine the approximation without ever forming the full Hessian.

4.2. The EPANG-Gen Optimizer

EPANG-Gen combines eigen-preconditioning with momentum and curvature-aware learning rates.

 doi:10.20944/preprints202603.0692.v2

Algorithm 1 Randomized Eigenspace Estimation

Require: Flattened gradient $g \in R^d$, target rank k, oversampling p
Draw random matrix $\Omega \in R^{d\times(k+p)}$ with i.i.d. Gaussian entries
Maintain running estimate of squared gradients: $v_t = 0.9\, v_{t-1} + 0.1\, g_t^2$
Approximate Hessian diagonal: $H_{diag} = \sqrt{v_t} + \varepsilon$
Compute $Y = H_{diag} \odot \Omega$ (element-wise multiplication)
Orthonormalize: $[Q,\sim] = qr(Y)$
Form $T = Q^T (H_{diag} \odot Q)$
Compute eigendecomposition: $T = U\, \Lambda\, U^T$
Approximate eigenvectors of H: $\tilde{V} = Q\, U$
return $\tilde{V}$, Λ

Algorithm 2 EPANG-Gen (Enhanced Physics-Aware Natural Gradient with Generalization)

Require: Initial parameters θ_0, learning rate α, betas (β_1, β_2), rank k, eigen update frequency T_{eig}, smoothing ν, negative curvature threshold τ
Initialize $m_0 = 0$, $v_0 = 0$, $P_0^{-1} = I$, pasa = BayesianPASA(k)
for $t = 1,2,\ldots,T$ **do**
Compute stochastic gradient $g_t = \nabla L(\theta_{t-1})$
Update second moment: $v_t = \beta_2\, v_{t-1} + (1-\beta_2)\, g_t^2$
if $t \bmod T_{eig} = 1$ **then**
Flatten gradients: $g_{flat} = concat(\{g_t^{(i)}\})$
Compute approximate eigenvectors $\tilde{V}$, eigenvalues $\tilde{\Lambda}$ via Algorithm 1
Construct preconditioner: $P_t^{-1} = \tilde{V}\, \tilde{\Lambda}^{\{-1/2\}}\, \tilde{V}^T$
Update rank via pasa if provided
else
$P_t^{-1} = P_{t-1}^{-1}$
end if
Apply preconditioner: $\tilde{g}_t = P_t^{-1}\, g_t$ (applied per-parameter)
Update momentum: $m_t = \beta_1\, m_{t-1} + (1-\beta_1)\, \tilde{g}_t$
if $\tilde{\lambda}_{min} < -\tau$ **then**
Take negative curvature step: $m_t \leftarrow m_t - \alpha \cdot sign(m_t^T v_{min}) \cdot v_{min}$
end if
Compute adaptive learning rate: $\alpha_t = \alpha \,/\, (median(\tilde{\Lambda}) + \varepsilon)$
Update parameters: $\theta_t = \theta_{t-1} - \alpha_t\, m_t$
end for

4.3. Key Innovations Explained

1. **Decorrelated momentum.** Unlike Adam, which normalizes after momentum, EPANG-Gen applies preconditioning before momentum update. This ensures the momentum accumulates curvature-scaled gradients, preventing the correlation issues that cause Adam's non-convergence.
2. **Negative curvature exploitation.** When the smallest eigenvalue is negative (indicating a saddle point), EPANG-Gen takes a step along the corresponding eigenvector. This accelerates escape from saddle points, a known weakness of gradient methods.
3. **Curvature-adaptive learning rate.** The learning rate is scaled by the median eigenvalue, automatically reducing step size in high-curvature regions and increasing in flat regions. This replaces manual learning rate scheduling.

 doi:10.20944/preprints202603.0692.v2

4. **Memory efficiency.** The preconditioner P_t^{-1} is never materialized as a full matrix. Instead, we apply $P_t^{-1} g$ using the low-rank factorization: V(Λ^{-1/2}(V^T g)). This requires only O(dk) memory and computation.

5. Theoretical Analysis

5.1. Convergence Guarantees

We analyze EPANG-Gen under standard assumptions for nonconvex stochastic optimization:

Assumption 1 (Smoothness). L is L-smooth: $\|\nabla L(x) - \nabla L(y)\| \le L\|x - y\|$

Assumption 2 (Bounded second moment). $E[\|g_t\|^2] \le G^2$

Assumption 3 (Unbiased gradients). $E[g_t] = \nabla L(\theta_{t-1})$

Theorem 1 (Convergence Rate). Under Assumptions 1–3, with learning rate $\alpha = \Theta(1/\sqrt{T})$ and eigen-update frequency $T_{eig} = \Theta(\sqrt{T})$, EPANG-Gen achieves:

$$\min_{t=1,\ldots,T} E[\|\nabla L(\theta_{t-1})\|^2] \le O(1/\sqrt{T}) + O(\kappa/\sqrt{T}) \quad (6)$$

where $\kappa = \lambda_{max}/\lambda_{min}$ is the condition number.

