



# Infinite Dimensional Optimization for SciML

Marius Zeinhofer

Strasbourg, France

20.05.2025

# Problem Setup

---

- Find  $u^*$  such that

$$u^* \in \operatorname{argmin}_{u \in \mathcal{MCH}} E(u, \nabla u, \nabla^2 u, \dots).$$

- Choose some ansatz

$$u \approx u_\theta, \quad \theta \in \Theta \subset \mathbb{R}^P \text{ trainable.}$$

Focus on *nonlinear parametrizations* like neural networks. We mean nonlinearity of

$$\theta \mapsto u_\theta.$$

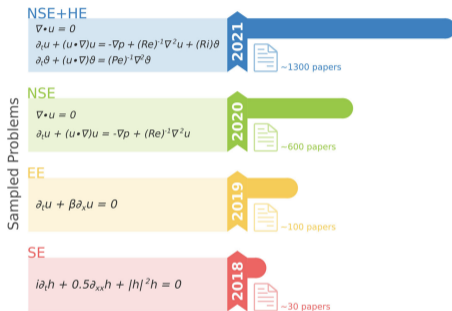
## Take-Home Message

*PDE* solvers for *nonlinear* ansatz classes deserve more attention.<sup>1</sup>

<sup>1</sup>J. Müller and M. Zeinhofer (2024). "Position: Optimization in SciML Should Employ the Function Space Geometry". In: *Forty-first International Conference on Machine Learning*.

# Neural PDE Solvers are Popular

- PINNs<sup>2</sup>
- Deep Ritz<sup>3</sup>
- Neural Galerkin<sup>4</sup> schemes
- Variational Monte Carlo with NN ansatz<sup>5</sup>.



<sup>2</sup>M. Raissi, P. Perdikaris, and G. E. Karniadakis (2019). "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations". In: *Journal of Computational Physics* 378, pp. 686–707.

<sup>3</sup>B. Yu et al. (2018). "The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems". In: *Communications in Mathematics and Statistics* 6.1, pp. 1–12.

<sup>4</sup>J. Bruna, B. Peherstorfer, and E. Vanden-Eijnden (2024). "Neural Galerkin schemes with active learning for high-dimensional evolution equations". In: *Journal of Computational Physics* 496, p. 112588.

<sup>5</sup>G. Carleo and M. Troyer (2017). "Solving the quantum many-body problem with artificial neural networks". In: *Science* 355.6325, pp. 602–606.

# Optimization is a Challenge

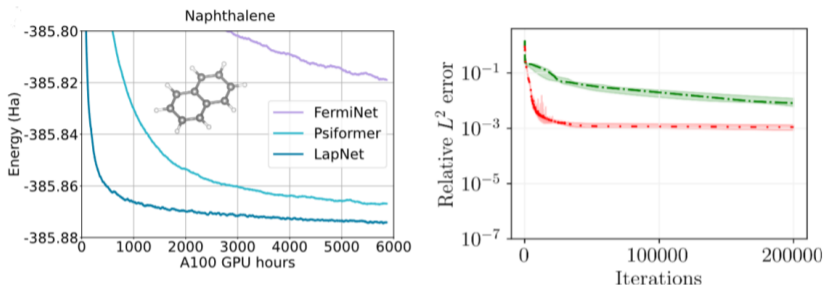


Figure: Illustration of neural network wavefunction optimization<sup>6</sup> and PINN optimization<sup>7</sup>.

## Empirical Observation

Convergence and accuracy problems are frequently encountered.

<sup>6</sup>R. Li et al. (2024). "A computational framework for neural network-based variational Monte Carlo with Forward Laplacian". In: *Nature Machine Intelligence*, pp. 1–11.

<sup>7</sup>J. Müller and M. Zeinhofer (2023). "Achieving High Accuracy with PINNs via Energy Natural Gradients". In: *International Conference on Machine Learning*.

