Gauss-Newton Natural Gradient Descent for Physics-Informed Computational Fluid Dynamics

Anas Jnini

Joint work with *Flavio Vella* and *Marius Zeinhofer*

University of Trento

Dipartimento di Ingegneria e Scienza dell'Informazione

February 22, 2024

Table of Contents

Physics-Informed Neural Networks (PINNs)

Consider a general partial differential equation (PDE) given by:

$$\mathcal{L}u = f \text{ in } \Omega \tag{1}$$

$$\mathcal{B}u = g \text{ on } \partial\Omega, \tag{2}$$

where $\Omega \subseteq \mathbb{R}^d$ is an open set, $\mathcal{L}$ is a differential operator, and $\mathcal{B}$ is a boundary operator. The solution u is sought in a Hilbert space X with f and g being square integrable on Ω and $\partial\Omega$ respectively.

Physics-Informed Neural Networks (PINNs) - Continued

We formulate minimization problem with the objective function:

$$E(u) = \int_{\Omega} (\mathcal{L}u - f)^2 \, dx + \tau \int_{\partial\Omega} (\mathcal{B}u - g)^2 \, ds, \tag{3}$$

To approximate a solution, we parametrize u by a neural network u_θ and optimize the network parameters $\theta \in \mathbb{R}^p$ by minimizing the loss function:

$$\mathcal{L}(\theta) := \int_{\Omega} (\mathcal{L}u_\theta - f)^2 \, dx + \tau \int_{\partial\Omega} (\mathcal{B}u_\theta - g)^2 \, ds. \tag{4}$$

one can approximate $E(u_\theta)$ by collocation

$$\mathcal{L}(\theta) = \frac{|\Omega|}{2N_\Omega} \sum_{i=1}^{N_\Omega} (\mathcal{L}u_\theta - f)^2 + \frac{|\partial\Omega|}{2N_{\partial\Omega}} \sum_{i=1}^{N_{\partial\Omega}} (\mathcal{B}u_\theta - g)^2.$$

Optimization Challenges in PINNs

- Loss minimization in PINNs is commonly done using first-order optimizers like Adam or SGD, L-BFGS. Challenges include long training times and modest accuracy.
- Errors below 10^{-4} in relative L^2 norm are rare [2, 4, 6].
- Recent studies focus on addressing accuracy for linear PDEs, but highly accurate solutions for nonlinear PDEs are still lacking [6, 5].

How to best minimize the Loss function

Consider the Energy minimization problem,

$$E(u) = \int_{\Omega} (\mathcal{L}u - f)^2 \, dx + \tau \int_{\partial\Omega} (\mathcal{B}u - g)^2 \, ds, \tag{5}$$

the goal is to decide for an iterative algorithm

$$u_{k+1} = u_k + \eta_k d_k, \quad k = 0, 1, 2, \ldots$$

that is "appropriate" for minimization of E on the function space $\mathcal{H}$.

Energy Natural Gradients

Energy Natural Gradient descent[1] is an approach to precondition the gradient in the gradient using Newton's method in function space:

$$
\begin{aligned}
u_{k+1} &= u_k - D^2E(u_k)^{-1}[DE(u_k)] \\
&= u_k + d_k.
\end{aligned}
$$

[1] Johannes Müller and Marius Zeinhofer. "Achieving High Accuracy with PINNs via Energy Natural Gradient Descent". In: *ICML* (2023).

