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

Anish Baradhi and Jennifer Zhang
Period 1 (Dr. Yilmaz)

Physics-informed neural networks (PINNs)

- Partial differential equations are important in fluid dynamics, heat transfer, and other physical systems
- PDEs are hard to solve with traditional numerical methods
- PINNs have a loss function that depends on the governing PDE

$$loss = \lambda_{PDE} loss_{PDE} + \lambda_{BC} loss_{BC} + \lambda_{IC} loss_{IC}$$

- After sufficient training, the PINN can approximate the PDEs

The problem

Standard PINNs use fixed loss weights

- Simple to implement but can lead to **slow convergence and poor solutions**
- **Spectral bias:** tendency to learn smooth regions better than sharp gradients and fine vortical structures

Our solution

Adaptive PINN with trainable weights

- Learns the optimal loss weights during training to balance competing loss terms
- Network can focus more on constraints that are currently harder to satisfy
    - Improves convergence and reduces bias toward any single loss term

Datasets and features

- PINNs do not require any labeled data
- Training data is composed of coordinates uniformly sampled from a rectangular domain and constraints derived from the Navier-Stokes equations
- Coordinates are passed to the neural network to output the predicted physical quantities based on the Navier-Stokes equations
  - **Output:** velocity components and pressure (u,v,p) of an incompressible fluid

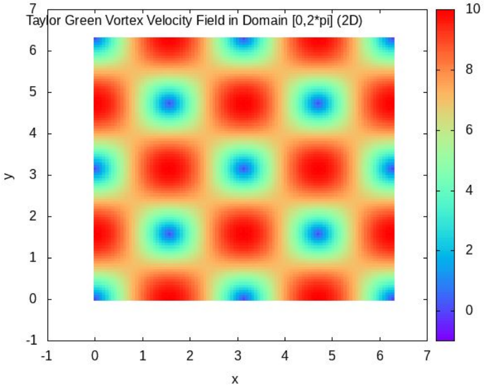
Navier–Stokes equations and Taylor–Green vortex

- **Navier–Stokes equations:** describe motion of viscous fluids, but are notoriously difficult to solve analytically
- **Taylor–Green vortex:** analytically solvable, decaying vortex flow; results used to validate methods for solving the Navier–Stokes equations

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

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

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



Pipeline

Pipeline Overview

1. Initialize model
2. Optimize with adam (and optimize coefficients)
3. Optimize with L-BFGS-B
4. Evaluate model on collocation points

Initialize

- (x, y, t) -> 4 fully connected layers with 67 neurons -> (u, v, p)
- Use Glorot Uniform initialization to ensure gradients remain stable
- For the FFT-weighted PINN add a Loss layer that will contain optimization parameters k_0, k_1, k_2 that will optimize the loss function as well as an FFT term hft
- Connect the network to DeepXDE geometry and PDE functions, allowing for automatic differentiation.

Optimize with adam (and optimize coefficients)

- Use adam (derived from gradient descent) to find a rough approximation of the answer
- Simultaneously update the network coefficients k_0, k_1, k_2 to accurately weight terms relative to each other
- Run an FFT on residuals and increase penalty if the solution becomes noisy (dominated by high frequencies)
- Adjust the hft weight

Optimize with L-BFGS-B

- Uses curvature information (Hessian approximations) to take precise steps
- While adam is good at finding general solutions, L-BFGS-B is much more efficient at driving the residual error to the precision limits of the computer
- k_0, k_1, k_2, and hft are no longer trained

Evaluate model on collocation points

- Generate a 50 x 50 grid of test points across the domain and query the trained network to predict velocity and pressure information
- Compare the predicted value against the actual TGV solution to get the relative L2 error