# One Reason: Ill-Conditioning<sup>8</sup>

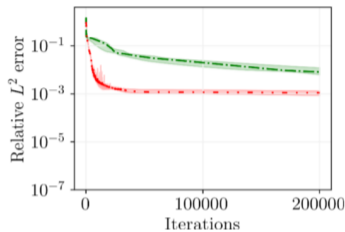
- Consider model problem with a linear PDE operator  $\mathcal{D}$

$$L(\theta) = \frac{1}{2} \int |\mathcal{D}u_\theta - f|^2 dx.$$

- Assume a linear ansatz  $\theta \mapsto u_\theta$ , then

$$L(\theta) = \frac{1}{2} \|G\theta - b\|^2, \quad G_{ij} = \int \mathcal{D}[\partial_{\theta_j} u_\theta] \cdot \mathcal{D}[\partial_{\theta_i} u_\theta] dx$$

- Gradient descent: Reach error  $\varepsilon$  in  $\mathcal{O}(\kappa(G) \log(\varepsilon^{-1}))$  steps, typically  $\kappa(G)$  large.



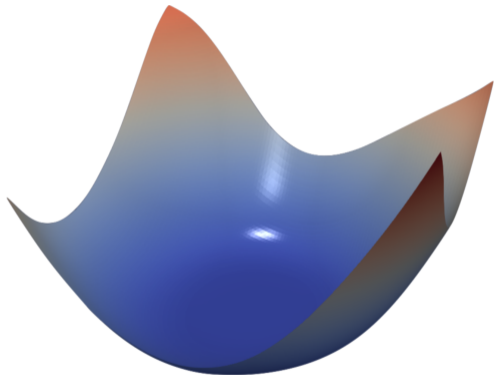
## Idea

Abandon first-order optimization. Exploit geometry induced by PDE operators.

<sup>8</sup>T. De Ryck, F. Bonnet, S. Mishra, and E. de Bézenac (2024). "An operator preconditioning perspective on training in physics-informed machine learning". In: *International Conference on Learning Representations*.

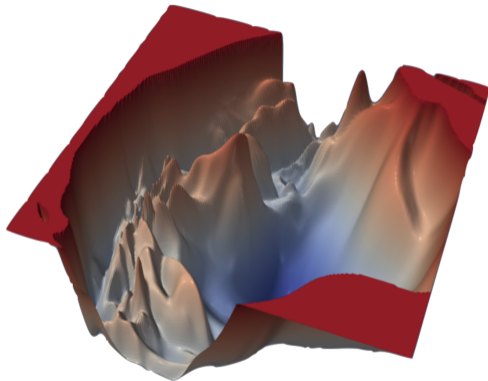
# Function Space vs Parameter Space Geometry<sup>9</sup>

---



$$E(u) = \frac{1}{2} \int |\mathcal{D}u - f|^2 dx$$

“Function Space”



$$L(\theta) = \frac{1}{2} \int |\mathcal{D}u_\theta - f|^2 dx$$

“Parameter Space”

---

<sup>9</sup>H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein (2018). “Visualizing the loss landscape of neural nets”. In: *Advances in neural information processing systems* 31.

# Table of Contents

---

## 1. Infinite Dimensional Optimization

## 2. Algorithmic Aspects

# Abstract Function Space Optimization<sup>10</sup>

## Informal Roadmap: “First Optimize, Then Discretize”

(i) **Continuous Formulation:** Formulate problem in a Hilbert space  $\mathcal{H}$

$$E : \mathcal{H} \rightarrow \mathbb{R}.$$

(ii) **Optimize:** Decide for an appropriate iterative algorithm *in function space*  $\mathcal{H}$

$$u_{k+1} = u_k + d_k.$$

(iii) **Discretize:** Project  $d_k$  on the tangent space of neural network ansatz. Yields

$$\theta_{k+1} = \theta_k + \delta\theta_k \quad \text{with} \quad u_{\theta_{k+1}} \approx u_{k+1}$$

<sup>10</sup>J. Müller and M. Zeinhofer (2024). “Position: Optimization in SciML Should Employ the Function Space Geometry”. In: *Forty-first International Conference on Machine Learning*.

