Building Trust in PINNs: Error Estimation through Finite Difference Methods

Aleksander Krasowski[1,*], René P. Klausen[1], Aycan Celik[1], Sebastian Lapuschkin[1,4], Wojciech Samek[1,2,3] and Jonas Naujoks[1,*]

[1]Department of Artificial Intelligence, Fraunhofer Heinrich Hertz Institute
[2]Department of Electrical Engineering and Computer Science, Technische Universität Berlin
[3]BIFOLD - Berlin Institute for the Foundations of Learning and Data
[4]Centre of eXplainable Artificial Intelligence, Technological University Dublin

Abstract

Physics-informed neural networks (PINNs) constitute a flexible deep learning approach for solving partial differential equations (PDEs), which model phenomena ranging from heat conduction to quantum mechanical systems. Despite their flexibility, PINNs offer limited insight into how their predictions deviate from the true solution, hindering trust in their prediction quality. We propose a lightweight post-hoc method that addresses this gap by producing pointwise error estimates for PINN predictions, which offer a natural form of explanation for such models, identifying not just whether a prediction is wrong, but where and by how much. For linear partial differential equations, the error between a PINN approximation and the true solution satisfies the same differential operator as the original problem, but driven by the PINN's PDE residual as its source term. We solve this error equation numerically using finite difference methods requiring no knowledge of the true solution. Evaluated on several benchmark PDEs, our method yields accurate error maps at low computational cost, enabling targeted and interpretable validation of PINNs.

Keywords

Physics-Informed Neural Networks, Finite Difference Methods, XAI4Science, Error Estimation, Post-Hoc

1. Introduction

A central aim in the physical sciences is the prediction and modeling of phenomena that arise in the natural world. Partial differential equations (PDEs) are among the most fundamental tools for this, describing how quantities evolve over space and time across domains from heat conduction to quantum mechanics. A solution to such an equation then provides a complete quantitative description of the phenomenon in question. However, closed-form solutions are available for only the simplest cases, and one must resort to numerical approximations. With the rise of artificial intelligence and ever-growing computational capabilities, neural networks have been explored as an alternative to classical numerical methods. Physics-informed neural networks (PINNs) [1] are one such approach for solving PDEs by incorporating the governing laws as prior physical knowledge into the training objective. Since their introduction, PINNs have been successfully applied across a wide range of domains [2, 3, 4].

Despite their flexibility, they are far from flawless. A growing body of research has documented systematic pitfalls arising from their composite loss formulation [5, 6, 7], as well as problems in propagating the solution from the initial condition throughout the domain [8, 9]. PINNs may hence produce incorrect solutions that are difficult to identify from the training loss alone, limiting their reliability and trustworthiness. Reliable deployment of such models therefore requires methods that can both identify where predictions fail and by how much, ideally without access to the true solution.

At the same time, interpretability has become a central concern in scientific machine learning [10], where trust in model predictions is grounded in physical consistency rather than statistical fits alone. For PINNs, such an explainable artificial intelligence (XAI) perspective remains largely absent. New models

Late-breaking work, Demos and Doctoral Consortium, colocated with the 4th World Conference on eXplainable Artificial Intelligence: July 01–03, 2026, Fortaleza, Brazil

*Corresponding author.
aleksander.krasowski@hhi.fraunhofer.de (A. Krasowski); jonas.naujoks@hhi.fraunhofer.de (J. Naujoks)

call for novel forms of explanation. In established XAI domains such as computer vision, explanations typically identify which input features drove a prediction. This paradigm does not transfer to PINNs, where inputs are spatio-temporal coordinates rather than semantically meaningful features. On the other hand, PINNs offer a benefit rarely available in other machine learning settings: the governing equation that the ground truth solution must satisfy is known by construction and embedded in the training objective. While the resulting residual quantifies local PDE violations, it does not directly reveal how far the approximation deviates from the actual ground truth solution. A method that efficiently converts these residuals into error estimates would therefore provide a natural and actionable form of explanation, analogous to attribution heat maps in computer vision, but quantifying where and by how much a prediction is wrong. In this work, we propose precisely such a method: a lightweight post-hoc approach that yields spatially and temporally resolved error maps for PINN predictions to efficiently validate and interpret model behavior. This quantitative angle on model reliability for PINNs enables practitioners to trust or reject predictions at specific locations in the domain.

