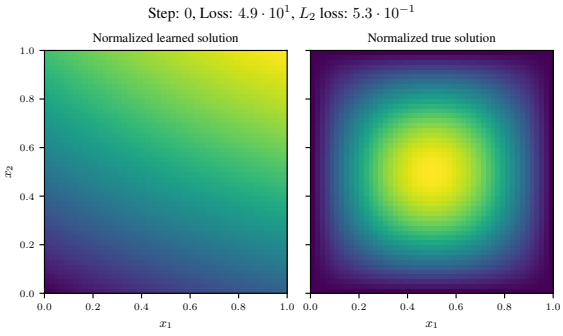
Kronecker-factored Approximate Curvature for Physics-Informed Neural Networks

Felix Dangel*, Johannes Müller*, Marius Zeinhofer*

NeurIPS 2024

Motivation & Contribution

We develop KFAC for, and apply it to, loss functions with differential operators.



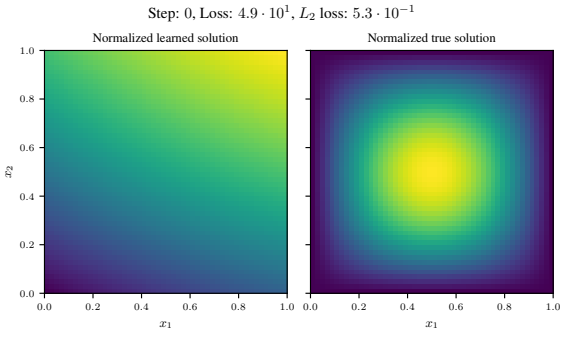
Motivation & Contribution

We develop KFAC for, and apply it to, loss functions with differential operators.

- Goal: Learn PDE solution

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

- Ansatz: Neural net $u_{\boldsymbol{\theta}}(\boldsymbol{x})$



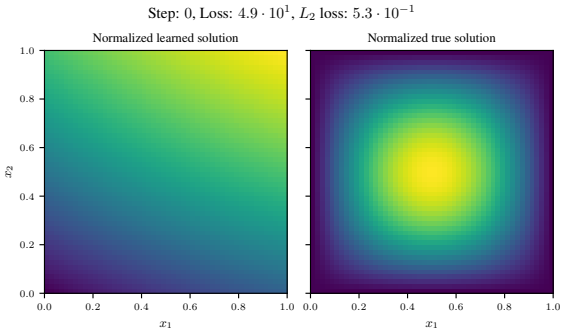
Motivation & Contribution

We develop KFAC for, and apply it to, loss functions with differential operators.

- Goal: Learn PDE solution

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

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



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

Motivation & Contribution

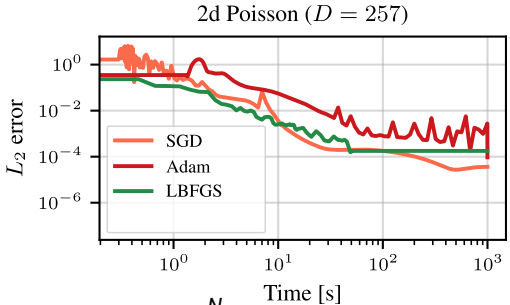
We develop KFAC for, and apply it to, loss functions with differential operators.

- Goal: Learn PDE solution

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

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

Hard to train with popular methods



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

Motivation & Contribution

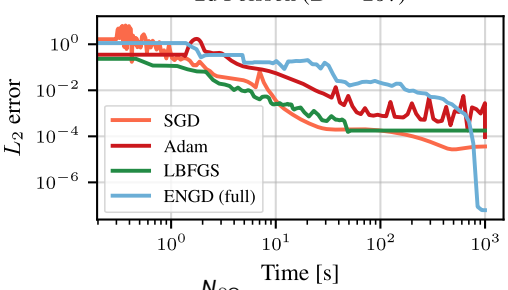
We develop KFAC for, and apply it to, loss functions with differential operators.

