Improving physics-informed neural networks with an adaptive Fast Fourier Transform (FFT) loss weighting algorithm

Jennifer Zhang and Anish Baradhi

1/12/2026

Period 1 - Dr. Yilmaz

Quarter 2 Project

Abstract

Physics-informed neural networks (PINNs) are a relatively new type of neural network that offers a promising solution to solving partial differential equations (PDEs), especially those that are too complex for traditional numerical approaches. They accomplish this by embedding the governing PDE into the neural network's loss function. However, applying PINNs to complex problems remains challenging. The training process tends to overlook sharp gradients or singular behavior such as those present in many open problems, including the Navier-Stokes existence/smoothness. Furthermore, traditional PINNs require extremely high precision and, therefore, a large amount of computing power to predict these outcomes. In this paper, we introduce a PINN that applies an adaptive loss weighting algorithm inspired by Fourier analysis. Our algorithm randomly samples line segments from the domain and calculates the proportion of error attributed to high-frequency oscillations. PINNs struggle with high-frequency information so this method compensates. Numerical experiments show that the adaptive loss weighting algorithm increases areas of the domain with the highest precision, potentially offering a more efficient approach for solving PDEs.

Introduction

Partial differential equations (PDEs) govern a wide range of physical systems, including fluid dynamics, heat transfer, and electromagnetism. While classical numerical methods are well established, they often require significant computational resources, especially for complex PDEs such as the Navier-Stokes equations (NSEs). Physics-informed neural networks (PINNs) have emerged as a promising alternative by using neural networks to approximate these PDEs.

PINNs incorporate the governing PDE, along with boundary and initial conditions, directly into the loss function of the neural network. A PINN loss function is a weighted sum of the PDE loss, boundary condition loss, and initial condition loss, allowing PINNs to learn solutions without relying on large labeled datasets. PINNs benefit from the universal approximation theorem, which states that neural networks with a sufficiently expressive structure can approximate any continuous function. However, training PINNs for complex PDEs remain challenging.

The loss function plays a major role in the prediction accuracy of PINNs. Yet, standard PINNs use fixed loss weights to balance the loss of the PDE, boundary conditions, and initial conditions. While this approach is simple to implement, it assumes that all constraints are equally important and similarly scaled, which is rarely the case for some nonlinear and multiscale systems such as the NSEs. Poorly chosen weights can lead to slow convergence, biased or incorrect solutions, or completely failed training. Neural networks also exhibit spectral bias, a tendency to learn smooth regions better than sharp gradients and fine vortical structures. This is problematic for fluid dynamics problems.

Our research seeks to address the issues of static weighting and spectral bias by implementing an adaptive PINN that learns the optimal loss weights during training. PINNs with adaptive weighting functions allow the relative importance of different loss components to change during training. Instead of prescribing fixed weights, the model introduces trainable weights that multiply individual loss terms, such as each PDE equation, boundary condition, or initial condition. These weights are optimized alongside the neural network parameters, enabling the training process to balance competing loss terms. As a result, the network can focus more on constraints that are currently harder to satisfy, improving convergence and reducing bias toward any single loss term.

Related work

Recent research on PINNs have focused on improving training stability and accuracy through adaptive weighting, sampling strategies, and optimization techniques. These approaches can be grouped into three main categories.

Adaptive loss weighting

Chen and Howard proposed a method that balances the convergence rates across all training points. They accomplished this by assigning higher weights to loss terms with slower residual decay. Interestingly, they also incorporated adaptive sampling, which introduces more training points in regions with large residuals. This is important because residuals vary across the domain of interest, and the objective is to minimize all residuals. We were intrigued by the simultaneous use of adaptive weighting and adaptive sampling, but chose to focus solely on adaptive weighting due to time constraints.

Wang et al. propose a hybrid PINN that uses maximum likelihood estimation, transformers, and attention to capture local features; a differentiation scheme that uses automatic differentiation; and a different scheme that uses differences to compute spatial derivatives. The primary advantage of this method is the lower L2 errors and faster convergence rates across a series of benchmarks in comparison to standard PINNs. Additionally, the mixed differentiation scheme is more efficient than traditional automatic differentiation. The main drawback is that it requires uniform grid spacing, which traditional PINNs don't need. We think this isn't a good direction to go in since the models we generate will not be able to solve general problems that won't necessarily have grid-like properties.