We focus on linear PDEs and exploit a key property: for a PINN approximation of a linear PDE, the pointwise error between the prediction and the true solution satisfies the same differential operator as the original problem, driven by the PINN's PDE residual. We solve this error equation via finite difference methods, without knowledge of the true solution, and evaluate on five benchmark settings, demonstrating accurate error maps at low computational cost.

Related Work Prior work on PINN reliability includes both a priori guarantees [11, 12, 13] and a posteriori bounds [14, 15, 16, 17]. [15, 18] derive rigorous error bounds for PINNs, based on constants derived from the strongly continuous semigroup of the governing PDE operator. Recently, [16] extended this approach by providing a method to derive these constants using finite differences. Crucially, however, finite differences are used there only to parametrize the bound, not to integrate the error equation itself. We note that unlike [18, 19], our estimates do not constitute provable upper bounds; however, we empirically show that they accurately track the true error efficiently.

Closely related to our work, [19] and [20] also build on the same error equation driven by the PINN's PDE residual, a principle also used in a posteriori finite element error estimation [21]. [19] invert it to derive provable error bounds, though their approach is limited to ordinary differential equations and first-order PDEs. [20], on the other hand, solve the error equation stochastically via multilevel Monte Carlo to correct PINN solutions in high-dimensional settings. In contrast, we consider low-dimensional PDEs and directly integrate the defect equation numerically using finite differences, yielding cheap, pointwise error estimates, applicable for a broad class of PDEs.

2. Theoretical Background

2.1. Physics-Informed Neural Networks

PINNs [1] are a deep learning approach aimed at solving initial-boundary value problems (IBVPs) by incorporating the governing equations as constraints during training. Let $\Omega \subset \mathbb{R}^n$ denote an open, bounded, connected domain, and $\Gamma_k \subseteq \partial\Omega$ be smooth components of the boundary, where $x \in \Omega$ may be purely spatial or may include time. Then we define the IBVP by

$$\mathcal{D}[u](x) = 0, \quad x \in \Omega \tag{1}$$

$$\mathcal{B}_k[u](x) = 0, \quad x \in \Gamma_k \text{ for } k = 1, \dots, K \quad . \tag{2}$$

where $\mathcal{D}$ is a differential operator encoding the governing PDE and $\mathcal{B}$ represents the initial-boundary condition (IBC) at Γ_k, where $k = 1, \dots, K$. We say that $u : \Omega \to \mathbb{R}^d$ is a solution to the IBVP defined by $\mathcal{D}, \mathcal{B}_1, \dots, \mathcal{B}_K$, if it satisfies Eqs. (1) and (2). Note that Eq. (2) is formulated to capture both Dirichlet and Neumann conditions, among others.

To find such a solution, PINNs approximate it by a neural network $\varphi(x; \theta)$, which depends on parameters $\theta \in \Theta$. Training proceeds by minimizing a composite loss that penalizes violations of both

the governing PDE and the prescribed boundary conditions

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{X}_{\text{pde}}|} \sum_{x \in \mathcal{X}_{\text{pde}}} \|\mathcal{D}[\varphi(x;\theta)]\|_2^2 + \sum_{k=1}^{K} \frac{1}{|\mathcal{X}_{\text{bc,k}}|} \sum_{x \in \mathcal{X}_{\text{bc,k}}} \|\mathcal{B}_k[\varphi(x;\theta)]\|_2^2 \quad , \tag{3}$$

over sampled training points $\mathcal{X}_{\text{pde}} \subset \Omega$ and $\mathcal{X}_{\text{bc,k}} \subset \Gamma_k$.