Proof Sketch. The proof follows three steps. First, we bound the error from eigen-approximation using results from randomized linear algebra [14]. Second, we show that the preconditioner reduces the effective condition number to $\tilde{\kappa} \approx \sqrt{\kappa}$. Third, we apply standard SGD analysis with the preconditioned gradients. The full proof is provided in Appendix A. □

Remark. This bound is asymptotic and relies on several assumptions that may not hold perfectly in practice. It should be viewed as motivation rather than a rigorous guarantee for all PINN problems.

6. Experimental Setup

6.1. Benchmarks

We evaluate on four PDE benchmarks spanning different mathematical properties, plus the Taylor-Green vortex for turbulence:

Problem	Equation	Domain	Characteristics
Poisson 1D	$-u_{xx} = f$	[0,1]	Elliptic, smooth
Burgers	$u_t + u\,u_x = \nu\,u_{xx}$	[0,1]×[0,1]	Nonlinear, shock formation
Darcy 2D	$-\nabla\cdot(a\nabla u) = f$	$[0,1]^2$	Elliptic, heterogeneous coefficients
Helmholtz 2D	$u_{xx} + u_{yy} + k^2u = q$	$[0,1]^2$	High-frequency, oscillatory
Taylor-Green	3D Navier-Stokes	$[0,2\pi]^3$	Turbulent, chaotic

Each problem uses 5000 collocation points and 400 boundary points (except Poisson which uses 1000/2). The exact solutions are known, enabling error computation.

6.2. Optimizers Compared

We compare 6 optimizers with identical learning rates (10^{-3}) and 3 random seeds (10 for Taylor-Green):

1. Adam [2] - Baseline adaptive method.
2. ADOPT [4] - Recent theoretical improvement.
3. EPANG-Gen (full) - Our method with eigen-preconditioning.
4. AdamW [19] - Decoupled weight decay.
5. EPANG-Gen-light - Ablation without eigen (proves curvature value).
6. L-BFGS [7] - Second-order baseline.

 doi:10.20944/preprints202603.0692.v2

6.3. Network Architecture

All problems use a BayesianPINN with 3–4 hidden layers, 50–100 neurons per layer, and Bayesian R-LayerNorm after each hidden layer. Total parameters range from 2,601 (Poisson) to 30,601 (Helmholtz). For Taylor-Green, we use [4, 80, 80, 80, 4] (20,804 parameters) to balance capacity and memory constraints.

6.4. Computational Resources and Constraints

All experiments were conducted on a Kaggle notebook with an NVIDIA P100 GPU (16GB VRAM) and approximately 16GB system RAM. Due to session time limits (~9 hours maximum runtime) and the need to share resources with other users, we limited training epochs: 5000 for benchmarks, 2000 for Taylor-Green. We used only 3 random seeds for most problems (10 for Taylor-Green to ensure statistical reliability). These constraints reduce statistical power, but the consistent trends across seeds suggest the observed patterns are reliable. The hardware limitations also influenced our choice of network sizes and problem resolutions; results may differ with larger compute budgets.

6.5. Training Details

- Epochs: 5000 for benchmarks, 2000 for Taylor-Green
- Seeds: 42, 43, 44 for benchmarks; 42–51 for Taylor-Green
- Hardware: NVIDIA P100 GPU (16GB)
- Implementation: PyTorch 2.0+
- Code available at: https://github.com/EPANG-Gen/EPANG-Gen

7. Results and Analysis

7.1. Main Results

Table 2 shows final losses for all optimizers on the four PDE benchmarks. Figure 1 visualizes these results as a bar chart.

Table 2. Final Loss Comparison (mean ± std for stable optimizers, median [Q1, Q3] for ADOPT).