In Gao et al., the adaptive weighting algorithm constantly monitors the learning speed of each loss term and adjusts weights to prevent certain loss terms from decreasing to quickly or too slowly. If one loss term improved 10 times faster than another, the algorithm flagged this as a problem. However, their approach assumes that all parts of the problem should be learned at roughly the same speed. In reality, this is not true, and forcing everything to progress at a similar pace can affect learning quality or waste effort on easy parts of the problem. Furthermore, constantly changing the importance of tasks can make training unstable.

Pratama et al. propose a variation of the self-adaptive PINN, where model parameters are changed through the training process. This implementation is distinct through its use of genetic algorithms to determine the loss functions through mutation, crossing over, and selection cycles to generate weights and biases. The strengths of this approach are that it is an order of magnitude more precise than naive L-BFGS in some cases, and it appears to improve as the network becomes more complex. The major weakness of this approach is the extremely high computational cost. Since entire populations are evaluated at each step, it would add a large constant factor to the already very arduous L-BFGS training process. This suggests that L-BFGS-based optimizations are intractable to us but optimizations during the adam stage might be promising.

Wong et al. suggest an evolutionary algorithm that is gradient-free and less sensitive to vanishing gradients. However, evolutionary algorithms require significantly more computational resources than gradient-based methods. Similar to Pratama et al., they require repeated evaluations of entire populations. Thus, Evo-PINNs are impractical for very large neural networks or high-dimensional PDEs on standard computers like the ones we are using.

Datasets and features

Unlike conventional supervised learning, PINNs do not require any labeled data. Instead, the training data is composed of coordinates sampled from a rectangular domain and constraints derived from the Navier-Stokes equations.

For our experiments, we use the two-dimensional TGV (see Appendix), which is defined on the domain below (x, y, t) where $x \in [0, 2\pi]$, $y \in [0, 2\pi]$, $t \in [0, 1]$. We generated collocation points by uniformly sampling coordinates from the domain. These coordinates are passed to a neural network that outputs the predicted physical quantities based on the Navier-Stokes equations. Here, the outputs are the velocity components and pressure (u, v, p) of an incompressible fluid.

The analytical TGV solution serves as a benchmark to validate model accuracy, but it is not used during training, except to define boundary and initial conditions.

Methods

We used the DeepXDE library, which is standard for scientific machine learning and physics-informed learning, to solve the PDEs with a TensorFlow backend. We used NumPy for data storage and manipulation and Matplotlib for data visualization.

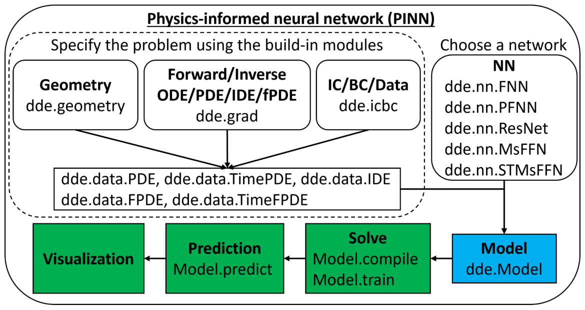

Figure 1. DeepXDE Implementation Flowchart

PINN with standard weighting function

The PINN loss function is defined as $loss = \lambda_{PDE} loss_{PDE} + \lambda_{BC} loss_{BC} + \lambda_{IC} loss_{IC}$. In standard PINNs, the weights λ_i are fixed parameters.

In DeepXDE, we created a rectangle to serve as our domain, where $0 \le x, y \le 2\pi$ and combined it with a time domain to create a 3D domain geometry. We imposed the Dirichlet boundary condition, which fixes the solution of the NSEs along the boundary of the domain using the analytical TGV solution.

The neural network architecture for the baseline PINN consists of four hidden layers with 67 neurons each, tanh activation functions, and Xavier initialization. We performed training using Adam optimization for 10,000 epochs, followed by fine tuning with L-BFGS-B optimization.

Trained adaptive weighting & FFT frequency optimization

Adaptive weighting is a broad class of techniques that change the relative importance of loss components during training. Typical methods compute weights from running statistics (loss magnitudes, gradient norms, or heuristics) and then apply those computed scalars to each term.

We implemented a parametric approach: the scalars themselves are explicit trainable variables inside the model. We optimized the coefficients for each of the loss terms concurrently with the PINN during the adam optimizer stage. We also included an extra FFT term that will be described next.