A common approach to mitigating the optimization difficulties inherent to this composite loss structure is hard-constraining several or all IBCs such that they are fulfilled by design [22]. Given an IBVP with initial condition $u(x, t_0) = u_0(x)$ and Dirichlet boundary conditions $u(x_L, t) = u(x_R, t) = 0$, we can transform the PINN as

$$\varphi(x, t; \theta) = u_0(x) + (t - t_0)(x - x_L)(x - x_R)\phi(x, t; \theta) \quad , \tag{4}$$

which satisfies these conditions for any PINN ϕ. Throughout this work, we hard-constrain all IBCs, ensuring that the PDE residual is the sole source of the approximation error.

2.2. Finite Difference Methods

Unlike most machine learning models, PINNs solve problems with known PDEs, which allows for generating reference solutions via numerical methods. In the following, we introduce finite difference methods (FDM) [23], a widely used numerical scheme for approximating solutions to IBVPs, and show in Sec. 3 how this framework can be exploited to achieve post-hoc error diagnostics directly.

The core idea is to discretize the domain Ω into a grid and approximate derivatives with finite differences between neighboring grid points. For instance, given a (one-dimensional) grid with spacing Δx and using the shorthand notation $u_i = u(x_i)$, the first and second spatial derivatives can be approximated as

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} \approx \frac{u_{i+1} - u_{i-1}}{2\,\Delta x} \quad , \qquad \left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} \quad . \tag{5}$$

These so-called central differences are one of several possible finite difference schemes. Higher-order, mixed derivatives as well as alternative schemes follow analogously [23].

To illustrate the core idea, consider the one-dimensional Poisson equation on $\Omega = (x_L, x_R) \subset \mathbb{R}$ with homogeneous Dirichlet boundary conditions $u(x_L) = u(x_R) = 0$, differential operator $\mathcal{D}[u] = \frac{\partial^2 u}{\partial x^2}$, and source term f, such that $\mathcal{D}[u] - f = 0$. We discretize the domain Ω into k grid points $x_0, x_1, \ldots, x_{k-1}$ with uniform spacing Δx, where $x_0 = x_L$ and $x_{k-1} = x_R$. Approximating $\mathcal{D}$ at interior points using Eq. (5) and assembling across all grid points together with the boundary conditions yields

$$\frac{1}{\Delta x^2} \begin{bmatrix} \Delta x^2 & 0 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & \cdots & 0 & 0 \\ 0 & 1 & -2 & \cdots & 0 & 0 \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & 0 & \cdots & 0 & \Delta x^2 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{k-2} \\ u_{k-1} \end{bmatrix} - \begin{bmatrix} 0 \\ f_1 \\ f_2 \\ \vdots \\ f_{k-2} \\ 0 \end{bmatrix} = 0 \quad . \tag{6}$$

We write this compactly as $\boldsymbol{L}\boldsymbol{u} - \boldsymbol{f} = 0$, where $\boldsymbol{L}$ denotes the discretized spatial operator. For other *spatial* differential operators $\mathcal{D}$ the discretization $\boldsymbol{L}$ follows analogously.

Time-Stepping For time-dependent problems, we discretize the time interval $[0, t_{\max}]$ with spacing Δt and write $u_i^n = u(x_i, t_n)$. Rather than solving the full spatio-temporal system at once, we exploit the causal structure and march forward in time from the initial condition u^0, solving a spatial system at each step. The choice of the time-stepping scheme depends on the order of the temporal derivative. Boundary conditions are incorporated analogously to Eq. (6) and omitted below for notational clarity.

First-order Time Derivatives (Crank-Nicolson) For first-order temporal derivatives, we employ the Crank-Nicolson scheme [24], which averages the spatial operator between the current and next time level. Taking the heat equation $\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0$ as an example, this yields

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} - \frac{1}{2}\left[\frac{\alpha}{\Delta x^2}(u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}) + \frac{\alpha}{\Delta x^2}(u_{i-1}^n - 2u_i^n + u_{i+1}^n)\right] = 0 \quad . \tag{7}$$

Note that this equation involves multiple unknowns at $t = n + 1$, coupled across neighboring spatial points. This scheme is implicit, requiring solving a linear system at each time step. For any spatial operator $\boldsymbol{L}$ we can assemble the following system across all interior points

$$(\boldsymbol{I} - \tfrac{1}{2}\Delta t \boldsymbol{L})\boldsymbol{u}^{n+1} = (\boldsymbol{I} + \tfrac{1}{2}\Delta t \boldsymbol{L})\boldsymbol{u}^n \quad , \tag{8}$$

with $\boldsymbol{I}$ denoting the identity matrix and $\boldsymbol{u}^n = [u_1^n, \ldots, u_{k-2}^n]$. Crank-Nicolson is second-order accurate in both space and time, and unconditionally stable for parabolic problems [23].