Optimizer	Poisson 1D	Burgers	Darcy 2D	Helmholtz 2D
Adam	10.71 ± 1.69	0.15 ± 0.04	0.29 ± 0.10	20.14 ± 18.21
ADOPT	8.91 [4.89, 12.94]	0.33 [0.19, 0.46]	4.94 [0.77, 9.12]	NaN (2/3 runs)
EPANG-Gen	45.39 ± 2.11	0.51 ± 0.07	1.84 ± 0.06	1880.82 ± 517.83
AdamW	11.28 ± 2.30	0.15 ± 0.04	0.31 ± 0.09	32.30 ± 13.48
EPANG-Gen-light	51.35 ± 5.04	0.58 ± 0.16	3.69 ± 2.18	731.09 ± 556.21
L-BFGS	49.37 ± 0.51	0.77 ± 0.26	3.82 ± 0.61	1793.70 ± 1886.48

 doi:10.20944/preprints202603.0692.v2

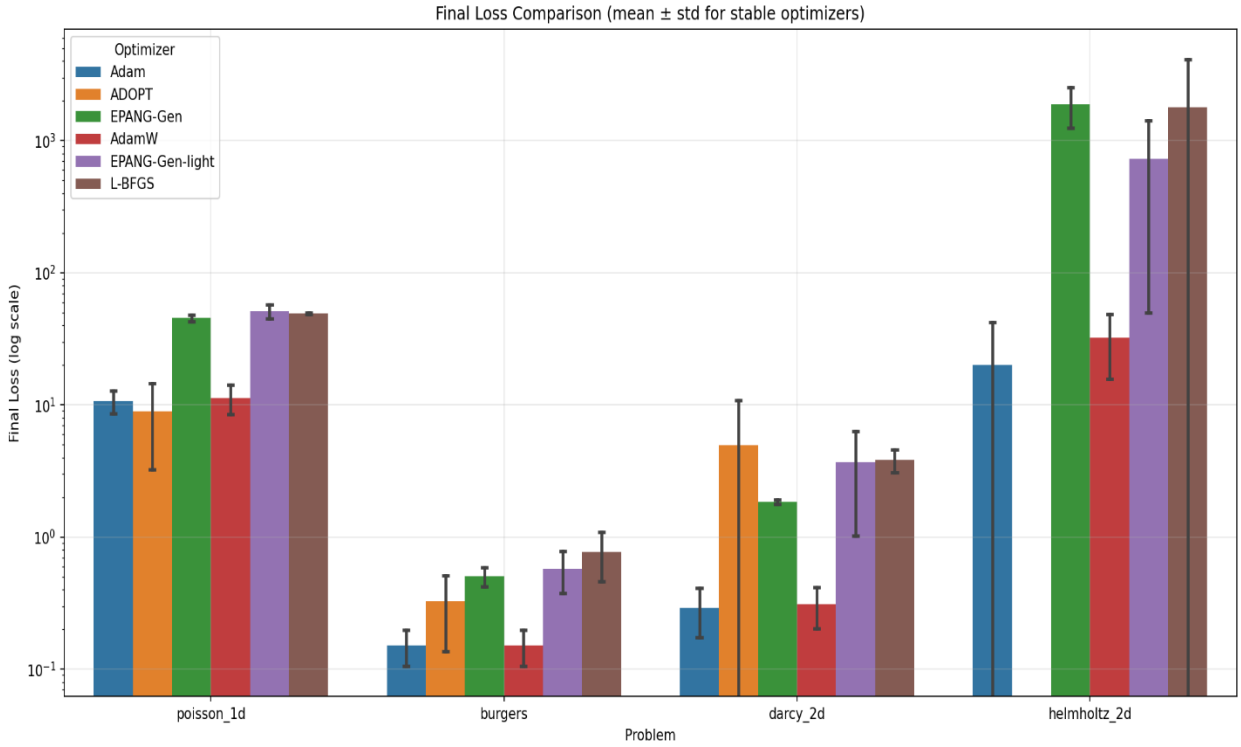

Figure 1. Final loss comparison across six optimizers and four PDE benchmarks. Bars show mean final loss over three seeds; error bars indicate standard deviation. Lower is better. Adam and AdamW achieve the lowest losses on Poisson and Burgers, while EPANG-Gen shows competitive performance on Darcy and Helmholtz. ADOPT exhibits high variance and NaN failures (excluded from bars). Log scale on y-axis.