Energy Natural Gradient Descent: Galerkin in Tangent Space

- Neural network ansatz, tangent space at a function u_θ and loss:

$$\mathcal{M} = \{u_\theta \mid \theta \in \Theta\}, \quad T_{u_\theta}\mathcal{M} = \operatorname{span}\{\partial_{\theta_1} u_\theta, \dots, \partial_{\theta_p} u_\theta\}, \quad L(\theta) = E(u_\theta)$$

- Discretize Newton using a Galerkin ansatz with the tangent vector-space $T_{u_\theta}\mathcal{M}$:

$$\begin{aligned} D^2E(u_\theta) &\approx G(\theta), & G(\theta)_{ij} &= D^2E(u_\theta)(\partial_{\theta_j} u_\theta, \partial_{\theta_i} u_\theta) \\ DE(u_\theta) &\approx \nabla L(\theta), & \nabla L(\theta)_i &= DE(u_\theta)(\partial_{\theta_i} u_\theta) \end{aligned}$$

For a linear PDE operator $\mathcal{L}$, the residual yields a quadratic energy and the energy Gram matrix takes the form:

$$G(\theta)_{ij} = \int_\Omega \mathcal{L}(\partial_{\theta_i} u_\theta)\mathcal{L}(\partial_{\theta_j} u_\theta)\, dx + \tau \int_{\partial\Omega} \mathcal{B}(\partial_{\theta_i} u_\theta)\mathcal{B}(\partial_{\theta_j} u_\theta)\, ds \qquad (6)$$

When discretized, the Gramian takes the form

$$G(\theta) \approx \frac{1}{N}\sum_{k=0}^{N} J(\mathcal{L}(u_{\theta_k}))\cdot J(\mathcal{L}(u_{\theta_k}))^T + \frac{\tau}{M}\sum_{k=0}^{M} J(\mathcal{B}(u_{\theta_k}))\cdot J(\mathcal{B}(u_{\theta_k}))^T \quad (7)$$

This yields an optimization algorithm :

$$\theta_{k+1} = \theta_k - \eta_k G(\theta_k)^{\dagger}\nabla L(\theta_k).$$

Instead of computing the pseudo-inverse of the Gram matrix $G_E(\theta)$, we solve a least square problem to find the argument that minimizes $||G_E(\theta)\psi - \nabla L(\theta)||_2^2$.

ENGD: Poisson equation

- Suppose we aim to solve Poisson's equation

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega \end{aligned}$$

- Reformulate, for instance, as an minimization problem[2]

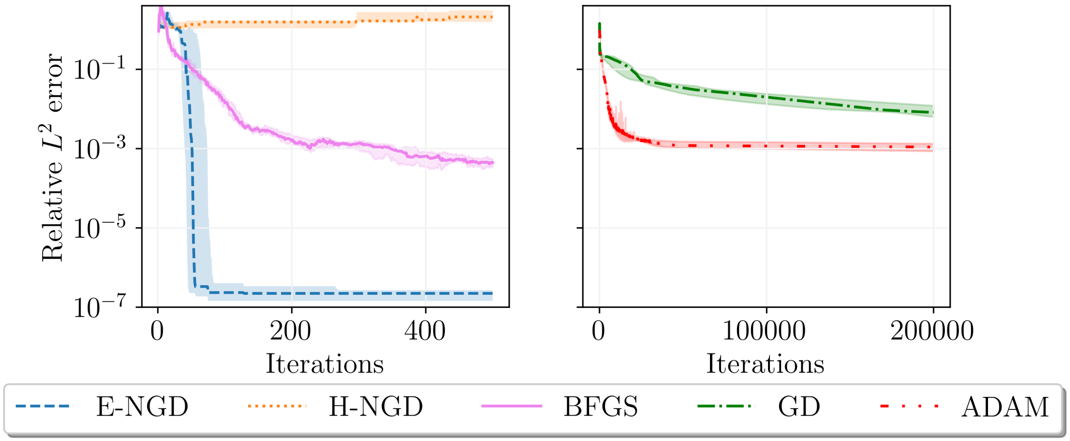
$$\min_{u \in H^2(\Omega)} E(u) = \frac{1}{2}\|\Delta u + f\|^2_{L^2(\Omega)} + \frac{1}{2}\|u - g\|^2_{L^2(\partial\Omega)}$$

- If $E(u) = \frac{1}{2}\|\Delta u + f\|^2_{L^2(\Omega)} + \frac{1}{2}\|u - g\|^2_{L^2(\partial\Omega)}$ we get:

$$G(\theta)_{ij} = \int_\Omega \Delta\partial_{\theta_i} u_\theta \Delta\partial_{\theta_j} u_\theta \,\mathrm{dx} + \int_{\partial\Omega} \partial_{\theta_i} u_\theta \partial_{\theta_j} u_\theta \mathrm{ds}$$

[2] A formulation as a root finding problem or a variational formulation make sense, too.