Second-order Time Derivatives (Central Differences) For PDEs with second-order time derivatives, we apply central differences in time, mirroring the spatial stencil. In matrix form this becomes

$$\boldsymbol{u}^{n+1} = 2\boldsymbol{u}^n + \Delta t^2 \cdot \boldsymbol{L}\boldsymbol{u}^n - \boldsymbol{u}^{n-1} \quad . \tag{9}$$

Here the scheme only involves known variables on the right-hand side, making it explicit. This allows for quick computations; however, introduces stability constraints [23].

3. Methodology

Rather than using FDM solely to generate a reference solution, we show that for linear PDEs the same framework can be used to produce spatially resolved error maps that explain where and by how much a PINN prediction deviates from the true solution, directly from its PDE residual.

Let $\hat{\varphi} := \varphi(x; \hat{\theta})$ denote a trained, hard-constrained PINN with parameters $\hat{\theta}$, which approximates the true solution $u(x)$ of a linear PDE $\mathcal{D}[u] = 0$. The pointwise error of the PINN is $e(x) := u(x) - \hat{\varphi}(x)$ and the PDE residual of the PINN is given by $R(x) = \mathcal{D}[\hat{\varphi}](x)$. Given the linearity of the operator, applying it to the error yields the defect equation

$$\mathcal{D}[e](x) = \underbrace{\mathcal{D}[u](x)}_{=0} - \mathcal{D}[\hat{\varphi}](x) = -R(x) \quad . \tag{10}$$

Hence, the error e satisfies the same PDE as the original problem, but with the PINN residual $R = \mathcal{D}[\hat{\varphi}]$ as a source term, which we can evaluate at any desired point $x \in \Omega$. Through hard-constraining, we make sure that the initial and boundary conditions are satisfied exactly, i.e., $e(x) = 0$ for all initial and boundary points. This allows the recovery of the error e from the PDE residual alone, without knowledge of the true solution.

Integrating the Residual using FDM We solve the error equation using the FDM framework as introduced in Sec. 2.2. The discretization is structurally identical to that of the PDE. The spatial operator $\boldsymbol{L}$ and time-stepping remain the same. Two things change, however: (i) we solve for the unknown error $\boldsymbol{e}$ instead of the solution $\boldsymbol{u}$, (ii) the PINN residual $\boldsymbol{R}$ appears as a source term.

For steady-state problems, the error equation becomes $\boldsymbol{L}\boldsymbol{e} - \boldsymbol{R} = 0$, where $\boldsymbol{R}$ is the PINN residual evaluated on the discrete grid points. For time-dependent problems we march forward in time starting from $\boldsymbol{e}^0$, which is 0 due to hard-constraining.

For first-order time derivatives, the Crank-Nicolson update (Eq. (8)) yields

$$(\boldsymbol{I} - \tfrac{1}{2}\Delta t \boldsymbol{L})\boldsymbol{e}^{n+1} = (\boldsymbol{I} + \tfrac{1}{2}\Delta t \boldsymbol{L})\boldsymbol{e}^n - \tfrac{1}{2}\Delta t(\boldsymbol{R}^n + \boldsymbol{R}^{n+1}) \quad . \tag{11}$$