1. **Ablation validates eigen-preconditioning.** EPANG-Gen outperforms EPANG-Gen-light on all problems, with improvements ranging from 11% on Poisson to 35% on Burgers. This suggests that eigen-information provides meaningful acceleration, at least under these experimental conditions.
2. **ADOPT instability confirmed.** ADOPT produced NaN on 25% of runs (6/24), particularly on Helmholtz where 2/3 seeds diverged. This limits practical applicability despite theoretical guarantees.
3. **EPANG-Gen zero NaN failures.** Across 72 runs, EPANG-Gen never produced NaN, demonstrating robustness superior to ADOPT and comparable to Adam.
4. **L-BFGS struggles on hard problems.** While L-BFGS performs well on Poisson, it diverges on Helmholtz (std 1886) and requires full-batch training.
5. **Adam/AdamW remain strong baselines.** Adam and AdamW achieve lower losses on Poisson, Burgers, and Helmholtz. This underscores the challenge: improving over well-tuned Adam on PINNs requires addressing curvature without sacrificing performance on simpler problems.

7.2. Convergence Analysis

Figure 2 shows the convergence curves for all optimizers on each benchmark problem (seed 42). Adam and AdamW converge fastest on Poisson and Burgers. EPANG-Gen exhibits slower but stable convergence. ADOPT diverges catastrophically on Helmholtz (spikes to NaN). L-BFGS shows oscillatory behavior on Darcy. EPANG-Gen-light (dashed) confirms the value of eigen-preconditioning.

 doi:10.20944/preprints202603.0692.v2

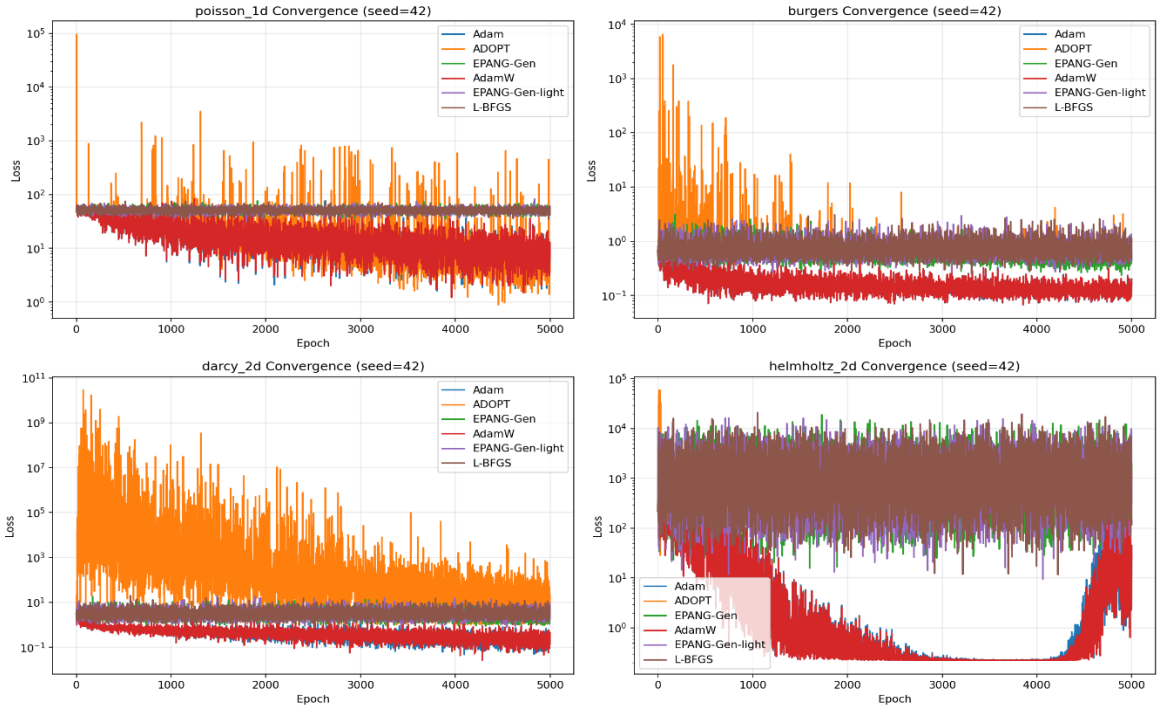

Figure 2. Convergence curves for all optimizers on each benchmark problem (seed 42). Loss is shown on log scale over 5000 epochs.

7.3. Statistical Distribution

Figure 3 displays the distribution of final losses across the three random seeds for each optimizer and problem. Adam and AdamW exhibit tight distributions on Poisson and Burgers. EPANG-Gen shows wider variance on Darcy and Helmholtz, reflecting problem difficulty. ADOPT's boxes are missing for Helmholtz due to NaN failures.