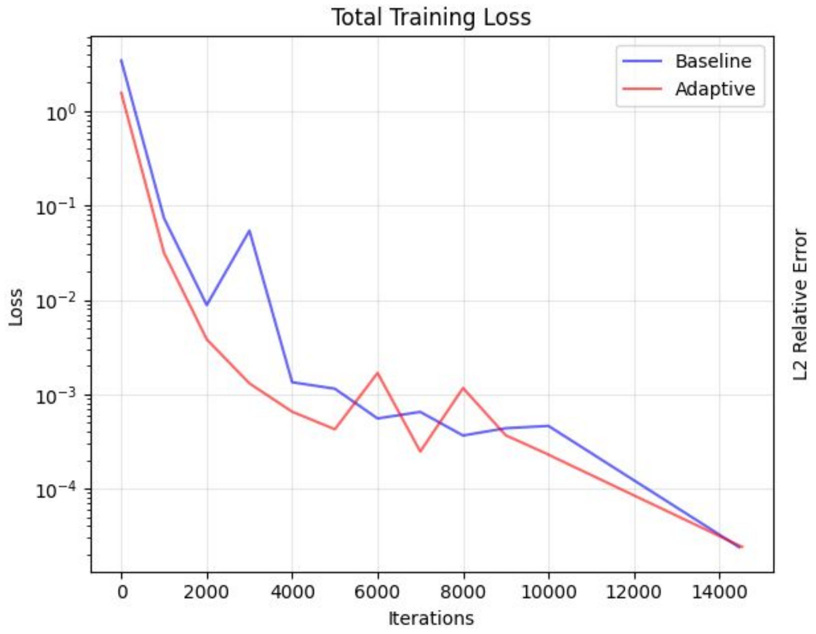
$$\epsilon_u = \frac{||u_{PINN} - u_{exact}||_2}{||u_{exact}||_2}$$

- Generate heatmaps, loss vs. iteration graphs, etc.

Experiments, Results, Discussion

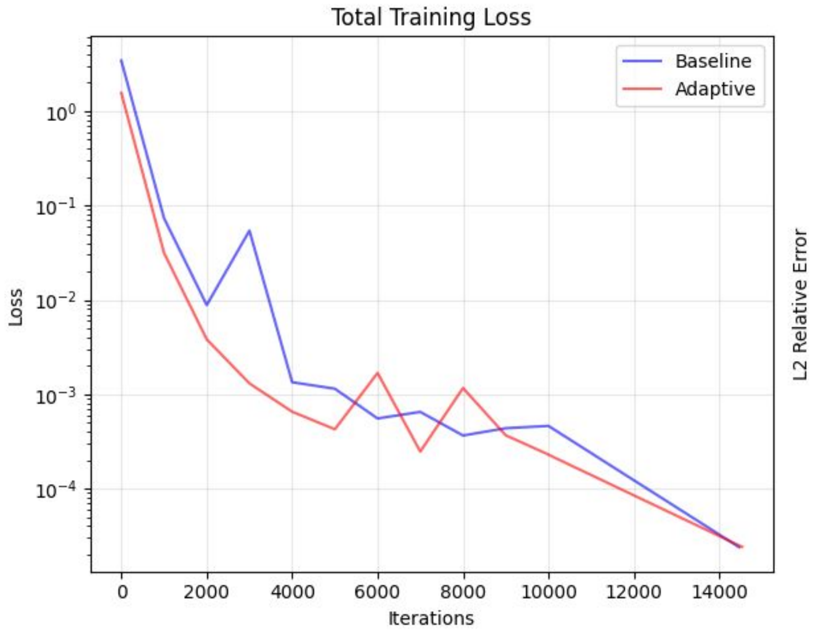
Total Training Loss

- During the Adam phase, loss decreased rapidly for both models
    - Adaptive PINN decreased slightly faster
    - Both models quickly learn the dominant low-frequency structure, adaptive PINN achieves this goal more efficiently