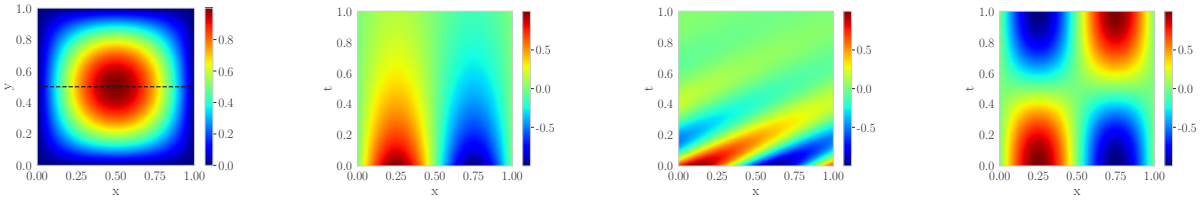
Analogously, for problems with second-order time derivatives the central difference (Eq. (9)) becomes

$$\boldsymbol{e}^{n+1} = 2\boldsymbol{e}^n + \Delta t^2 \cdot (\boldsymbol{L}\boldsymbol{e}^n - \boldsymbol{R}^n) - \boldsymbol{e}^{n-1} \quad . \tag{12}$$

Code is available at https://github.com/aleks-krasowski/pinn-fdm-error-estimation.

Table 1
Benchmark problems. The initial condition is $u(x, 0) = \sin(2\pi x)$ for all time-dependent problems. Parameters are: $\alpha = \frac{1}{20}$, $\beta = 2$, $c = \frac{1}{2}$ with $\Omega = (0, 1)$ for 1D Poisson and $\Omega = (0, 1) \times (0, 1)$ for all other problems.

Problem	PDE	Boundary Conditions
Poisson	$\nabla^2 u - f = 0$	Dirichlet: $u(x_L) = u(x_R) = 0$
Heat	$\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0$	Dirichlet: $u(x_L) = u(x_R) = 0$
Drift-Diffusion	$\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} - \beta \frac{\partial u}{\partial x} = 0$	Periodic: $u(x_L) = u(x_R)$
Wave	$\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = 0$	Dirichlet: $u(x_L) = u(x_R) = 0$



(a) Poisson 2D (b) Heat (c) Drift-Diffusion (d) Wave

Figure 1: Exact solutions for the benchmark problems. The horizontal line in (a) at $y = 0.5$ represents the solution for the 1D Poisson problem.

4. Experiments & Discussion

We evaluate whether the proposed method produces error maps that accurately explain where and by how much a PINN prediction deviates from the true solution. For this, we consider four benchmark PDEs (Tab. 1, Fig. 1), with the Poisson equation evaluated in both one and two dimensions. All problems admit known analytical solutions u, enabling direct comparison of our error estimates against the ground truth. For each problem we train a hard-constrained PINN φ using the DeepXDE library [25]: an MLP with 3 hidden layers of 20 neurons, $\tanh$ activations, optimized with Adam at a learning rate of 10^{-3}. We consider two PINN configurations, *well-trained* (10 000 collocation points and 10 000 iterations) and *randomly initialized* (no training), to demonstrate that our method works across a range of PINN qualities.

We compare three error estimates against the true error $e_{\text{true}} = u - \hat{\varphi}$:

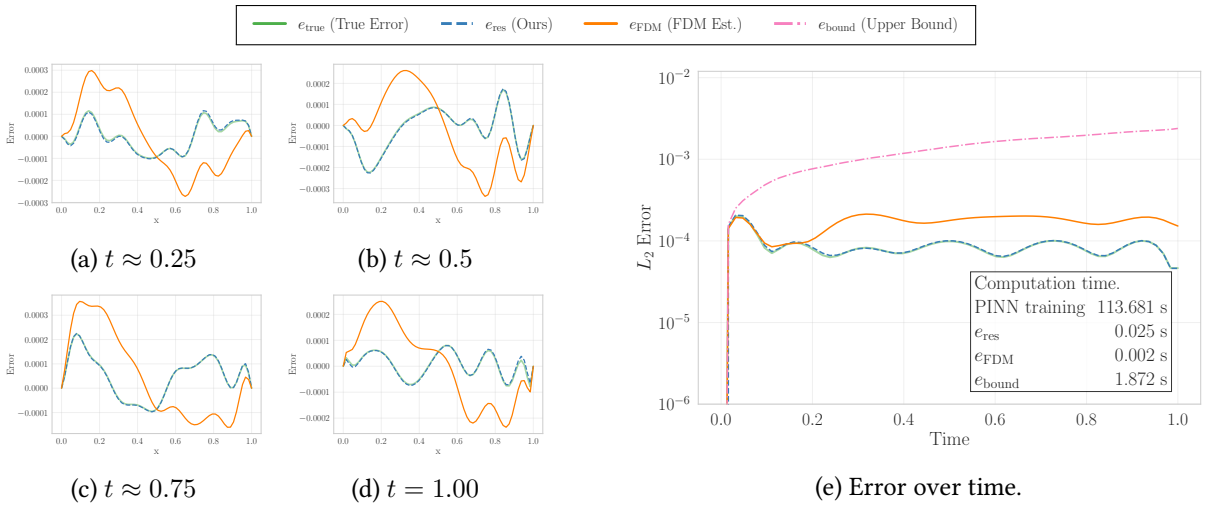
1. **Residual-based** (ours): e_{res}, obtained by solving $\mathcal{D}[e] = -R$ via FDM (Sec. 3).
2. **FDM reference**: $e_{\text{FDM}} = u_{\text{FDM}} - \hat{\varphi}$, where u_{FDM} is obtained by solving the original PDE via FDM using the same grid.
3. **Certified bound** [18]: e_{bound}, a rigorous upper bound applied to the heat equation[1].

e_{FDM} represents the standard approach for evaluating PINNs when analytical solutions are not available, i.e., comparing against a numerically obtained solution. The certified bound e_{bound} represents a PINN-specific baseline proposed for error estimation [18]; however it offers no spatial resolution of the error and thus we only include it for comparison over the time domain.

Fig. 2 presents results for the heat equation for a well-trained PINN. At four time steps (Fig. 2a–d), the proposed residual-based estimate e_{res} tracks the true error closely in both shape and magnitude throughout the spatial domain. The FDM reference e_{FDM}, in contrast, performs worse at equal discretization. The error-over-time results (Fig. 2e) illustrate this gap from another perspective[2]. Throughout time our method matches the true error closely and thus offers a complementary approach to the conservative error bound e_{bound}, which overestimates the error.

[1]The upper bound is given as $e_{\text{bound}}(t) \leq \int_{s=0}^{t} M e^{\omega(t-s)} \|\widetilde{R}(\cdot, s)\|_2 \, \mathrm{d}s$, with $M = 1$ and $\omega = -\frac{1}{20}\pi^2$ derived from the semigroup of the PDE operator and $\widetilde{R}$ being a smoothed version of R, following [18].

[2]The spatial L_2 error is measured as $\|e(\cdot, t)\|_{L^2(\Omega)} = \sqrt{\int_\Omega e(x, t)^2 \, dx}$ and computed via the trapezoidal rule for each t.



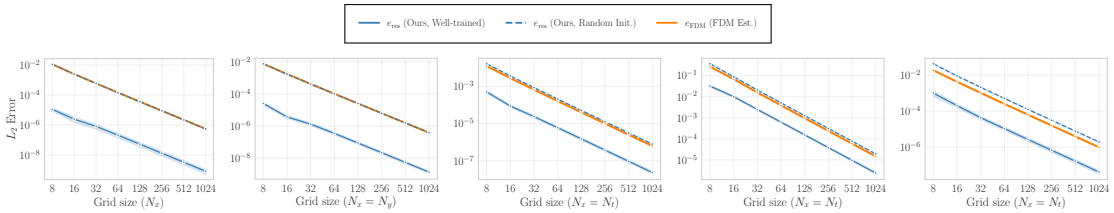
(a) $t \approx 0.25$ (b) $t \approx 0.5$ (c) $t \approx 0.75$ (d) $t = 1.00$ (e) Error over time.

Figure 2: Error estimates for a *well-trained* PINN on the heat equation, (a)–(d) at four different time-points across space and (e) L_2 error over time, including the bound e_{bound}. The FDM-based methods use a 64×64 spatio-temporal grid (time slices shown at nearest grid point to specified values); e_{bound} uses 64 spatial points with adaptive time integration (up to 165 steps at $t = 1$).