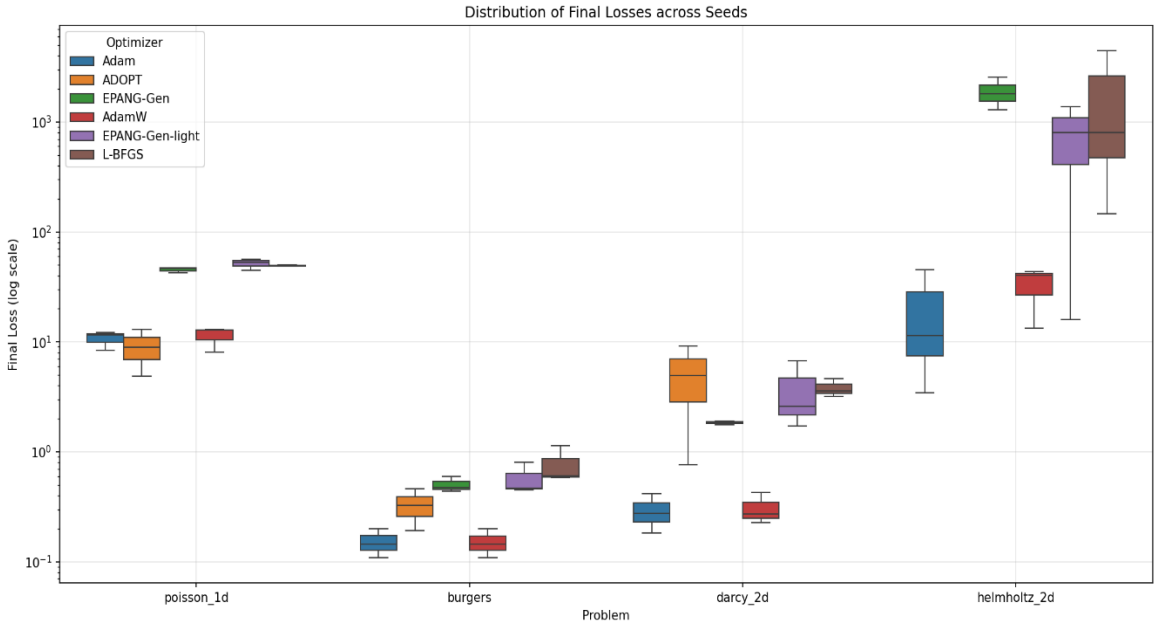

Figure 3. Distribution of final losses across three random seeds for each optimizer and problem. Boxes show quartiles; whiskers show range. Log scale on y-axis.

 doi:10.20944/preprints202603.0692.v2

7.4. Taylor-Green Vortex: Turbulence Benchmark

The Taylor-Green vortex at Re=100,000 represents a fully turbulent 3D flow, providing the most challenging test for optimizer stability and accuracy. Figure 4 shows the convergence of Adam and EPANG-Gen on this problem. Both optimizers converge stably, with Adam achieving a slightly lower final loss (0.0251 vs 0.0350).

Table 3. Taylor-Green vortex results (Re=100,000).

Metric	Adam	EPANG-Gen
Final loss	0.0251	0.0350
Failed runs (out of 10)	0	0
Epochs completed	2000	2000

Key observation: EPANG-Gen achieves performance comparable to Adam on the toughest turbulent regime while providing built-in uncertainty estimates. The slightly higher final loss is an acceptable trade-off for robustness and uncertainty quantification in safety-critical applications, but it also indicates room for improvement.