The second optimization monitors the residuals in frequency space and applies frequency information to guide training. We periodically sampled random line segments within the domain and constructed a list of the residuals. Neural networks tend to learn smooth, low-frequency structures first. If the FFT shows a persistent high-frequency component of the error, we temporarily amplify the PDE loss so that the optimizer prioritizes resolving fine features. This feedback loop directly targets the PINN spectral bias problem without changing network architecture or adding specialized basis functions.

Training strategy

Training occurs in two phases. First, Adam optimization is used with adaptive FFT-weighted loss to encourage more learning near sharp gradients and high-frequency structures. Second, L-BFGS-B optimization is applied to refine the solution without FFT adaptive weighting.

Experiments, results, and discussion

The PINNs are evaluated on the TGV, a classical benchmark with a known analytical solution. We evaluated the performance using the relative L2 error for u, v, and p. For example, the relative L2 error for u is formalized as:

$$\epsilon_u = \frac{||u_{PINN} - u_{exact}||_2}{||u_{exact}||_2}.$$

We compared the baseline PINN with fixed loss weights against the adaptive FFT-weighted PINN with identical network architectures and training parameters.

Figure 2 illustrates the change in total training loss as the number of iterations for both the baseline PINN and the adaptive FFT-weighted PINN. During the early stages of training with the Adam optimizer, both models exhibit a rapid decrease in loss, with the adaptive PINN decreasing at a slightly faster rate. This indicates that the neural networks quickly learn the dominant low-frequency structure of TGV and that the adaptive PINN achieves this goal more efficiently than the baseline PINN.

We observe notable differences as training progresses. The baseline PINN shows multiple oscillations in training loss during the intermediate iterations from approximately $n = 2000$ to $n = 10,000$. Meanwhile, the adaptive PINN demonstrates a smoother decline in loss during the Adam phase. This indicates that FFT-weighted PINN is better adapted to the high-frequency nature of TGV and doesn’t need to retroactively correct for spectral bias, unlike the baseline PINN.

The performance gap between the baseline and adaptive PINNs narrows during L-BFGS-B fine-tuning, converging towards a similar total training loss. This suggests that the primary benefit of adaptive FFT loss weighting is achieving faster convergence during the first round of optimization.

The bar chart in Figure 2 shows the final relative L2 error for the velocity and pressure fields. The adaptive PINN achieves lower errors for u and p, with the most noticeable improvement in the pressure field.