- Goal: Learn PDE solution

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

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

2nd-order works [Müller and Zeinhofer, 2023, ICML]



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

Motivation & Contribution

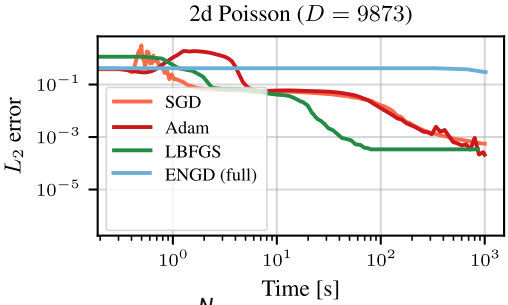
We develop KFAC for, and apply it to, loss functions with differential operators.

- Goal: Learn PDE solution

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

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

but proposed methods don't scale



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

Motivation & Contribution

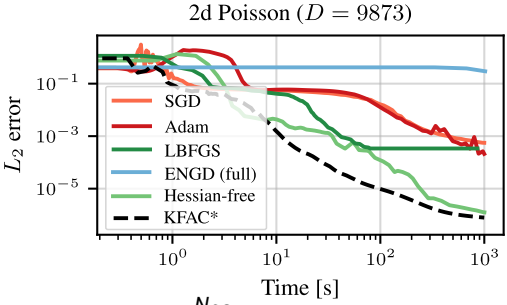
We develop KFAC for, and apply it to, loss functions with differential operators.

- Goal: Learn PDE solution

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

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

We derive KFAC, which scales well



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

High-level Walkthrough

- Neural network ansatz $u_\theta = f^{(L)} \circ f^{(L-1)} \circ \ldots \circ f^{(1)}$

High-level Walkthrough

- Neural network ansatz $u_\theta = f^{(L)} \circ f^{(L-1)} \circ \ldots \circ f^{(1)}$
- Evaluating $u_\theta(x)$

$$z^{(0)} \mapsto z^{(1)} \mapsto z^{(2)} \mapsto \cdots \mapsto z^{(L)} = u_\theta(x)$$

with **vector-valued** $z^{(l)} = f^{(l)}(z^{(l-1)})$ and $z^{(0)} = x$

High-level Walkthrough

- Neural network ansatz $u_\theta = f^{(L)} \circ f^{(L-1)} \circ \ldots \circ f^{(1)}$
- Evaluating $u_\theta(\boldsymbol{x})$

$$\boldsymbol{z}^{(0)} \mapsto \boldsymbol{z}^{(1)} \mapsto \boldsymbol{z}^{(2)} \mapsto \cdots \mapsto \boldsymbol{z}^{(L)} = u_\theta(\boldsymbol{x})$$

  with **vector-valued** $\boldsymbol{z}^{(l)} = f^{(l)}(\boldsymbol{z}^{(l-1)})$ and $\boldsymbol{z}^{(0)} = \boldsymbol{x}$
- Evaluating $\mathcal{L}u_\theta(\boldsymbol{x})$

$$\boldsymbol{Z}^{(0)} \mapsto \boldsymbol{Z}^{(1)} \mapsto \boldsymbol{Z}^{(2)} \mapsto \ldots \mapsto \boldsymbol{Z}^{(L)} \mapsto \mathcal{L}u_\theta(\boldsymbol{x})$$

  with **matrix-valued** $\boldsymbol{Z}^{(l)}$ and dependencies determined by Taylor-mode

High-level Walkthrough

Computing differential operators yields networks with linear weight sharing layers.
$\rightarrow$ Apply KFAC for linear layers with weight sharing [Eschenhagen et al., 2023, NeurIPS]

- Neural network ansatz $u_\theta = f^{(L)} \circ f^{(L-1)} \circ \ldots \circ f^{(1)}$
- Evaluating $u_\theta(x)$ **Linear layer: $z^{(l)} = Wz^{(l-1)}$**

$$z^{(0)} \mapsto z^{(1)} \mapsto z^{(2)} \mapsto \cdots \mapsto z^{(L)} = u_\theta(x)$$

with **vector-valued** $z^{(l)} = f^{(l)}(z^{(l-1)})$ and $z^{(0)} = x$

- Evaluating $\mathcal{L}u_\theta(x)$ **Linear layer: $Z^{(l)} = WZ^{(l-1)}$**

$$Z^{(0)} \mapsto Z^{(1)} \mapsto Z^{(2)} \mapsto \ldots \mapsto Z^{(L)} \mapsto \mathcal{L}u_\theta(x)$$

with **matrix-valued** $Z^{(l)}$ and dependencies determined by Taylor-mode

Algorithm, Experiments & Conclusion

$$L(\boldsymbol{\theta}) = L_{\Omega}(\boldsymbol{\theta}) + L_{\partial\Omega}(\boldsymbol{\theta})$$