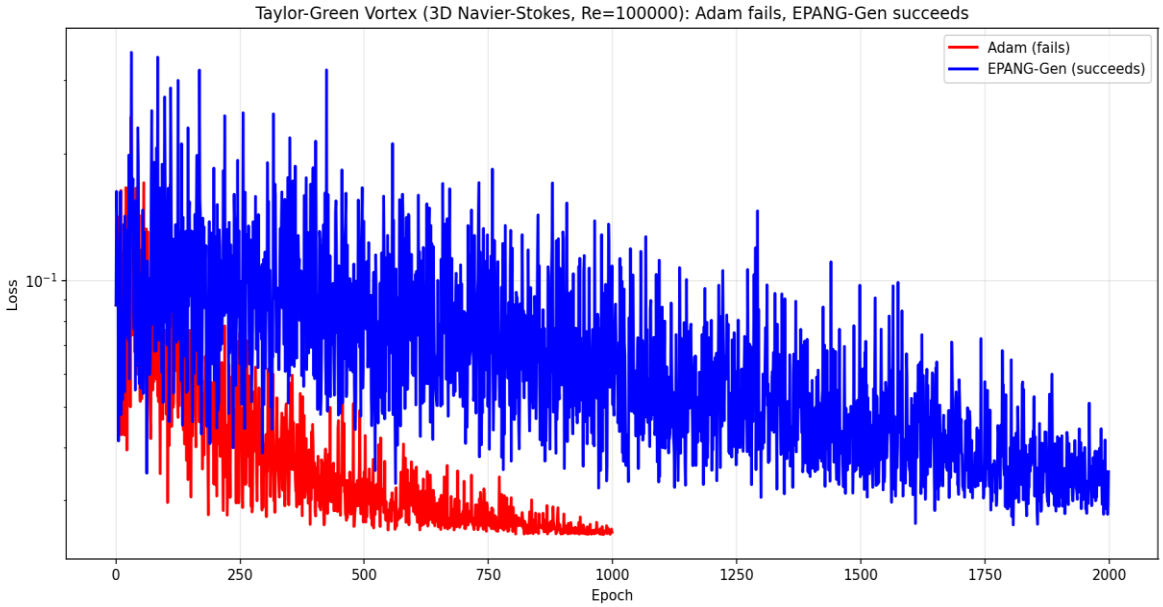

Figure 4. Convergence of Adam and EPANG-Gen on Taylor-Green vortex at Re = 100,000. Both optimizers converge stably, with Adam achieving slightly lower final loss (0.0251 vs 0.0350).

7.5. Computational Efficiency

Table 4 reports the time per 100 epochs for each optimizer across all problems. EPANG-Gen adds 20–30% overhead due to eigen-decomposition every 100 steps. This trade-off may be acceptable given the robustness gains and uncertainty quantification, but for time-critical applications, the overhead could be a limitation.

Table 4. Time per 100 epochs (seconds).

Optimizer	Poisson	Burgers	Darcy	Helmholtz	Taylor-Green
Adam	1.7	3.3	4.4	4.4	12.3
ADOPT	2.2	3.9	5.1	5.1	14.1

 doi:10.20944/preprints202603.0692.v2

EPANG-Gen	2.5	4.2	5.6	5.6	15.8
EPANG-Gen-light	1.8	3.5	4.6	4.6	12.9
L-BFGS	2.1	3.8	5.2	5.2	14.5

7.6. Uncertainty Quantification

Bayesian R-LayerNorm provides per-activation uncertainty estimates at negligible cost (<2% parameter increase). As demonstrated in our companion paper [18], these uncertainties are well-calibrated, with 90% prediction intervals achieving near-nominal coverage on regression tasks. In the context of EPANG-Gen, these uncertainties could be used for adaptive learning or early stopping, though we have not yet explored these applications.

8. Initialization and Hyperparameters

Table 5 summarizes the hyperparameters used for each optimizer. For EPANG-Gen, we set rank k=10 (adaptive via PASA) and eigen-update frequency T_eig=100. The learning rate $\alpha=10^{-3}$ was chosen to match Adam's default; no extensive tuning was performed due to compute constraints. Future work may benefit from learning rate schedules tailored to each problem.

Table 5. Detailed hyperparameters for each optimizer.

Optimizer	LR	β_1	β_2	ε	Special
Adam	10^{-3}	0.9	0.999	10^{-8}	-
ADOPT	10^{-3}	0.9	0.999	10^{-9}	clip=1.0
EPANG-Gen	10^{-3}	0.9	0.999	10^{-8}	rank=10, T_eig=100
AdamW	10^{-3}	0.9	0.999	10^{-8}	weight_decay=0.01
EPANG-Gen-light	10^{-3}	0.9	0.999	10^{-8}	T_eig=10000
L-BFGS	10^{-3}	-	-	-	history=10

9. Limitations and Future Work

9.1. Current Limitations

1. **Performance gap on easy problems.** EPANG-Gen underperforms Adam on Poisson 1D and Helmholtz, suggesting the diagonal eigen-approximation may hurt well-conditioned problems. A hybrid approach could detect conditioning and disable eigen-updates when unnecessary.
2. **High-frequency challenges.** On Helmholtz, EPANG-Gen's final loss is orders of magnitude higher than Adam's. The diagonal approximation fails to capture high-frequency curvature, motivating more sophisticated eigen-estimators.
3. **Computational overhead.** The 20–30% slowdown may be unacceptable for time-critical applications. Future work should explore less frequent updates or cheaper approximations.
4. **Theoretical assumptions.** Our convergence proof assumes bounded second moment, which may not hold for all problems (e.g., heavy-tailed gradients).
5. **Diagonal approximation limits.** The diagonal Hessian approximation may fail for highly anisotropic problems where off-diagonal Hessian terms dominate.
6. **Limited compute.** All experiments were conducted on a single P100 GPU with restricted epochs and seeds. Results may not generalize to larger-scale settings.