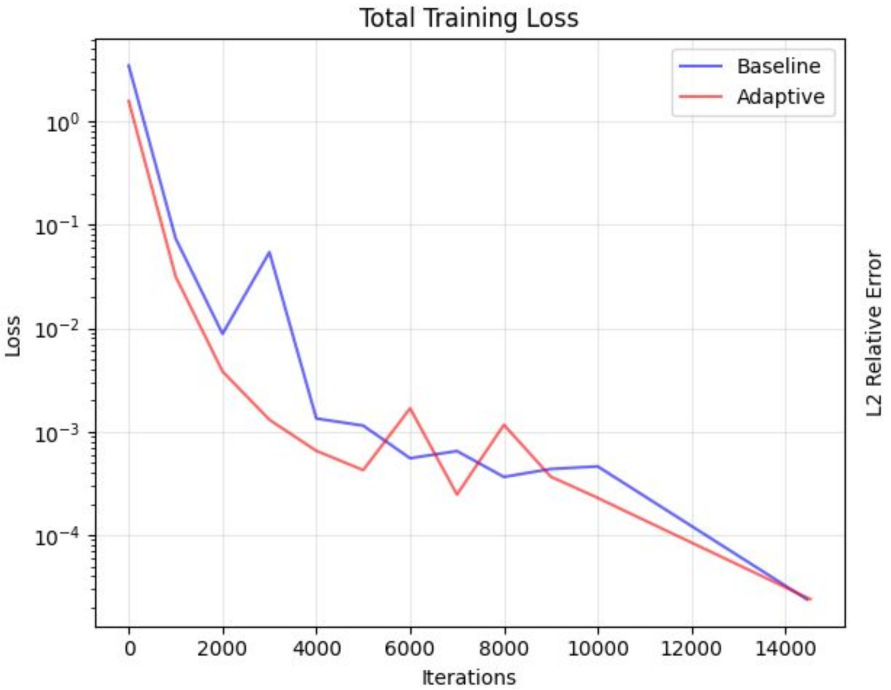
Total Training Loss

- Baseline PINN oscillates more
  - FFT-weighted PINN better adapted to the high-frequency nature of TGV
  - Didn't need to retroactively correct for spectral bias, unlike the baseline PINN



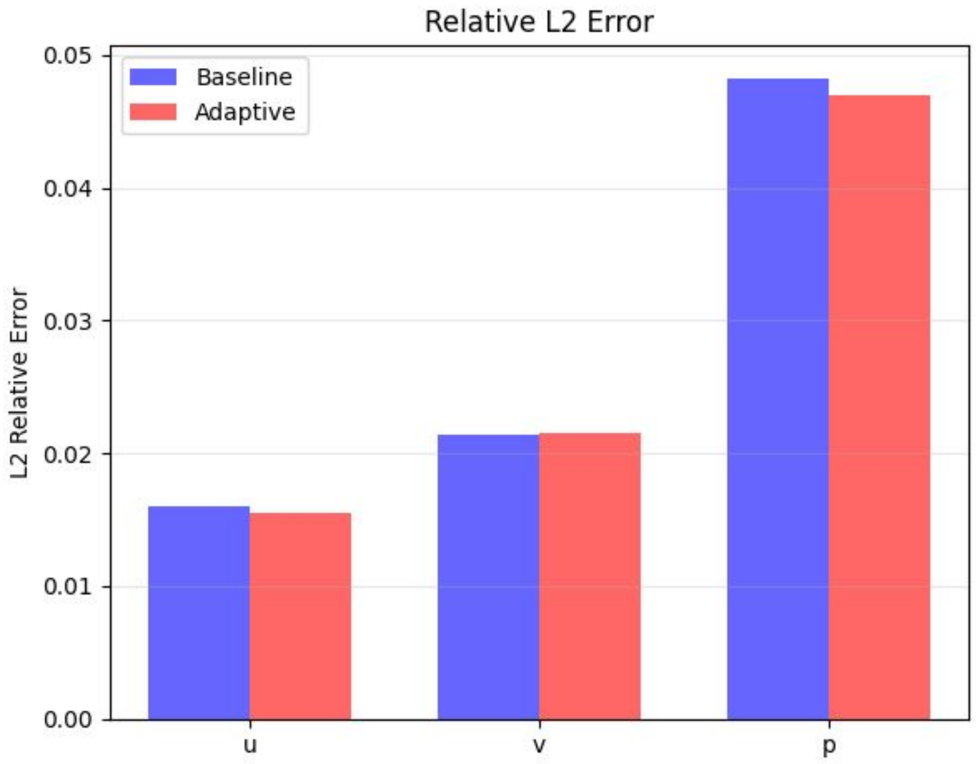
Total Training Loss

- The performance gap between the baseline and adaptive PINNs narrows during L-BFGS-B fine-tuning
  - They converge towards a similar loss
- <u>Main benefit of adaptive weighting:</u> faster convergence during first optimization phase

