Thomas Richter

Hybrid Finite Element / Neural Network Simulations

April 25, 2024, Thomas Richter[1]

Robert Jendersie[1], Uladzislau Kapustsin[1], Utku Kaya[1], Christian Lessig[2], Nils Margenberg[3], Dirk Hartmann[4]

(1) OVGU Magdeburg
(2) ECMWF
(3) HSU Hamburg
(4) Siemens AG

Institut für Analysis und Numerik
Otto-von-Guericke-University Magdeburg

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

DFG-Graduiertenkolleg MATHEMATISCHE KOMPLEXITÄTSREDUKTION

1. Why using neural networks in model-based simulations?
2. Hybrid Simulations in Fluid Dynamics
3. Analysis of hybrid finite element / neural network approximations

The promises of neural networks

Simulation approaches like finite elements, finite differences, dG, finite volumes, etc.

- are really established and well understood
- efficient and good software and solvers are available
- offer excellent conservation properties and usually good accuracy

But, they get to their limits

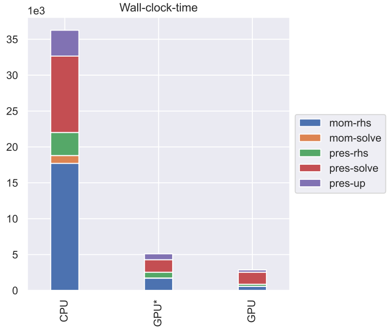
- if scales cannot be resolved
- nonlinearities often require high resolution and yield suboptimal efficiency

Neural networks

- offer good approximation powers
- can discover new and better models
- are little understood

The (unpractible) power of modern hardware

- The compute power of modern accelerators exceeds CPU's
  - AMD Ryzen: 5.4 TFLOPS (FP64) at 350 Watts
  - Nvidia H100: 60 TFLOPS (FP64) 2 000 TFLOPS (FP16) at 350 Watts

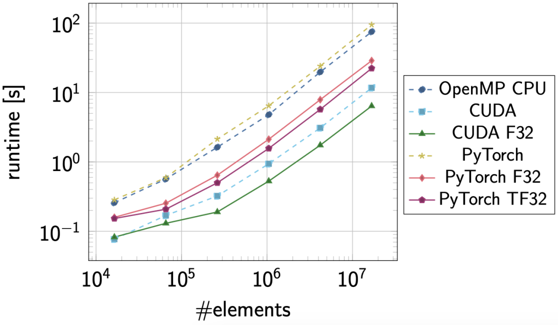


- 3D Navier-Stokes problem
- Explicit pressure correction scheme with Krylow-Multigrid solver for the pressure Poisson problem
- All implemented on the GPU using cuSparse and cuBLAS

▶ M. Liebchen, U. Kaya, C. Lessig, T. Richter. *An adaptive finite element multigrid solver using GPU acceleration*, coming soon

The (unpractible) power of modern hardware

- GPU's are challenging to program
- They have their own memory and little management on the chip, transfer is costly



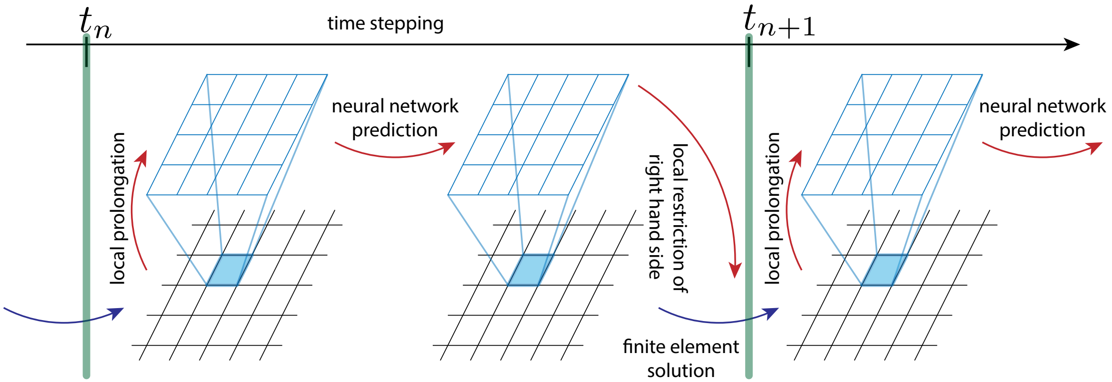
- Dynamic sea-ice model as part of the SASIP project[a]
- Comparing the efficiency of programming models including machine learning frameworks

