Kronecker-Factored Approximate Curvature for Physics-Informed Neural Networks

VECTOR INSTITUTE ETH zürich

Felix Dangel*, Johannes Müller*, Marius Zeinhofer*

Vector Institute (Canada), RWTH Aachen University (Germany), ETH Zürich (Switzerland)

RWTH AACHEN UNIVERSITY

We develop KFAC for loss functions with differential operators.
For training PINNs, our optimizer consistently outperforms SGD/Adam.

Background: What Are PINNs?

Main idea: Train a neural network to satisfy a PDE → loss contains the PDE's differential operator.

- **Goal:** Learn PDE solution $u(\mathbf{x})$

$$\mathcal{L}u(\mathbf{x}) = f(\mathbf{x}) \qquad \mathbf{x} \in \Omega$$
$$u(\mathbf{x}) = g(\mathbf{x}) \qquad \mathbf{x} \in \partial\Omega$$

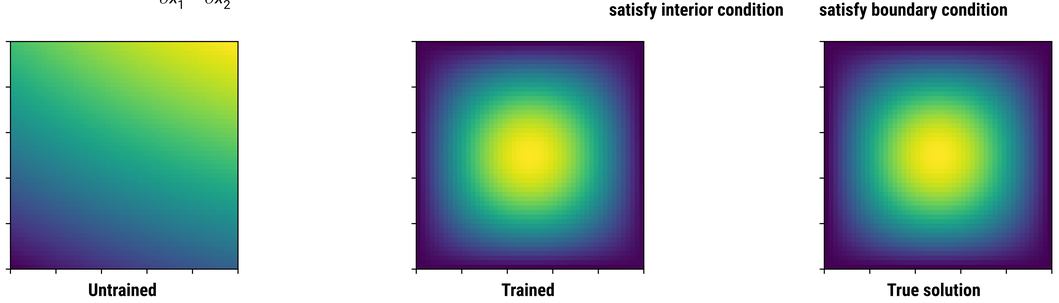
- **Example:** 2d Poisson equation

$$-\mathcal{L} = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}$$

- **Data:** Sample $\mathbf{x}_n \sim \Omega$, $\mathbf{x}_n^\mathrm{b} \sim \partial\Omega$
- **Ansatz:** Neural net $u_\theta(\mathbf{x})$

$$L(\theta) = \underbrace{\frac{1}{2N_\Omega}\sum_{n=1}^{N_\Omega}\left(\mathcal{L}u_\theta(\mathbf{x}_n) - f(\mathbf{x}_n)\right)^2}_{L_\Omega(\theta)} + \underbrace{\frac{1}{2N_{\partial\Omega}}\sum_{n=1}^{N_{\partial\Omega}}\left(u_\theta(\mathbf{x}_n^\mathrm{b}) - g(\mathbf{x}_n^\mathrm{b})\right)^2}_{L_{\partial\Omega}(\theta)}$$

satisfy interior condition **satisfy boundary condition**



Contribution: How Do We Derive KFAC for PINN Losses?

We show that computing differential operators with Taylor-mode autodiff yields networks with linear weight sharing layers
→ This allows us to apply the existing definition of KFAC for linear weight sharing layers.

- **Goal:** Approximate Gauss-Newton matrix

$$\mathbf{G}(\mathbf{W}) = \mathbf{G}_\Omega(\mathbf{W}) + \mathbf{G}_{\partial\Omega}(\mathbf{W})$$

for each linear layer with weight $\mathbf{W}$

$$\mathbf{G}_\Omega(\mathbf{W}) \approx \mathbf{A}_\Omega \otimes \mathbf{B}_\Omega$$
$$\mathbf{G}_{\partial\Omega}(\mathbf{W}) \approx \mathbf{A}_{\partial\Omega} \otimes \mathbf{B}_{\partial\Omega}$$

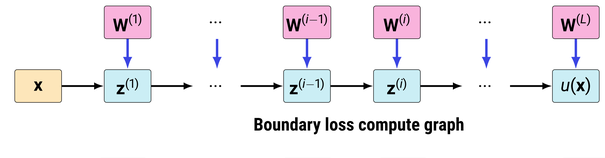
- **Contribution:** Make computation of $L(\theta)$ explicit
  - Linear layer in boundary loss

$$\mathbf{z} \mapsto \mathbf{W}\mathbf{z} \qquad (\mathbf{z} \text{ is a vector})$$

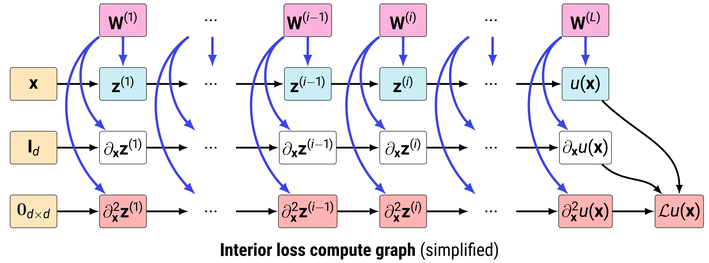
  - Linear layer in interior loss

$$\mathbf{Z} \mapsto \mathbf{W}\mathbf{Z} \qquad (\mathbf{Z} \text{ is a matrix})$$

The computation reduces to a neural net with linear weight sharing layers. We can just apply the existing KFAC definition from Eschenhagen (NeurIPS 2023).



Boundary loss compute graph

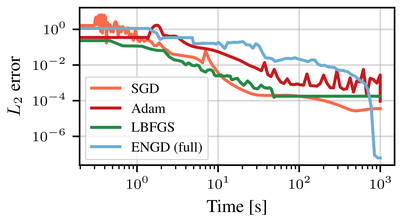


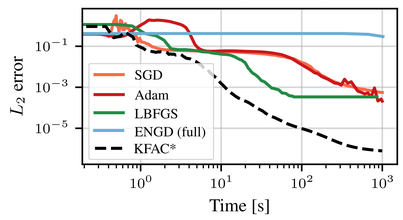
Interior loss compute graph (simplified)

Motivation: PINNs Are Hard to Train, Second-order Methods Can Help

Natural gradient methods beat first-order methods on small problems... but do not scale well to larger nets → our KFAC scales.

Small net (D = 257)

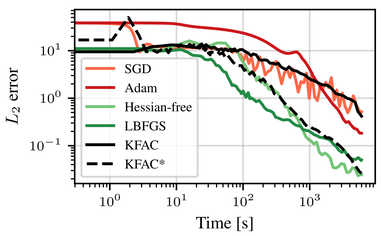




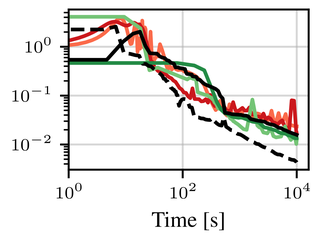
Medium net (D = 9873)

Evaluation: Our KFAC Optimizer Outperforms First-order Methods and Scales Well

9+1d log-Fokker-Planck equation, D ≈ 10^5



100d Poisson equation, D ≈ 10^6



4+1d Heat equation, D ≈ 10^5