Energy Natural Gradient Descent: Results



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

$$G(\theta)_{ij} = \int_\Omega \Delta\partial_{\theta_i} u_\theta \Delta\partial_{\theta_j} u_\theta \,\mathrm{d}x + \int_{\partial\Omega} \partial_{\theta_i} u_\theta \partial_{\theta_j} u_\theta \mathrm{d}s$$

- **Architecture:** One shallow layer of width 64.
- **Initialization:** 10 different initializations.
- **Optimizer:** Adam optimizer with exponential decaying learning rate.

ENGD: Recap

- ENGD is designed for linear PDEs, for which it is highly effective
- For non linear PDEs : Highly non-convex energy term E, G is non Symmetric-Definite Positive by construction: not adapted to Newton's Method.
- The method does not scale favorably with the parameter space.

Gauss-Newton Natural Gradient Descent for Physics-Informed Computational Fluid Dynamics

To address these issues we propose Gauss-Newton Natural Gradient Descent for Physics-Informed Computational Fluid Dynamic[3].

- **Gauss-Newton in Function Space:** We propose a second-order optimization in Function Space with unprecedented accuracy in nonlinear PDEs.
- **Matrix-Free and Scalable:** We propose a formulation that adapts to large network sizes through a matrix-free formulation, ensuring scaling to large neural networks.

[3] Anas Jnini, Flavio Vella, and Marius Zeinhofer. *Gauss-Newton Natural Gradient Descent for Physics-Informed Computational Fluid Dynamics*. 2024. arXiv: `2402.10680 [math.OC]`.

GNNG I: ∞-Dimensional Formulation

- Suppose we aim to solve the Navier-Stokes equations

$$\begin{aligned} -\Delta u + (u \cdot \nabla) u + \nabla p &= f \quad \text{in } \Omega, \\ \operatorname{div} u &= 0 \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega. \end{aligned}$$

- Reformulate, for instance, as a minimization problem[4]

$$\min_{(u,p)\in\mathcal{H}} E(u,p) = \frac{1}{2}\| -\Delta u + (u\cdot\nabla)u + \nabla p - f\|^2_{L^2(\Omega)}$$

[4] For simplicity assume solenoidal ansatz functions that respect the boundary values.

GNNG II: Choose Function-Space Algorithm

- The functional

$$E(u,p) = \frac{1}{2}\|R(u,p)\|^2_{L^2(\Omega)} = \frac{1}{2}\| -\Delta u + (u \cdot \nabla)u + \nabla p - f\|^2_{L^2(\Omega)}$$

  yields a *nonlinear least-squares problem*.

GNNG II: Choose Function-Space Algorithm

- The functional

$$E(u,p) = \frac{1}{2}\|R(u,p)\|^2_{L^2(\Omega)} = \frac{1}{2}\| -\Delta u + (u\cdot\nabla)u + \nabla p - f\|^2_{L^2(\Omega)}$$

  yields a *nonlinear least-squares problem*.
- We choose Gauss-Newton's method in function space

$$\begin{pmatrix} u_{k+1} \\ p_{k+1} \end{pmatrix} = \begin{pmatrix} u_k \\ p_k \end{pmatrix} - [DR(u_k,p_k)^* DR(u_k,p_k)]^{-1} DE(u_k,p_k).$$

GNNG II: Choose Function-Space Algorithm

- Gauss-Newton's method in function space is

$$\begin{pmatrix} u_{k+1} \\ p_{k+1} \end{pmatrix} = \begin{pmatrix} u_k \\ p_k \end{pmatrix} - [DR(u_k, p_k)^* DR(u_k, p_k)]^{-1} DE(u_k, p_k).$$