[a] The Scale Aware Sea-Ice Project
`https://sasip-climate.github.io`

- R. Jendersie, C. Lessig, and T. Richter. *Towards a GPU-Parallelization of the neXtSIM-DG Dynamical Core*, PASC' 24: Proceedings of the Platform for Advanced Scientific Computing Conference, 2024

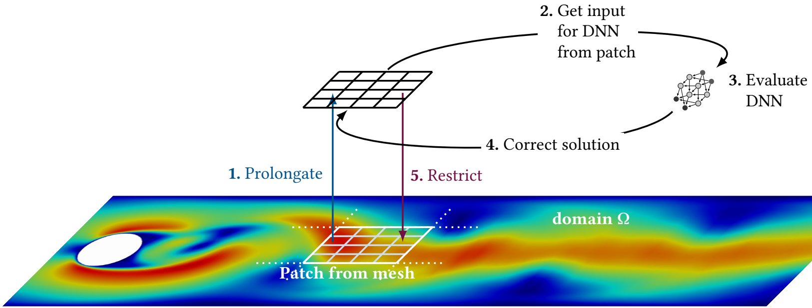
1. Why using neural networks in model-based simulations?
2. Hybrid Simulations in Fluid Dynamics
3. Analysis of hybrid finite element / neural network approximations

Idea of the Deep Neural Network Multigrid Solver



- Combine classical simulation on coarse mesh Ω_H with neural network on fine mesh Ω_h
- Embedded in time-stepping

It is a local approach - no PINN

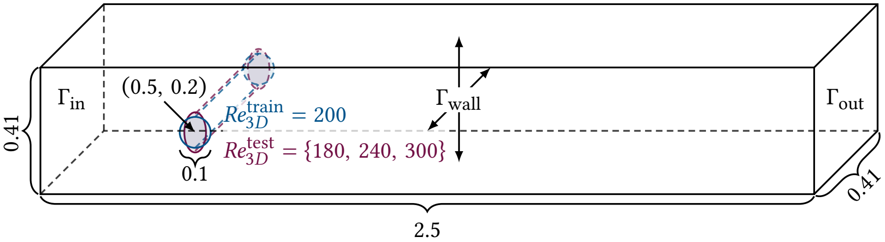


(Multigrid) Finite elements on **coarse meshes**, local neural network update on **fine meshes**

- (Small) local network with input dimension $N_{in} < 500$ in 3D

- Nils Margenberg, Dirk Hartmann, Christian Lessig, Thomas Richter. *A neural network multigrid solver for the Navier-Stokes equations*, Journal of Computational Physics, 2022

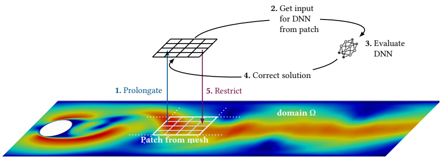
Training of the neural network - Benchmark problem



- Flow around an obstacle at $Re = 200$

	# DoF	Patch size
Coarse	390 720	0
Reference for DNN-MG(3+1)	3 003 520	8
Reference for DNN-MG(3+1)	23 546 112	64

Generation of training data



- A single problem setup on a protoypical time interval
- Simultaneously running two simulations on coarse and fine mesh. Each time-step starting restriction of fine solution

$$
\begin{aligned}
u_{h_F}^{n+1} + \Delta t \mathcal{A}_{h_F}(u_{h_F}^{n+1}) &= u_{h_F}^{n} - \Delta t \mathcal{B}_{h_F}(u_{h_F}^{n}) \\
u_{h_C}^{n+1} + \Delta t \mathcal{A}_{h_C}(u_{h_C}^{n+1}) &= I_{h_C} u_{h_F}^{n} - \Delta t \mathcal{B}_{h_C}(I_{h_C} u_{h_F}^{n})
\end{aligned}
$$