# Newton: Optimize

---

- Newton converges in one step on a quadratic functional like<sup>11</sup>

$$\min_{u \in H^2(\Omega) \cap H_0^1(\Omega)} E(u) = \frac{1}{2} \int_{\Omega} (\Delta u + f)^2 dx.$$

- Starting at  $u_0$ , we find the minimizer  $u^*$  as

$$\begin{aligned} u^* &= u_0 - D^2 E(u_0)^{-1} [DE(u_0)] \\ &= u_0 + d. \end{aligned}$$

- Use the equation  $d = -D^2 E(u_0)^{-1} [DE(u_0)]$  to do a Galerkin projection.

---

<sup>11</sup>Mathematically, a first-order least squares formulation is preferable.

# Newton: Discretize

---

- The projection of  $d = -D^2 E(u_0)^{-1}[DE(u_0)]$  is computed via a Galerkin discretization using the linear space

$$T_{u_\theta} \mathcal{M} = \text{span}\{\partial_{\theta_1} u_\theta, \dots, \partial_{\theta_p} u_\theta\} \quad \text{where} \quad \mathcal{M} = \{u_\theta \mid \theta \in \Theta\}.$$

- Hence the projected update  $\delta\theta$  is given as the solution of

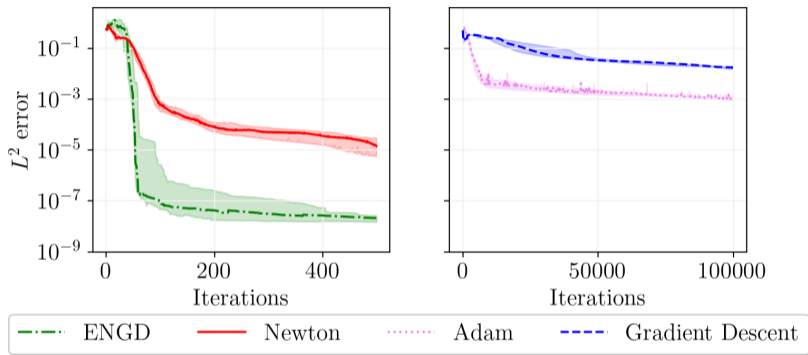
$$G(\theta)\delta\theta = -\nabla L(\theta)$$

- Where  $\nabla L(\theta)_i = DE(u_\theta)(\partial_{\theta_i} u_\theta)$  is the Euclidean gradient and

$$G(\theta)_{ij} = D^2 E(u_\theta)(\partial_{\theta_j} u_\theta, \partial_{\theta_i} u_\theta) = \int_{\Omega} \Delta \partial_{\theta_j} u_\theta \Delta \partial_{\theta_i} u_\theta \, dx.$$

- Natural Gradient:  $G(\theta)$  is the pull-back of  $D^2 E(u_\theta)$  along  $\theta \mapsto u_\theta$  in  $(\partial_{\theta_i} u_\theta)_i$ .

# Result: Efficient Natural Gradient Optimizer<sup>12</sup>

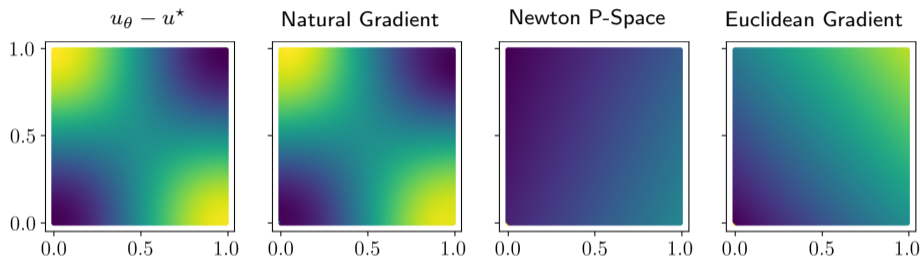


$$\theta_{k+1} = \theta_k - \eta_k G(\theta_k)^\dagger \nabla L(\theta_k), \quad k = 0, 1, 2 \dots$$

$$G(\theta)_{ij} = \int_{\Omega} \Delta \partial_{\theta_i} u_{\theta} \Delta \partial_{\theta_j} u_{\theta} dx$$

<sup>12</sup>J. Müller and M. Zeinhofer (2023). "Achieving High Accuracy with PINNs via Energy Natural Gradients". In: *International Conference on Machine Learning*.