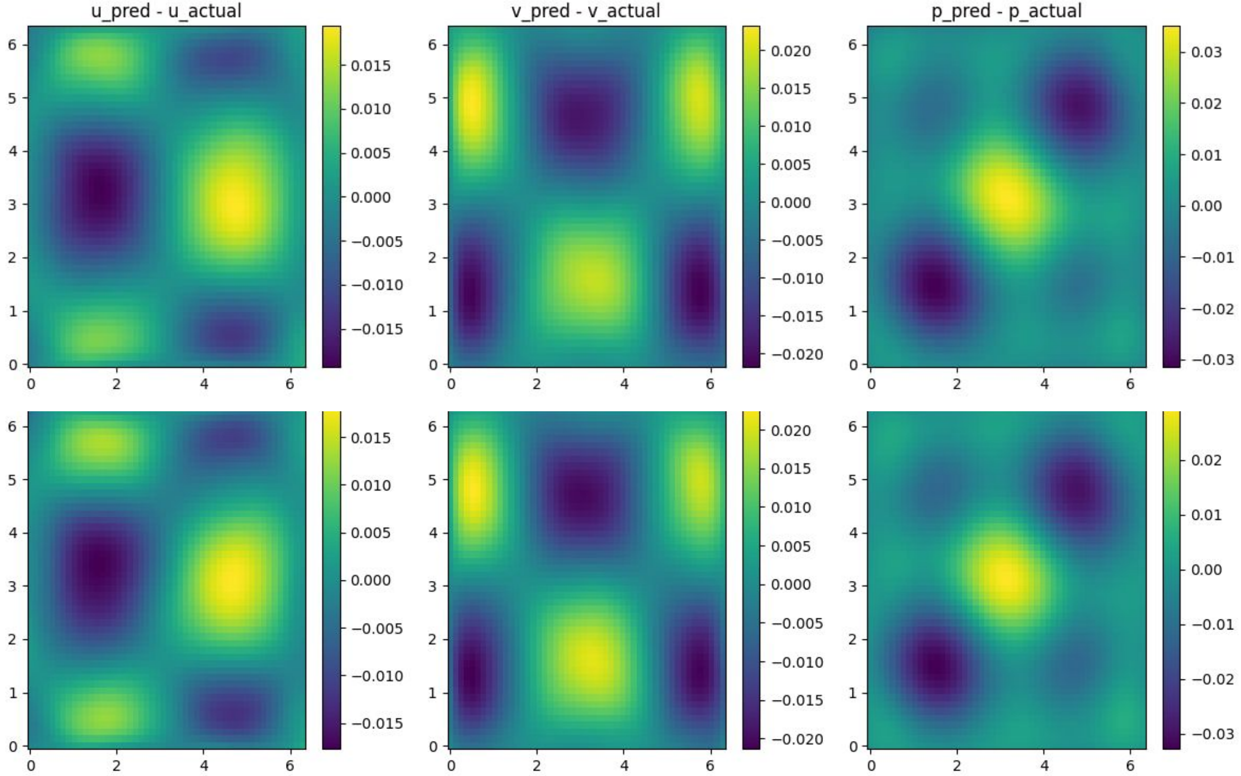


Relative L2 error

Adaptive PINN achieves lower errors for *u* and p, with the most noticeable improvement in *p*, the pressure field.



Error between analytical vs. baseline PINN solutions (top) and analytical vs. adaptive PINN solutions (bottom)



Error between analytical vs. baseline PINN solutions (top) and analytical vs. adaptive PINN solutions (bottom)

- **Baseline PINN:** error fields for velocity and pressure are periodic and align with regions of high curvature and interactions between vortical structures
- **Adaptive PINN:** reduced error, especially in regions where the baseline model exhibited the greatest error
    - Better able to correct bias
- Regions with lower error remain similar between the two models → adaptive PINN does not harm performance in smooth areas where the solution is learned well

Conclusion

Key Takeaways

- **Frequency feedback is valuable**
  - PINNs struggle at high frequency and need to be compensated
- Adaptive weighting **improves convergence**
- More complex data and better compute are need
- Higher dimensions likely benefit from adaptive weighting

Thanks!