- Data is generated by splitting solution into patches

$$
\left(u_{h_C}^{n}\Big|_{\mathcal{P}_i}, u_{h_C}^{n+1}\Big|_{\mathcal{P}_i}, \mathcal{P}_i\right) \mapsto u_{h_F}^{n+1}\Big|_{\mathcal{P}_i}, \quad n = 1, \dots, N, \quad i = 1, \dots, N_{\mathcal{P}}
$$

Training data and neural networks

- One single simulation $Re = 200$. Training data $I = [4, 7]$ with $N_T = 375$ time steps
- Validation set $I = [2, 4]$ with $N_T^v = 250$ time steps
- Approximately $N_P = 50\,000$ patches (coarse mesh elements) for each time-step
- Total dataset has 1 TB data

Loss function

- Data loss plus model information ($\mathcal{R}$ is the equation's residual)

$$loss = \sum_{i,n} \alpha \left\| u_{h_F}^{n+1} - \mathcal{N}\left(u_{h_C}^{n}\big|_{\mathcal{P}_i}, u_{h_C}^{n+1}\big|_{\mathcal{P}_i}, \mathcal{P}_i \right) \right\|^2 + \beta \left\| \mathcal{R}\left(\mathcal{N}\left(u_{h_C}^{n}, u_{h_C}^{n+1}, \mathcal{P} \right) \right) \right\|^2 + \gamma\, loss_{reg}$$

Neural network training

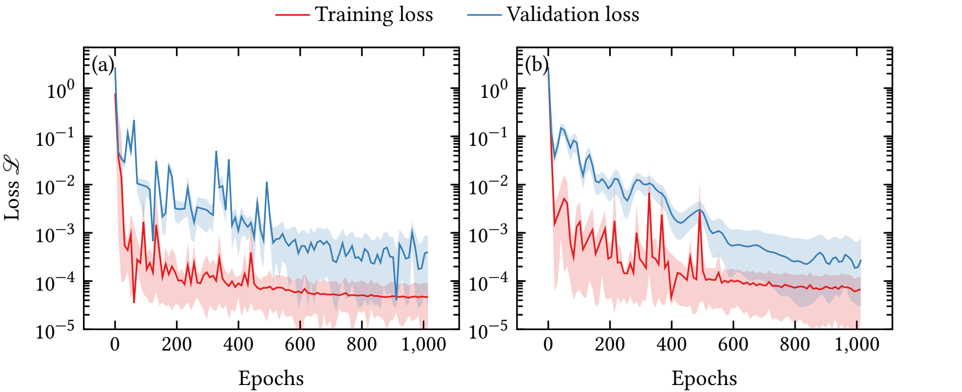
Neural networks

- Different architectures: standard MLP, Convolutional layers, Transformers,
- Memory elements like GRU's (work and help but make training costly)
- Network ensembles for error control

Computer infrastructure

- 2 GPU nodes with 2 Nvidia A100 GPU's each
- Average GPU load between 80% (small network) and 90% (large network)