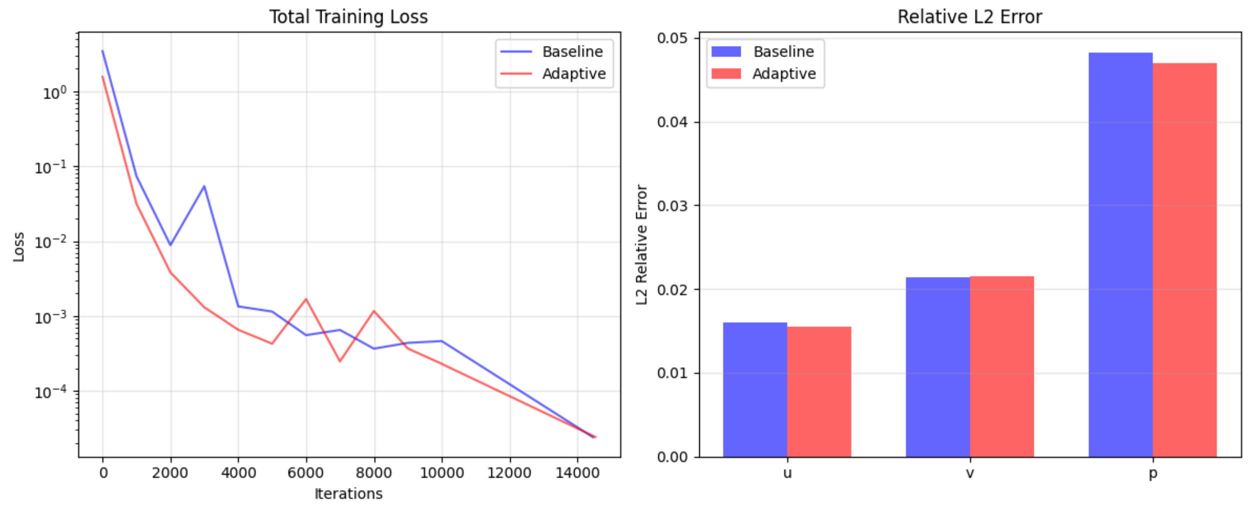

Figure 2. Training loss vs. iteration and relative error on final iteration

Figures 3 and 4 show spatial visualizations of error across the rectangular domain between the analytical TGV solution and the PINN-predicted fields. For the baseline PINN (Figure 3), the error fields for velocity and pressure are periodic and align with regions of high curvature and interactions between vortical structures. The adaptive PINN (Figure 4) shows reduced error, reflected by the lower upper limit for the scale of p's error plot. The most significant improvements occur in regions where the baseline model exhibited the greatest error. Thus, the adaptive model is better able to correct these inaccuracies. Still, regions with lower error remain similar between the two models, reflecting how the adaptive PINN does not harm performance in areas where the solution is learned well.

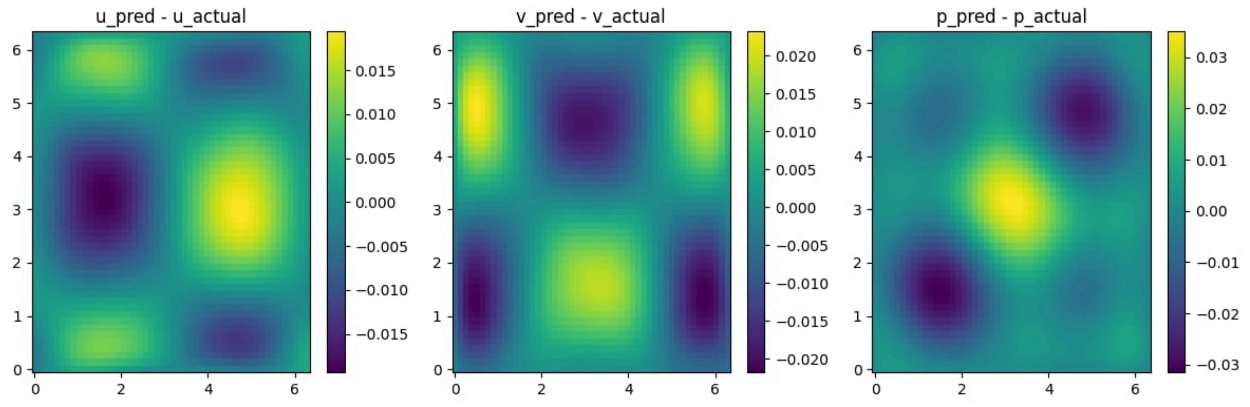

Figure 3. Error between analytical and baseline PINN solutions for TGV

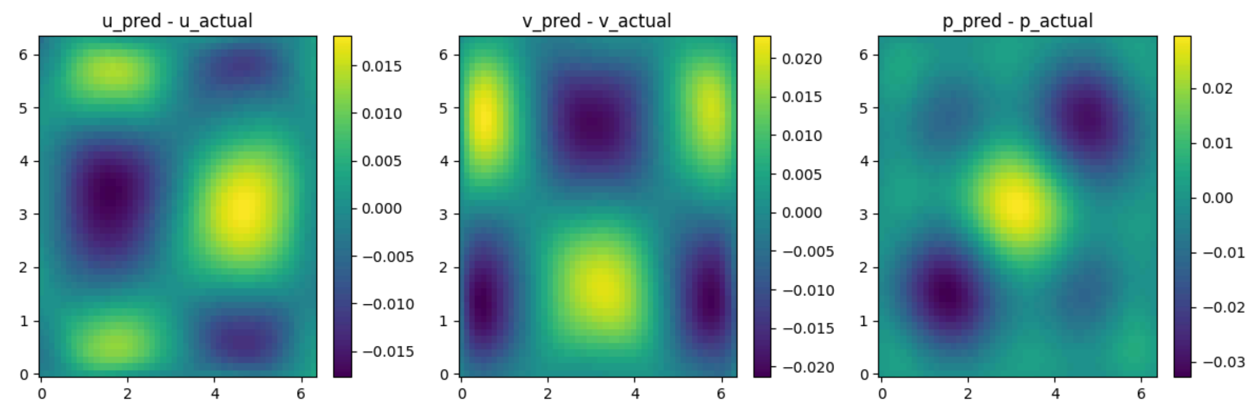

Figure 4. Error between analytical and adaptive PINN solutions for TGV

Conclusion and future work

In this work, we presented an adaptive physics-informed neural network that combines trainable loss weights with frequency feedback from the FFT analysis of PDE residuals. The adaptive PINN improves convergence and the accuracy of the solution before fine-tuning, especially in regions where the baseline PINN experiences the greatest error. However, the final improvement in performance is modest since the baseline PINN converges to approximately the same total training loss after the Adam optimization phase. This suggests that our method has the greatest advantage in early and intermediate training.