# Visualization: The Projection Property<sup>13</sup>



- Exact Newton  $d = u_\theta - u^*$  (gold-standard for reference, unavailable in applications)
- Natural Gradient/Projected Newton direction  $G(\theta)^\dagger \nabla L(\theta)$
- Newton parameter space  $\nabla^2 L(\theta)^\dagger \nabla L(\theta)$
- Euclidean gradient  $\nabla L(\theta)$

<sup>13</sup>J. Müller and M. Zeinhofer (2024). "Position: Optimization in SciML Should Employ the Function Space Geometry". In: *Forty-first International Conference on Machine Learning*.

# Interpretation as Generalized Gauss-Newton

---

- We can relate the loss Hessian  $\nabla^2 L(\theta)$  and the matrix  $G(\theta)$ :

$$\nabla^2 L(\theta)_{ij} = \underbrace{D^2 E(u_\theta)(\partial_{\theta_i} u_\theta, \partial_{\theta_j} u_\theta)}_{G(\theta)_{ij}} + DE(u_\theta)[\partial_{\theta_i} \partial_{\theta_j} u_\theta]$$

- Using  $G(\theta) \approx \nabla^2 L(\theta)$  is known as the generalized Gauss-Newton approximation.
- Empirically: In neural network context GGN is always much better than full Newton.

## Wrap-up Newton

- Start with functional version.
- Project the directions using a Galerkin scheme in tangent space.
- Interpretation: Projected Newton/natural gradient/generalized Gauss-Newton.

# Further Examples: Variational Monte Carlo<sup>14</sup>

Goal: Find ground state wavefunction for a molecular system with Hamiltonian  $H$

$$\min_{\psi \neq 0} E(\psi) = \frac{\langle \psi | H | \psi \rangle}{\langle \psi | \psi \rangle}$$

## The Hamiltonian

Denote by  $R_I$  and  $Z_I$  the positions and charges of  $M$  atomic nuclei and  $r_i \in \mathbb{R}^3$  are the position of the  $N$  electrons.

$$H = -\frac{1}{2} \sum_{i=1}^N \Delta_{r_i} - \sum_{i=1}^N \sum_{I=1}^M \frac{Z_I}{|r_i - R_I|} + \sum_{i < j} \frac{1}{|r_i - r_j|} + \sum_{I < J} \frac{Z_I Z_J}{|R_I - R_J|}$$

<sup>14</sup>D. Pfau, J. S. Spencer, A. G. Matthews, and W. M. C. Foulkes (2020). "Ab Initio Solution of the Many-Electron Schrödinger Equation with Deep Neural Networks". In: *Physical Review Research* 2.3, p. 033429.

# VMC: Optimize

---

- The objective is to minimize the Rayleigh quotient

$$E(\psi) = \frac{\langle \psi | H | \psi \rangle}{\langle \psi | \psi \rangle} = \mathbb{E}_p \left[ \frac{H\psi}{\psi} \right], \quad p = \psi^2 / \|\psi\|^2.$$

- Optimize via  $L^2$  gradient flow on the sphere  $\mathbb{S} = \{\psi : \|\psi\| = 1\}$

$$\psi'(t) = DE(\psi(t)) = 2\langle H\psi(t), \cdot \rangle \quad \text{in } T_{\psi(t)}^* \mathbb{S}.$$

- Choose explicit time discretization and retraction  $\psi \mapsto \psi / \|\psi\|$

$$\psi_{k+\frac{1}{2}} = \psi_k - \eta_k R_k^{-1}[DE(\psi_k)], \quad \psi_k = \frac{\psi_{k+\frac{1}{2}}}{\|\psi_{k+\frac{1}{2}}\|},$$

where  $R_k$  is the Riesz isometry of  $T_{\psi_k} \mathbb{S}$  as a subspace of  $L^2$ .