Training

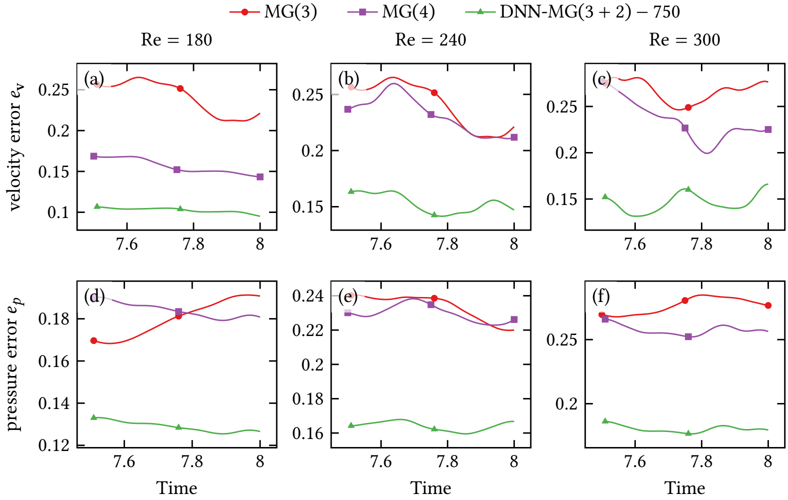


Size of hidden layer

a) Neural network with width 512 and 2 500 000 parameters

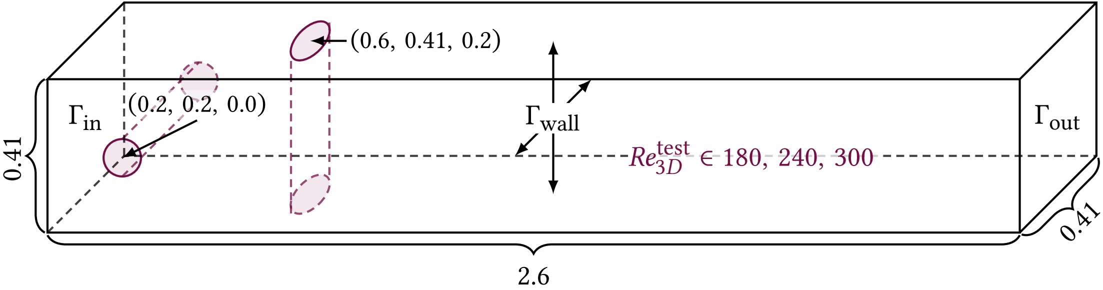
b) Neural network with width 750 and 5 000 000 parameters

Testing the neural network for different Reynolds numbers



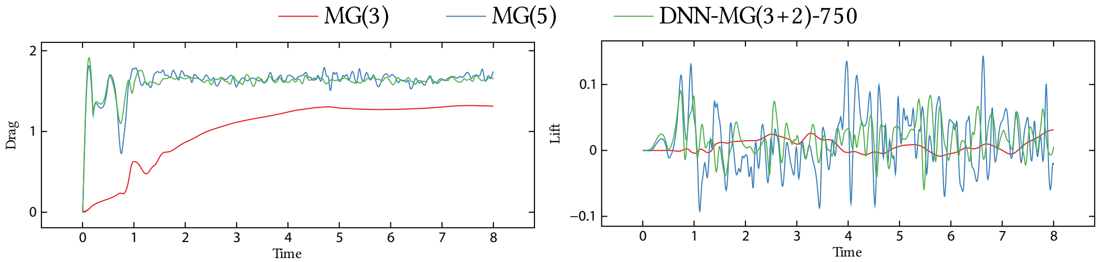
Modified benchmark problem: Ellipse instead of circle, higher Reynolds number

Generalization - Two obstacle problem



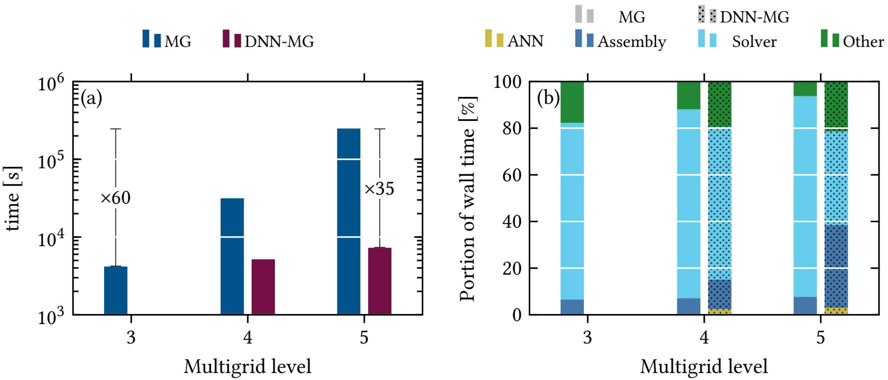
- Measure drag and lift on second obstacle