Fig. 3 shows the accuracy in the error estimate as the L_2 error between the ground truth and the given method. For well-trained PINNs, e_{res} is consistently orders of magnitude more accurate than the FDM baseline e_{FDM}. Notably, the error estimates are already accurate at low discretization. For randomly initialized PINNs, e_{res} is less accurate than for well-trained ones, though it remains comparable to the baseline e_{FDM}. Furthermore, all configurations become more accurate with higher discretization, though at increased computational cost.

Discussion The experiments demonstrate the method's usefulness. Most notably, when the PINN is well-trained, solving the error equation using FDM achieves more accurate error estimates than through solving the original IBVP. We note, however, that the method is sensitive to the PINN's residual, achieving worse estimates for randomly initialized models. This sensitivity is likely tied to the smoothness of the PINN's PDE residual, which affects the approximated derivatives used in the FDM calculation. Our method also achieves accuracies comparable to the baseline for randomly initialized models.

Computationally, our method adds negligible overhead relative to PINN training (Fig. 2e), though the repeated backward passes required for residual evaluation render it more expensive than the baseline e_{FDM}. From an explainability perspective, the spatially resolved error maps produced by our method provide a diagnostic that localizes model failures. Our error maps reveal where and by how much a predicted approximation deviates from the true solution. This enables local, informed decisions about whether to trust or reject PINN predictions at specific locations in the domain.



(a) Poisson 1D (b) Poisson 2D (c) Heat (d) Drift-Diffusion (e) Wave

Figure 3: Accuracies of the error estimates across different discretizations, measured as $\|e_{\text{true}} - e_{\text{method}}\|_2$ on respective grid points, for all benchmark problems. Averaged over 10 runs, bands show one standard deviation.

Limitations Our experiments are restricted to hard-constrained PINNs, where boundary and initial conditions are satisfied exactly. We note that the output transformations, ensuring the hard-constraining, combined with normalized weight initialization, produce relatively smooth residuals even for untrained networks, which likely benefits our method. Extending the framework to soft-constrained PINNs by including the errors at boundaries into the FDM computation is conceptually straightforward but may, in practice, yield non-smooth residuals close to the boundaries. Whether the method retains similar robustness for soft-constrained PINNs remains an open question.

Furthermore, the considered benchmark problems are all linear and are constructed similarly. The practical transferability of the proposed approach to more complex problems, including non-linear, high-dimensional ones and those with complex geometries, may require more sophisticated solvers or alternative discretizations. From an interpretability point of view, the proposed method explains how far a prediction deviates but not why. Identifying sources of PINN failures remains future work. Finally, our method provides error estimates, not rigorous bounds. The well-studied error theory of FDM schemes, however, opens up avenues to extend the proposed method to also include certified bounds for the approximated error.

5. Conclusion

Trustworthy deployment of PINNs requires explanations that go beyond training loss and reveal where and by how much predictions deviate from the true solution. We present a lightweight post-hoc method for estimating the prediction error of PINNs by solving the associated error equation using finite difference methods. This approach requires no knowledge of the true solution and yields accurate, spatially resolved error maps that serve as quantitative diagnostics of PINN prediction quality, identifying where a model can be trusted and where it fails. Across all benchmark problems, our method consistently achieves more accurate error estimates than a direct FDM solution for well-trained and hard-constrained PINNs, and remains competitive for randomly initialized ones. Promising extensions include more complex problem classes, as well as building on top of the well-understood error theory of FDM schemes towards certified bounds derived from the estimated error.

Acknowledgments

This work was supported by the Fraunhofer Internal Programs under Grant No. PREPARE 40-08394.

Declaration on Generative AI

During the preparation of this work, the authors used Claude Sonnet 4.5, 4.6, Claude Opus 4.5, 4.6 in order to: check spelling and grammar, improve writing style, and as a coding assistant.

References

[1] M. Raissi, P. Perdikaris, 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 (2019) 686–707.

[2] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nature Reviews Physics 3 (2021) 422–440.