# VMC: Discretize

---

- Choose an ansatz  $\{\psi_\theta \mid \theta \in \Theta\}$ <sup>15</sup> and define the loss function  $L(\theta) = E(\psi_\theta)$ .
- Galerkin discretize  $R_k^{-1}[DE(\psi_k)]$  with space spanned by

$$\partial_{\theta_i} \left( \frac{\psi_\theta}{\|\psi_\theta\|} \right) = \frac{1}{\|\psi_\theta\|} \left[ \partial_{\theta_i} \psi_\theta - \frac{\langle \partial_{\theta_i} \psi_\theta, \psi_\theta \rangle}{\|\psi_\theta\|^2} \psi_\theta \right] \in T_{\psi_\theta} \mathbb{S}, \quad i = 1, \dots, p$$

- This yields “stochastic reconfiguration”<sup>16</sup>

$$\theta_{k+1} = \theta_k - \eta_k (S(\theta_k) + \lambda_k I)^{-1} \nabla L(\theta_k), \quad k = 0, 1, \dots$$

where  $S(\theta_k)$  represents the  $L^2$  inner product in the generating system above.

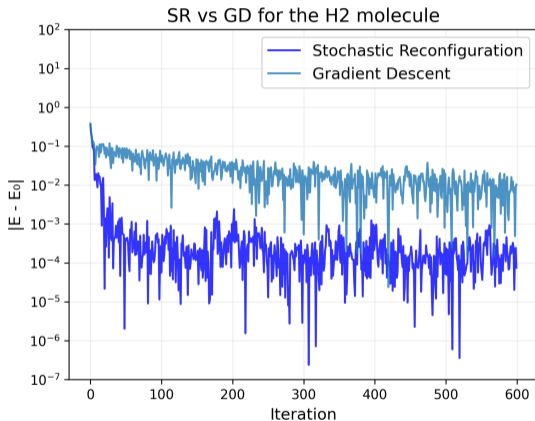
---

<sup>15</sup>Here, one encodes information about the molecule at hand.

<sup>16</sup>S. Sorella (1998). “Green function Monte Carlo with stochastic reconfiguration”. In: *Physical review letters* 80.20, p. 4558.

# Result: An Efficient Optimization Method

---



$$\theta_{k+1} = \theta_k - \eta_k (S(\theta_k) + \lambda_k I)^{-1} \nabla L(\theta_k), \quad k = 0, 1, \dots$$

# What about Newton Methods?

---

- A shifted Riemannian Newton method for the Rayleigh quotient with shift  $\tau_k$  is given by

$$d_k = -(D^2 E(\psi_k) + \tau_k I)^{-1}[DR(\psi_k)],$$
$$\psi_{k+1} = \frac{\psi_k + d_k}{\|\psi_k + d_k\|}$$

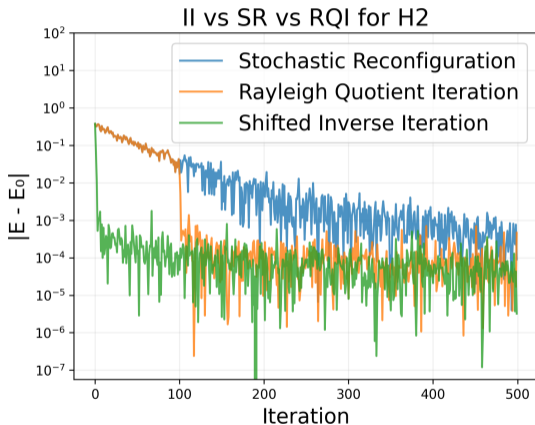
- The linear system in the tangent space is

$$\langle [\hat{H} + (\tau_k - E(\psi_k))I]d_k, v \rangle = -\langle \hat{H}\psi_k, v \rangle \quad \text{for all } v \in T_{\psi_k}\mathbb{S}.$$

- The choice  $\tau_k = 0$  yields Rayleigh quotient iteration.
- The choice  $\tau_k = E(\psi_k) - \mu$  yields inverse iteration with shift  $\mu$ .