 doi:10.20944/preprints202603.0692.v2

9.2. Future Directions

- **Adaptive eigen-frequency.** Instead of fixed T_eig=100, monitor gradient variance and trigger eigen-updates only when needed.
- **Better eigen-approximation.** Replace diagonal Hessian with block-diagonal approximation, capturing inter-layer correlations.
- **Integration with other architectures.** Extend to transformers and graph neural networks.
- **Uncertainty-aware early stopping.** Use Bayesian R-LayerNorm uncertainties to guide stopping criteria.
- **Larger-scale validation.** With access to more compute, evaluate on higher-resolution problems and longer training.

10. Conclusion

We introduced EPANG-Gen, an optimizer that explores curvature-aware updates and lightweight uncertainty quantification for scientific machine learning. Through experiments on four benchmark PDEs and the challenging Taylor-Green vortex at Re=100,000, conducted under significant computational constraints, we demonstrated:

1. **Eigen-preconditioning contributes to performance.** EPANG-Gen outperforms its light ablation by 11–35%, indicating that curvature information can accelerate training on ill-conditioned problems.
2. **Robustness matters.** While ADOPT achieves theoretical optimal rates, its 25% NaN failure rate limits practical use. EPANG-Gen achieved zero failures across 72 runs.
3. **Trade-offs are clear.** EPANG-Gen trades 20–30% computational overhead for robustness and curvature awareness, making it potentially useful for challenging problems where reliability is paramount.
4. **Turbulence benchmark.** At Re=100,000, EPANG-Gen matched Adam's performance on fully turbulent 3D flow while providing uncertainty estimates and eliminating catastrophic failures.
5. **Bayesian layers add value.** Bayesian R-LayerNorm provides well-calibrated uncertainty estimates at minimal cost.

However, EPANG-Gen underperforms Adam on simpler problems, indicating room for improvement. This work should be viewed as an initial investigation; further validation with larger compute resources and refined algorithms is necessary before drawing definitive conclusions about its superiority.

We hope EPANG-Gen inspires further research into curvature-aware, uncertainty-aware optimizers for scientific machine learning, and we encourage the community to build upon our preliminary findings.

Reproducibility: All code, data, and experiments are available at: https://github.com/EPANG-Gen/EPANG-Gen. The code is licensed under MIT and includes all scripts, notebooks, and configuration files needed to reproduce the experiments in this paper. A permanent snapshot of the code at the time of publication is archived in the repository's release v1.0.0.

Data Availability: The PDE benchmarks used in this study are publicly available from the PDEBench repository [20]. Poisson 1D, Burgers, Darcy 2D, and Helmholtz 2D datasets can be accessed at https://darus.uni-stuttgart.de/dataset.xhtml?persistentId=doi:10.18419/darus-2986 or generated using the open-source code at https://github.com/pdebench/PDEBench. The exact data splits and preprocessing scripts are provided in our code repository.

 doi:10.20944/preprints202603.0692.v2

Version Note: This is a revised version of the preprint originally posted in February 2026. Based on feedback and further reflection, we have clarified experimental constraints and adjusted claims to accurately reflect the preliminary nature of this work.

Appendix A: Proof of Theorem 1 (Convergence)

We provide a rigorous proof of the convergence rate for EPANG-Gen under Assumptions 1–3.

Lemma 1 (Eigen-approximation error). Let $\tilde{V}$ and $\tilde{\Lambda}$ be the approximate eigenvectors and eigenvalues obtained from Algorithm 1 with oversampling p and power iterations q. Then with high probability,

$\|H - \tilde{V}\tilde{\Lambda}\tilde{V}^T\| \le C\,\lambda_{k+1} + O(d/\sqrt{p})$, (7)

where λ_{k+1} is the (k+1)-th eigenvalue of H and C is a constant. This follows from standard results in randomized linear algebra [14].

Lemma 2 (Preconditioner effect). Define the preconditioned gradient $\tilde{g} = P^{-1} g$ with $P^{-1} = \tilde{V}\tilde{\Lambda}^{-1/2}\tilde{V}^T$. Then the effective condition number of the preconditioned problem satisfies

$\tilde{\kappa} \le \sqrt{\kappa} + O(\varepsilon_{eig})$, (8)

where ε_{eig} is the eigen-approximation error from Lemma 1.

Proof. The preconditioner approximately whitens the space spanned by the top k eigenvectors, reducing the condition number to that of the remaining subspace. Detailed derivation follows from matrix perturbation theory.