GNNG II: Choose Function-Space Algorithm

- Gauss-Newton's method in function space is

$$\begin{pmatrix} u_{k+1} \\ p_{k+1} \end{pmatrix} = \begin{pmatrix} u_k \\ p_k \end{pmatrix} - [DR(u_k, p_k)^* DR(u_k, p_k)]^{-1} DE(u_k, p_k).$$

- We linearize R around the current iterate and explicitly solve the resulting quadratic minimization problem . We obtain $DR(u,p)(\delta_u, \delta_p) = -\Delta\delta_u + (u \cdot \nabla)\delta_u + (\delta_u \cdot \nabla)u + \nabla\delta_p$ and

$$DR(u,p)^* DR(u,p)(\delta_u, \delta_p) = (DR(u,p)(\delta_u, \delta_p), DR(u,p)(\cdot,\cdot))_{L^2(\Omega)}$$

GNNG III: Galerkin in Tangent Space

- Neural network ansatz and tangent space at (u_θ, p_ψ):

$$\mathcal{M} = \{(u_\theta, p_\theta) \mid \theta \in \Theta\},$$

$$\mathrm{T}_{(u_\theta, p_\theta)}\mathcal{M} = \mathrm{span}\{\partial_{\theta_1}(u_\theta, p_\theta), \dots, \partial_{\theta_p}(u_\theta, p_\theta)\}$$

- Discretize Gauss-Newton using a Galerkin ansatz:

$$DR(u_\theta, p_\theta)DR(u_\theta, p_\theta)^* \approx G(\theta), \quad DE(u_\theta) \approx \nabla L(\theta)$$

with

$$G(\theta)_{ij} = (DR(u_\theta, p_\theta)\partial_{\theta_i}(u_\theta, p_\theta), DR(u_\theta, p_\theta)\partial_{\theta_j}(u_\theta, p_\theta))_{L^2(\Omega)}$$

- This yields an optimization algorithm:

$$\theta_{k+1} = \theta_k - \eta_k G(\theta_k)^\dagger \nabla L(\theta_k).$$

GNNG: Gramian Block structure

The matrix $G(\theta, \psi)$ has block structure

$$G(\theta, \psi) = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}.$$

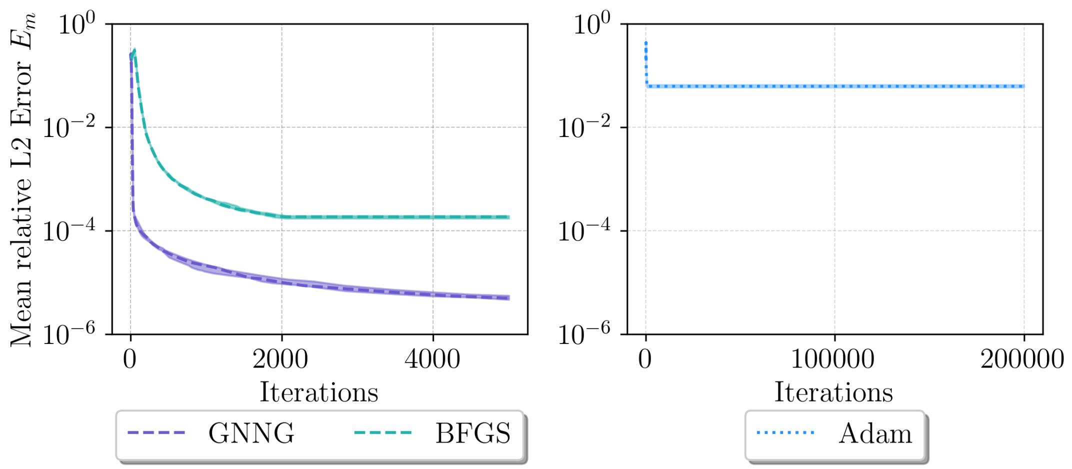
For the case of the stationary Navier-Stokes equations, the blocks are given by