# Result: Even Faster Convergence

---



$$\theta_{k+1} = \theta_k - \eta_k (H(\theta_k) - E_0 S(\theta_k) + \lambda_k I)^{-1} \nabla L(\theta_k), \quad k = 0, 1, \dots$$

# Summary

---

Connections between function space and parameter space optimization<sup>17</sup>

Function Space	Parameter Space
Newton	Generalized Gauss-Newton <sup>18</sup>
Gauss-Newton	Gauss-Newton <sup>19</sup>
Time Stepping	Neural Galerkin <sup>20</sup>
$L^2$ -GD on Sphere	Stochastic Reconfiguration
SQP	Competitive Gradient Descent <sup>21</sup>

---

<sup>17</sup>J. Müller and M. Zeinhofer (2024). "Position: Optimization in SciML Should Employ the Function Space Geometry". In: *Forty-first International Conference on Machine Learning*.

<sup>18</sup>J. Müller and M. Zeinhofer (2023). "Achieving High Accuracy with PINNs via Energy Natural Gradients". In: *International Conference on Machine Learning*.

<sup>19</sup>A. Jnini, F. Vella, and M. Zeinhofer (2024). "Gauss-Newton Natural Gradient Descent for Physics-Informed Computational Fluid Dynamics". In: *arXiv preprint arXiv:2402.10680*.

<sup>20</sup>J. Bruna, B. Peherstorfer, and E. Vanden-Eijnden (2024). "Neural Galerkin schemes with active learning for high-dimensional evolution equations". In: *Journal of Computational Physics* 496, p. 112588.

<sup>21</sup>F. Schäfer and A. Anandkumar (2019). "Competitive gradient descent". In: *Advances in Neural Information Processing Systems* 32.

# Table of Contents

---

1. Infinite Dimensional Optimization

**2. Algorithmic Aspects**

# The Price of Non-local Ansatz Functions

---

## Newton's Method

In every step the following matrix needs to be assembled and inverted

$$G(\theta)_{ij} = D^2 E(u_\theta)(\partial_{\theta_i} u_\theta, \partial_{\theta_j} u_\theta), \quad i, j = 1, \dots, P.$$

For neural network ansatz functions the matrix is dense, rank deficient and ill-conditioned.

- A direct solve works well for networks of modest size, say  $P < 10^4$  trainable parameters
- Storing requires  $\mathcal{O}(P^2)$  memory and the solve  $\mathcal{O}(P^3)$  computations.

# What are our Options?

---

## Computational Bottleneck

Assembling  $G(\theta) \in \mathbb{R}^{P \times P}$  and solving  $G(\theta)d = -\nabla L(\theta)$ .

- Matrix-free solvers<sup>22, 23</sup>
- Exploit low-rank structure originating from small batch-sizes (minSR).<sup>24</sup>
- Derive cheap-to-invert approximations of  $G(\theta)$ , e.g. using the Kronecker-factored approximate curvature (KFAC) approximation<sup>25, 26</sup>

---

<sup>22</sup>N. N. Schraudolph (2002). “Fast curvature matrix-vector products for second-order gradient descent”. In: *Neural computation* 14.7, pp. 1723–1738.

<sup>23</sup>A. Jnini, F. Vella, and M. Zeinhofer (2024). “Gauss-Newton Natural Gradient Descent for Physics-Informed Computational Fluid Dynamics”. In: *arXiv preprint arXiv:2402.10680*.

<sup>24</sup>A. Chen and M. Heyl (2023). “Empowering deep neural quantum states through efficient optimization”. In: *Nature Physics*.

<sup>25</sup>J. Martens (2020). “New Insights and Perspectives on the Natural Gradient Method”. In: *The Journal of Machine Learning Research* 21.1, pp. 5776–5851.

<sup>26</sup>F. Dangel, J. Müller, and M. Zeinhofer (2024). “Kronecker-Factored Approximate Curvature for Physics-Informed Neural Networks”. In: *NeurIPS*.

# Structure of the Gramian

---

- Consider PINN formulation of Poisson equation with hard-coded boundary conditions

$$L(\theta_k) = \frac{1}{2N_\Omega} \sum_{i=1}^{N_\Omega} (\Delta u_{\theta_k}(x_i) + f(x_i))^2 = \frac{1}{2N_\Omega} \|r(\theta_k)\|^2$$

where  $r : \Theta = \mathbb{R}^P \rightarrow \mathbb{R}^N$  with  $r(\theta)_i = \Delta u_\theta(x_i) + f(x_i)$ .