Future work includes extending the adaptive weighting to boundary and initial condition losses, which is possible with more computing power and time. We also think it would be interesting to apply the method to other types of flows, including those that are three-dimensional instead of two-dimensional.

Appendix

Navier-Stokes equations

The Navier-Stokes equations (NSEs) are PDEs that describe the motion of viscous fluids, such as liquids and gases, by applying the fundamental physics concepts of the conservation of momentum and the conservation of mass. However, they are notoriously difficult to solve analytically, resulting in solutions that use complex numerical methods.

Taylor-Green vortex

The Taylor-Green vortex (TGV) is a classic fluid dynamics problem that represents an unsteady flow of a decaying vortex. The velocity components are $\vec{v} = (u, v, w)$, defined as follows:

$$u(x, y, z) = A\, cos(ax)\, sin(by)\, sin(cz),$$

$$v(x, y, z) = B\, sin(ax)\, cos(by)\, sin(cz),$$

$$w(x, y, z) = C\, sin(ax)\, sin(by)\, cos(cz).$$

Unlike most turbulent flows, the TGV has an analytical solution for velocity and pressure fields. In the domain $0 \leq x, y \leq 2\pi$, the 2-dimensional solution for the TGV is $\vec{v} = (u, v)$, where ν is a constant describing the fluid's kinematic viscosity:

$$u(x, y, t) = sin(x)\, cos(y)\, e^{-2\nu t},$$

$$v(x, y, t) =- cos(x)\, sin(y)\, e^{-2\nu t},$$

$$\rho(x, y, t) = \frac{\rho}{4}\, (cos(2x) + cos(2y))e^{-4\nu t}.$$

This solution provides functions that explicitly describe the velocity and pressure fields. The NSEs also seek to determine velocity and pressure fields, making the TGV a benchmark for NSE numerical solvers.

Contributions

Jennifer and Anish worked together to develop the approach for PINNs. Anish implemented the approach in Python. Jennifer and Anish completed the paper and slideshow.

References

B. Gao, R. Yao, and Y. Li, "Physics-informed neural networks with adaptive loss weighting algorithm for solving partial differential equations," *Computers & Mathematics with Applications*, vol. 181, pp. 216–227, Mar. 2025. doi: 10.1016/j.camwa.2025.01.007.

C. R. Harris et al., "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020. doi: 10.1038/s41586-020-2649-2.

D. A. Pratama, R. R. Abo-Alsabeh, M. A. Bakar, A. Salhi, and N. F. Ibrahim, "Solving partial differential equations with hybridized physic-informed neural network and optimization approach: Incorporating genetic algorithms and L-BFGS for improved accuracy," *Alexandria Engineering Journal*, vol. 77, pp. 205–226, Aug. 2023, doi: 10.1016/j.aej.2023.06.047.

G. I. Taylor and A. E. Green, "Mechanism of the production of small eddies from large ones," *Proc. R. Soc. Lond.* A, vol. 158, pp. 499–521, 1937.

J. Wang, X. Xiao, X. Feng, and H. Xu, “An improved physics-informed neural network with adaptive weighting and mixed differentiation for solving the incompressible Navier–Stokes equations,” *Nonlinear Dyn*, vol. 112, no. 18, pp. 16113–16134, Sep. 2024, doi: 10.1007/s11071-024-09856-6.

L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, “DeepXDE: A deep learning library for solving differential equations,” *SIAM Review*, vol. 63, no. 1, pp. 208-228, 2021, doi: 10.1137/19M1274067.

J. C. Wong, A. Gupta, C. C. Ooi, P.-H. Chiu, J. Liu, and Y.-S. Ong, “Evolutionary optimization of physics-informed neural networks: Evo-PINN frontiers and opportunities,” *arXiv preprint arXiV: 10.48550/arXiv.2501.06572*, Jan. 2026.

J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing Sci. Eng.*, vol. 9, no. 3, pp. 90-95, 2007.

M. Abadi et al., “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,” 2015. Software available from https://www.tensorflow.org.

W. Chen, A. Howard, and P. Stinis, “Self-adaptive weighting and sampling for physics-informed neural networks.,” *arXiv preprint arXiv:2511.05452*, Nov. 2025