$$\begin{aligned} A_{ij} &= (-\nu \Delta \partial_{\theta_j} u_\theta + (\partial_{\theta_j} u_\theta \cdot \nabla) u_\theta + (u_\theta \cdot \nabla) \partial_{\theta_j} u_\theta, -\nu \Delta \partial_{\theta_i} u_\theta + (\partial_{\theta_i} u_\theta \cdot \nabla) u_\theta \\ &\quad + (\mathrm{div}(\partial_{\theta_j} u_\theta), \mathrm{div}(\partial_{\theta_i} u_\theta))_{L^2(\Omega)} + (\partial_{\theta_j} u_\theta, \partial_{\theta_i} u_\theta)_{L^2(\Omega)}, \\ B_{ij} &= (-\nu \Delta \partial_{\theta_j} u_\theta + (\partial_{\theta_j} u_\theta \cdot \nabla) u_\theta + (u_\theta \cdot \nabla) \partial_{\theta_j} u_\theta, \nabla \partial_{\psi_i} p_\psi)_{L^2(\Omega)} \\ C_{ij} &= (\nabla \partial_{\psi_j} p_\psi, \nabla \partial_{\psi_i} p_\psi)_{L^2(\Omega)}. \end{aligned}$$

GNNG: Algorithm

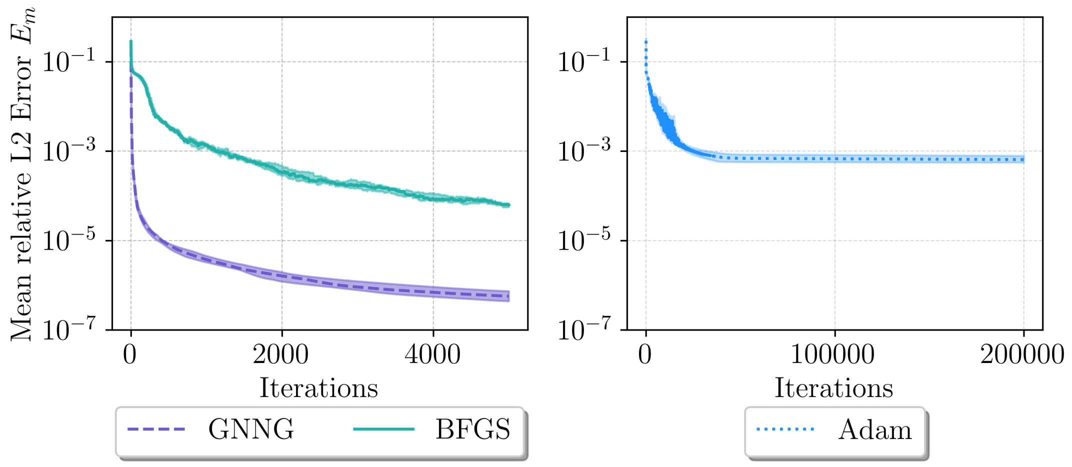
```
1: **Input:** initial parameters $\theta_0 \in \Theta, \psi_0 \in \Psi$, max iterations $N_{max}$
2: **for** $k = 1, \dots, N_{max}$ **do**
3:     Compute gradient $\nabla L(\theta, \psi)$
4:     Assemble Gram matrix $G(\theta, \psi)$
5:     Compute natural grad. $\nabla^G L(\theta, \psi) = G^\dagger(\theta, \psi) \nabla L(\theta, \psi)$
6:     Determine step size $\eta^* = \arg\min_{\eta \in [0,1]} L((\theta, \psi) - \eta \nabla^G L(\theta, \psi))$
7:     Update params $(\theta_k, \psi_k) = (\theta_{k-1}, \psi_{k-1}) - \eta^* \nabla^G L(\theta, \psi)$
8: **end for**
```

GNNG: Results for unsteady 3D Beltrami flow



- Convergence plot for unsteady 3d Beltrami flow, $Re = 1$.
- **Network Architecture:** 4 layers, each with 50 neurons.
- **Initialization:** Weights initialized using Glorot initialization, 10 different initializations.
- **Optimizer:** Adam optimizer with a exponential decaying learning rate.