- The Gramian in this case is given by

$$G(\theta_k) = \frac{1}{N_\Omega} \sum_{n=1}^{N_\Omega} \partial_\theta \Delta u_{\theta_k}(x_n)^\top \partial_\theta \Delta u_{\theta_k}(x_n) = \frac{1}{N_\Omega} J_k^\top J_k$$

where  $J_k \in \mathbb{R}^{N \times P}$  is the Jacobian of the residual, i.e.,  $J_k = J_\theta r(\theta_k)$ . Set also  $r_k = r(\theta_k)$  and note  $\nabla L(\theta_k) = J_k^\top r_k$ .

# Leveraging Small Batch Sizes

- The natural gradient scheme with damping  $\lambda_k > 0$  and step size  $\eta_k$  thus is

$$\theta_{k+1} = \theta_k - \eta_k \underbrace{(J_k^\top J_k + \lambda_k I)^{-1}}_{\in \mathbb{R}^{P \times P}} J_k^\top r_k.$$

## Natural Gradients in Sample Space

If  $N \ll P$  we can use the push-through identity  $(AB + \lambda I)^{-1}A = A(BA + \lambda I)^{-1}$

$$\underbrace{(J_k^\top J_k + \lambda_k I)^{-1}}_{\in \mathbb{R}^{P \times P}} J_k^\top r_k = J_k^\top \underbrace{(J_k J_k^\top + \lambda_k I)^{-1}}_{\in \mathbb{R}^{N \times N}} r_k$$

and improve the complexity drastically from  $\mathcal{O}(P^3)$  to  $\mathcal{O}(N^2P)$ .

# Considerations on Efficiency

---

- Computation of the *neural tangent kernel matrix*  $J_k J_k^\top$  is the bottleneck  $\mathcal{O}(N^2 P)$ .
- The matrix is a sum over the parameters

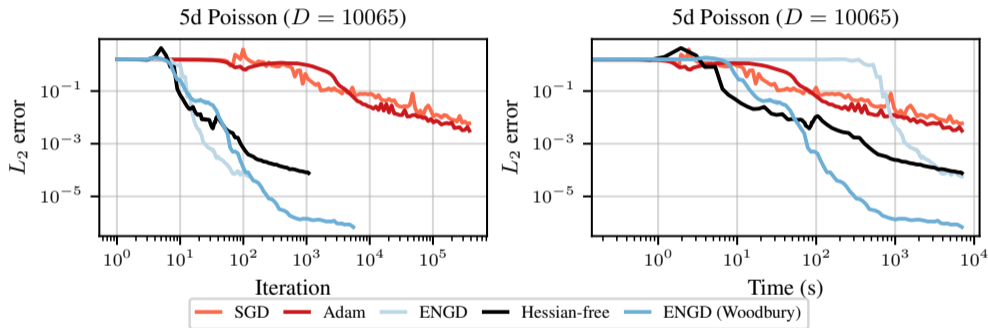
$$(J_k J_k^\top)_{ij} = \sum_{\theta=1}^P \partial_{\theta_n} u_\theta(x_i) \partial_{\theta_n} u_\theta(x_j)$$

- Looping over layers, contracting and summing the results works reasonably well – allows parameters of order  $10^6$  and batch-sizes around  $10^3$ .
- Certainly not the last word<sup>27</sup>.

---

<sup>27</sup>R. Novak, J. Sohl-Dickstein, and S. S. Schoenholz (2022). “Fast finite width neural tangent kernel”. In: *International Conference on Machine Learning*. PMLR, pp. 17018–17044.

# Numerical Results



- We observe significant speed-ups. Batchsize is  $N = N_{\Omega} + N_{\partial\Omega} = 3500$  in this case.