[3] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next, Journal of Scientific Computing 92 (2022) 88.

[4] J. D. Toscano, V. Oommen, A. J. Varghese, Z. Zou, N. A. Daryakenari, C. Wu, G. E. Karniadakis, From PINNs to PIKANs: Recent Advances in Physics-Informed Machine Learning, Machine Learning for Computational Science and Engineering 1 (2025) 15.
[5] S. Wang, Y. Teng, P. Perdikaris, Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks, SIAM Journal on Scientific Computing (2021).
[6] S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, Journal of Computational Physics 449 (2022) 110768.
[7] P. Rathore, W. Lei, Z. Frangella, L. Lu, M. Udell, Challenges in Training PINNs: A Loss Landscape Perspective, in: Proceedings of the 41st international conference on machine learning, 2024.
[8] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, in: Advances in Neural Information Processing Systems, volume 34, Curran Associates, Inc., 2021, pp. 26548–26560.
[9] A. Daw, J. Bu, S. Wang, P. Perdikaris, A. Karpatne, Mitigating Propagation Failures in Physics-informed Neural Networks using Retain-Resample-Release (R3) Sampling, in: Proceedings of the 40th International Conference on Machine Learning, PMLR, 2023, pp. 7264–7302.
[10] S. J. Wetzel, S. Ha, R. Iten, M. Klopotek, Z. Liu, Interpretable Machine Learning in Physics: A Review, 2025. arXiv:2503.23616.
[11] T. De Ryck, S. Mishra, Generic bounds on the approximation error for physics-informed (and) operator learning, in: Proceedings of the 36th international conference on neural information processing systems, 2022.
[12] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating PDEs, IMA Journal of Numerical Analysis 43 (2023) 1–43.
[13] T. D. Ryck, S. Mishra, Numerical analysis of physics-informed neural networks and related models in physics-informed machine learning, Acta Numerica 33 (2024) 633–713.
[14] F. Eiras, A. Bibi, R. Bunel, K. D. Dvijotham, P. H. Torr, M. P. Kumar, Efficient error certification for physics-informed neural networks, in: Proceedings of the 41st international conference on machine learning, 2024.
[15] B. Hillebrecht, B. Unger, Certified machine learning: A posteriori error estimation for physics-informed neural networks, in: 2022 International Joint Conference on Neural Networks (IJCNN), IEEE, 2022, pp. 1–8.
[16] B. Hillebrecht, B. Unger, Prediction error certification for PINNs: Theory, computation, and application to Stokes flow, 2025. arXiv:2508.07994.
[17] L. Ernst, N. Rekatsinas, K. Urban, A posteriori certification for neural network approximations to PDEs, 2025. arXiv:2502.20336.
[18] B. Hillebrecht, B. Unger, Rigorous a Posteriori Error Bounds for PDE-Defined PINNs, IEEE Transactions on Neural Networks and Learning Systems 36 (2025) 1583–1593.
[19] S. Liu, X. Huang, P. Protopapas, Residual-based error bound for physics-informed neural networks, in: Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence, PMLR, 2023, pp. 1284–1293.
[20] Z. Fan, Y. Sun, S. Yang, Y. Lu, Physics-Informed Inference Time Scaling for Solving High-Dimensional PDE via Defect Correction, 2025. arXiv:2504.16172.
[21] T. Grätsch, K.-J. Bathe, A posteriori error estimation techniques in practical finite element analysis, Computers & Structures 83 (2005) 235–265.
[22] S. Wang, S. Sankaran, H. Wang, P. Perdikaris, An Expert's Guide to Training Physics-informed Neural Networks, 2023. arXiv:2308.08468.
[23] J. C. Strikwerda, Finite difference schemes and partial differential equations, number 88 in Other titles in applied mathematics, 2nd ed., Society for Industrial and Applied Mathematics, 2004.
[24] J. Crank, P. Nicolson, A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type, Mathematical Proceedings of the Cambridge Philosophical Society 43 (1947) 50–67.
[25] L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, DeepXDE: A Deep Learning Library for Solving Differential Equations, SIAM Review 63 (2021) 208–228.