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

  $$\boldsymbol{G}(\boldsymbol{\theta}) = \boldsymbol{G}_{\Omega}(\boldsymbol{\theta}) + \boldsymbol{G}_{\partial\Omega}(\boldsymbol{\theta})$$

Algorithm, Experiments & Conclusion

$$L(\boldsymbol{\theta}) = L_{\Omega}(\boldsymbol{\theta}) + L_{\partial\Omega}(\boldsymbol{\theta})$$

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

  $$\boldsymbol{G}(\boldsymbol{\theta}) = \boldsymbol{G}_{\Omega}(\boldsymbol{\theta}) + \boldsymbol{G}_{\partial\Omega}(\boldsymbol{\theta})$$

- **KFAC**

  $$G_{\Omega}(\boldsymbol{W}) \approx \boldsymbol{A}_{\Omega} \otimes \boldsymbol{B}_{\Omega}$$

  (generalized to diff. ops)

  $$G_{\partial\Omega}(\boldsymbol{W}) \approx \boldsymbol{A}_{\partial\Omega} \otimes \boldsymbol{B}_{\partial\Omega}$$

  [Martens and Grosse, 2015, ICML]

Algorithm, Experiments & Conclusion

Our KFAC-based optimizer outperforms popular methods, scales to large neural networks, and performs similarly to Hessian-free [Martens, 2010, ICML].

$$L(\boldsymbol{\theta}) = L_\Omega(\boldsymbol{\theta}) + L_{\partial\Omega}(\boldsymbol{\theta})$$

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

$$\boldsymbol{G}(\boldsymbol{\theta}) = \boldsymbol{G}_\Omega(\boldsymbol{\theta}) + \boldsymbol{G}_{\partial\Omega}(\boldsymbol{\theta})$$

- **KFAC**

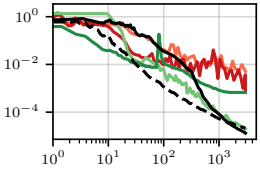
$$G_\Omega(\boldsymbol{W}) \approx \boldsymbol{A}_\Omega \otimes \boldsymbol{B}_\Omega$$

(generalized to diff. ops)

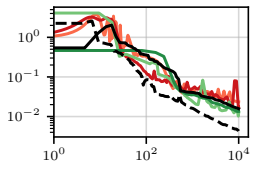
$$G_{\partial\Omega}(\boldsymbol{W}) \approx \boldsymbol{A}_{\partial\Omega} \otimes \boldsymbol{B}_{\partial\Omega}$$

[Martens and Grosse, 2015, ICML]

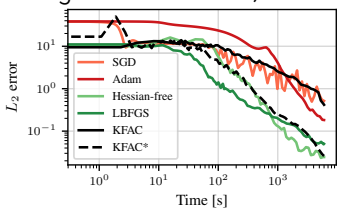
4+1d heat, $D \approx 10^5$



100d Poisson, $D \approx 10^6$



9+1d log Fokker-Planck, $D \approx 10^5$



References I

Runa Eschenhagen, Alexander Immer, Richard E. Turner, Frank Schneider, and Philipp Hennig. Kronecker-Factored Approximate Curvature for Modern Neural Network Architectures. In **Advances in Neural Information Processing Systems (NeurIPS)**, 2023.

James Martens. Deep learning via Hessian-free optimization. In **International Conference on Machine Learning (ICML)**, 2010.

James Martens and Roger Grosse. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In **International Conference on Machine Learning (ICML)**, 2015.

Johannes Müller and Marius Zeinhofer. Achieving high accuracy with PINNs via energy natural gradient descent. In **International Conference on Machine Learning (ICML)**, 2023.