# Including “Momentum”

---

- Empirically<sup>28</sup> we know, that exponential moving averages are crucial to transport information between iterations, however meaningless in sample space.
- To transport information between iterations, realize that:

$$J_k^\top (J_k J_k^\top + \lambda I)^{-1} r_k = \operatorname{argmin}_\phi \|J_k \phi - r_k\|^2 + \lambda_k \|\phi\|^2. \quad (1)$$

- Now include the previous iterate<sup>29</sup>

$$\phi_k = \operatorname{argmin}_\phi \|J_k \phi - r_k\|^2 + \lambda_k \|\phi - \mu \phi_{k-1}\|^2.$$

- This yields

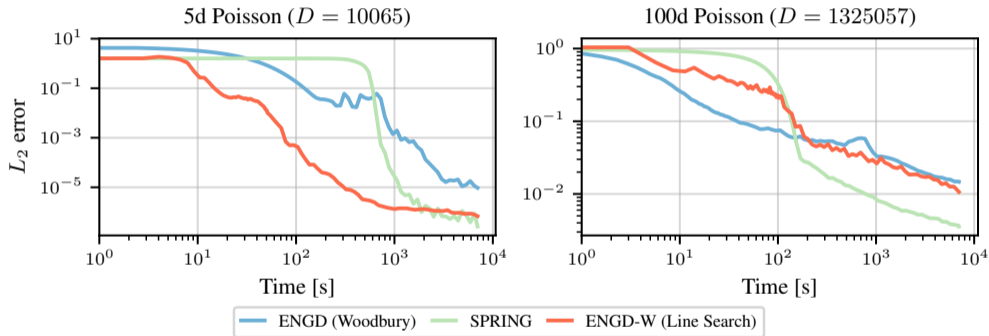
$$\phi_k = \mu \phi_{k-1} + J_k^\top (J_k J_k^\top + \lambda I)^{-1} (r_k - \mu J_k \phi_{k-1}),$$

---

<sup>28</sup>F. Dangel, J. Müller, and M. Zeinhofer (2024). “Kronecker-Factored Approximate Curvature for Physics-Informed Neural Networks”. In: *NeurIPS*.

<sup>29</sup>G. Goldshlager, N. Abrahamsen, and L. Lin (2024). “A Kaczmarz-inspired approach to accelerate the optimization of neural network wavefunctions”. In: *arXiv preprint arXiv:2401.10190*.

# Numerical Results



- Momentum helps in high-dimensional, noisy settings.

## Further Aspects

---

- We investigated “Nyström-on-the-kernel” to further accelerate the method, sketching the kernel via

$$J_k J_k^\top \approx \text{nys}(J_k J_k^\top).$$

- For sketch size  $S < N$  and random matrix  $\Omega \in \mathbb{R}^{N \times S}$ , the Nyström approximation is

$$\text{nys}(J_k J_k^\top) = J_k J_k^\top \underbrace{\Omega (\Omega^\top J_k J_k^\top \Omega)^\dagger}_{\in \mathbb{R}^{S \times S}} (J_k J_k^\top \Omega)^\top.$$

- GPU efficient implementation requires care, needs to avoid SVDs at all cost, we tweaked<sup>30</sup> for optimal performance, details in pre-print<sup>31</sup>.

---

<sup>30</sup>Z. Frangella, J. A. Tropp, and M. Udell (2023). “Randomized nyström preconditioning”. In: *SIAM Journal on Matrix Analysis and Applications* 44.2, pp. 718–752.

<sup>31</sup>A. Guzmán-Cordero, F. Dangel, G. Goldshlager, and M. Zeinhofer (2025). “Improving Energy Natural Gradient Descent through Woodbury, Momentum, and Randomization”. In: <https://arxiv.org/abs/2505.12149>.

# Conclusion

---

## Function Space Viewpoint

Blueprint for the design of optimization methods for loss functions involving PDE terms and nonlinear ansatz spaces<sup>32</sup>.

- Recent, successful methods can be understood from this angle.
- Many areas are untouched: Quasi-Newton methods, PDE constrained optimization, operator preconditioning for multiphysics, ...
- Talk/email me if you are interested in projects in this direction!
- Thank you for your attention!

---

<sup>32</sup>J. Müller and M. Zeinhofer (2024). "Position: Optimization in SciML Should Employ the Function Space Geometry". In: *Forty-first International Conference on Machine Learning*.