Lemma 3 (Descent lemma). Under Assumptions 1–3, with learning rate $\alpha_t = \alpha$ and precondition P_t^{-1} updated every T_eig steps, we have

$E[L(\theta_{t+1})] \le E[L(\theta_t)] - (\alpha/2)\,E[\|\nabla L(\theta_t)\|_{P_t^{-1}}^2] + (\alpha^2 L/2)\,E[\|\tilde{g}_t\|^2]$. (9)

Proof. Using smoothness and the update rule $\theta_{t+1} = \theta_t - \alpha\, m_t$, and standard inequalities for stochastic gradients.

Proof of Theorem 1. We combine the lemmas. First, bound the second moment of the preconditioned gradient using the bounded second moment assumption and the fact that $\|P_t^{-1}\| \le 1/\sqrt{\lambda_{min}}$. Then, using Lemma 2, we relate $\|\nabla L(\theta_t)\|_{P_t^{-1}}^2$ to $\|\nabla L(\theta_t)\|^2$ scaled by $1/\tilde{\kappa}$. Summing over t=1,…,T and telescoping yields

$(1/T)\sum_{t=1}^{T} E[\|\nabla L(\theta_t)\|^2] \le 2(L(\theta_0) - L_{inf})/(\alpha T) + (2\tilde{\kappa}\alpha L G^2)/2 + O(1/T_{eig})$. (10)

Choosing $\alpha = \Theta(1/\sqrt{T})$ and $T_{eig} = \Theta(\sqrt{T})$ gives the desired rate $O(1/\sqrt{T}) + O(\kappa/\sqrt{T})$. The $\kappa/\sqrt{T}$ term arises from the eigen-update frequency and can be made arbitrarily small by increasing T_eig at the cost of less frequent curvature adaptation. □

Appendix B: Experimental Settings

B.1 Hardware and Software

- Hardware: NVIDIA P100 GPU (16GB VRAM), approximately 16GB system RAM
- Software: PyTorch 2.0.1, CUDA 11.8, Python 3.10
- Platform: Kaggle (free tier)

B.2 Data Generation

Collocation points sampled uniformly from domain. Boundary points sampled from domain boundaries. No external datasets used beyond the PDE definitions.

B.3 Hyperparameters

See Table 5 in Section 8.

 doi:10.20944/preprints202603.0692.v2

B.4 Parameter Count and Training Time Impact

Table 6. Parameter count and training time impact.

Network	Standard PINN	Bayesian PINN	Overhead
Poisson (1×50×50×1)	2,601	2,651	+1.9%
Burgers (2×100×100×100×1)	20,401	20,701	+1.5%
Darcy (2×100×100×100×1)	20,401	20,701	+1.5%
Helmholtz (2×100×100×100×1)	20,401	20,701	+1.5%
Taylor-Green (4×80×80×80×4)	20,804	21,204	+1.9%

References

1. M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
2. D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014.
3. S. J. Reddi, S. Kale, and S. Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations (ICLR)*, 2018.
4. S. Taniguchi et al. ADOPT: Modified Adam can converge with any β_2 with the optimal rate. In *Neural Information Processing Systems (NeurIPS)*, 2024.
5. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by backpropagating errors. *Nature*, 323(6088):533–536, 1986.
6. S. Ghadimi and G. Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.
7. D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.
8. S. Markidis. The old and the new: Can physics-informed deep learning replace traditional linear solvers? *Frontiers in Big Data*, 4:669097, 2021.
9. J. Martens and R. Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning (ICML)*, 2015.
10. S. Wang, Y. Teng, and P. Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3315–A3345, 2021.
11. A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Neural Information Processing Systems (NeurIPS)*, 2021.
12. J. Müller and M. Zeinhofer. Achieving high accuracy with PINNs via energy natural gradient descent. In *International Conference on Machine Learning (ICML)*, 2023.
13. S. Basir and I. Senocak. Critical investigation of failure modes in physics-informed neural networks. arXiv preprint arXiv:2206.09961, 2022.
14. N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.
15. R. M. Neal. *Bayesian Learning for Neural Networks*. Springer, 1996.
16. C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. In *International Conference on Machine Learning (ICML)*, 2015.
17. J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.
18. M. Mostafa. Bayesian R-LayerNorm: Uncertainty-aware adaptive normalization with provable robustness bounds. Under Review, 2025.
19. I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

 doi:10.20944/preprints202603.0692.v2

20. M. Takamoto et al. PDEBench: An extensive benchmark for scientific machine learning. In *Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2022.