GNNG: Results for Taylor-Green Vortex: Hard Boundaries



- Convergence plot for unsteady Taylor Green Vortex with hard imposed constraints, $Re = 500$.
- **Network Architecture:** 4 layers, each with 50 neurons.
- **Initialization:** Weights initialized using Glorot initialization, 10 different initializations.
- **Optimizer:** Adam optimizer with exponential decaying learning rate.

GNNG: Visualization of the step direction

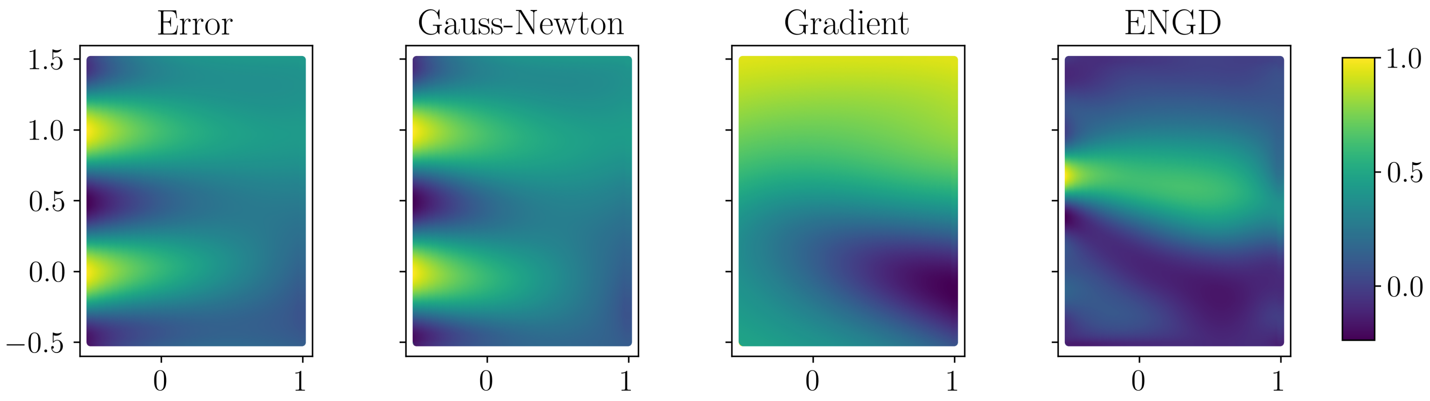


Figure 1: Shown are the error $u_\theta - u^*$ and the push forwards of GNNG, ENGD and vanilla gradient; all functions normed to lie in $[-1, 1]$ to allow for a visual comparison.

GNNG IV: Matrix-Free Formulation

- **Discretized Gramian:** When discretized, the Gramian takes the form

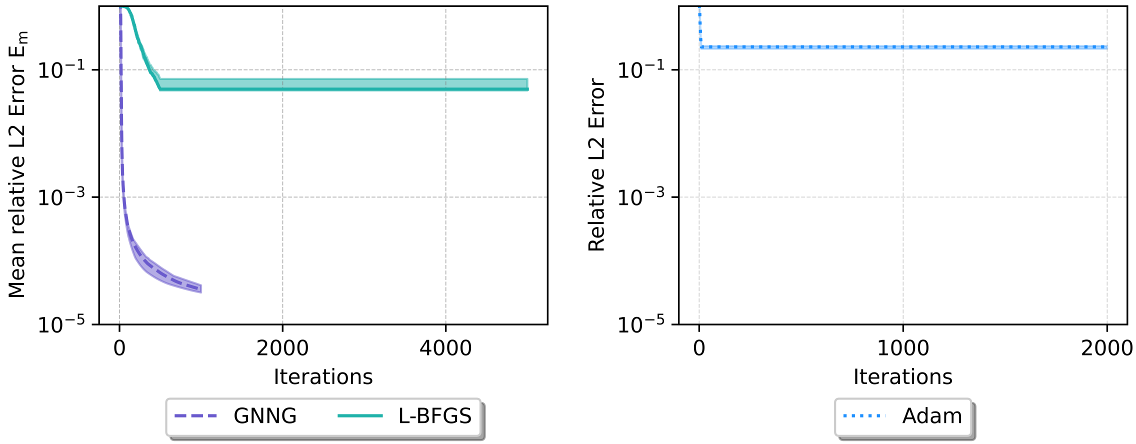
$$G(\theta) \approx \frac{1}{N}\sum_{k=0}^{N} J(\mathcal{L}(u_{\theta_k})) \cdot J(\mathcal{L}(u_{\theta_k}))^T + \frac{\tau}{M}\sum_{k=0}^{M} J(\mathcal{B}(u_{\theta_k})) \cdot J(\mathcal{B}(u_{\theta_k}))^T \tag{8}$$

- **Matrix-Free Gramian Computation:** Employ forward and backward mode automatic differentiation for computing Gramian-vector products.
- **JVP and VJP Combination:**
    - Forward mode yields Jacobian-vector product (JVP): Jv.
    - Backward mode yields vector-Jacobian product (VJP): $J^T w$.
- **Implementation:**

$$G(\theta, \psi)v = J^T w, \text{ with } w \text{ computed as } Jv.$$

- **Matrix-Free Solver:** Use conjugate gradient method to solve $G(\theta, \psi)^{\dagger} \nabla L(\theta, \psi)$, without ever forming G.

Matrix-Free Taylor-Green Vortex: Convergence Visualization



- Convergence plot for unsteady Taylor Green Vortex with soft constraints, $Re = 500$.
- **Network Architecture:** 10 layers, each with 100 neurons.
- **Initialization:** Weights initialized using Glorot initialization, 5 different initializations.
- **Optimizer:** Adam optimizer with exponential decaying learning rate. Benchmark against L-BFGS instead of BFGS.

Conclusions and Future Work

- We proposed a **Gauss-Newton Natural Gradient method in function space** tailored for the Navier-Stokes equation, yielding **highly accurate results**.
- Developed a **matrix-free formulation** enabling the method to **scale effectively for large networks**.
- Future work will focus on:
  - Exploring **matrix-free preconditioning methods** to further **speed up convergence**.
  - Develop parallel computations for improved **scalability**, more specifically for Gramian evaluation.

Thank You

Thank you for listening.
Questions?
email: anas.jnini@unitn.it

References I

[1] Anas Jnini, Flavio Vella, and Marius Zeinhofer. *Gauss-Newton Natural Gradient Descent for Physics-Informed Computational Fluid Dynamics*. 2024. arXiv: `2402.10680 [math.OC]`.

[2] Aditi Krishnapriyan et al. "Characterizing possible failure modes in physics-informed neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 26548–26560.

[3] Johannes Müller and Marius Zeinhofer. "Achieving High Accuracy with PINNs via Energy Natural Gradient Descent". In: *ICML* (2023).

[4] Sifan Wang, Yujun Teng, and Paris Perdikaris. "Understanding and mitigating gradient flow pathologies in physics-informed neural networks". In: *SIAM Journal on Scientific Computing* 43.5 (2021), A3055–A3081.

References II

[5] Yongji Wang and Ching-Yao Lai. "Multi-stage neural networks: Function approximator of machine precision". In: *arXiv preprint arXiv:2307.08934* (2023).

[6] Qi Zeng, Spencer H Bryngelson, and Florian Tobias Schaefer. "Competitive Physics Informed Networks". In: *ICLR 2022 Workshop on Gamification and Multiagent Solutions*. 2022. URL: https://openreview.net/forum?id=rMz_scJ6lc.