2 Obstacles, $Re = 240$



Type	min C_d	max C_d	$\overline{C_d}$	amp C_d
Coarse FE	0.767	1.321	1.218	0.554
Mid. FE	0.791	1.560	0.770	1.375
Reference FE	**1.509**	**1.791**	**1.664**	**0.282**
DNN-(3 + 1)	**1.453**	**1.653**	**1.555**	**0.204**
DNN-(3 + 2)	**1.464**	**1.682**	**1.593**	**0.218**

Computational effort



- N. Margenberg, R. Jendersie, C. Lessig, and T. Richter. *DNN-MG: A hybrid neural network/finite element method with applications to 3d simulations of the Navier-Stokes equations*, CMAME, 2023, `https://arxiv.org/abs/2307.14837`

Simple model problem

Laplace equation

$$-\Delta u = f \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega$$

Finite Element approximation

$$u_h \in V_h \subset H_0^1(\Omega) \quad (\nabla u_h, \nabla \phi_h) = (f, \phi_h) \quad \forall \phi_h \in V_h$$

Hybrid Finite Element / Neural network approach

$$\underbrace{u_{h_C} \in V_{h_C} \quad (\nabla u_{h_C}, \nabla \phi_{h_C}) = (f, \phi_{h_C}) \quad \phi_{h_C} \in V_{h_C}}_{\text{coarse finite element solution}} \quad \to \quad \underbrace{u_{h_F} = u_{h_C} + \mathcal{N}(u_{h_C}, f)}_{\text{neural network correction}}$$

Available Tools

Finite Elements

- Fast solvers give good accuracy

$$\left|(f, \phi_h) - (\nabla u_h, \nabla \phi_h)\right| \leq \epsilon$$

- Galerkin orthogonality / Cea's Lemma

$$\|\nabla(u - u_h)\| \lesssim \|\nabla(u - v_h)\| \quad \forall v_h \in V_h$$

- Optimal interpolation results

$$\|\nabla(u - u_h)\| \leq ch^r \|f\|_{r-1}$$

Neural Network approaches

- Optimization problems

$$loss(\mathcal{N}) \to 0$$

- Approximation in Sobolev spaces [1]

$$\inf_{\mathcal{N}} \|\mathcal{N} - f\|_{W^{s,p}} \sim \frac{1}{N_{\mathcal{N}}^{\gamma(dim,s,p)}}$$

- PINN's: best approximation-like [2]

$$\|u - u_{\mathcal{N}}\| \lesssim \sqrt{loss(\mathcal{N}) - \inf_{\tilde{\mathcal{N}}} loss(\tilde{\mathcal{N}})} + \inf_{\hat{\mathcal{N}}} \|u - \hat{\mathcal{N}}\|$$

- [1] Gühring, Kutyniok, Petersen. *Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$-norms*, Analysis and Applications 2019
- [2] Müller, Zeinhofer. *Error estimates for the variational training of neural networks with boundary penalty*, 2021

Generalization error

Describes the error resulting from the application to new data

- Loss-function for hybrid Laplace solution

$$loss = \sum_{i,p} \left\| u^i_{h_F}\big|_{\mathcal{P}} - \left(u^i_{h_C}\big|_{\mathcal{P}} + \mathcal{N}(u^i_{h_C}\big|_{\mathcal{P}}, f_i\big|_{\mathcal{P}}) \right) \right\|^2$$

- How is the network performing on data that is not part of the training?

$$\left| \mathcal{N}(u^i_{h_C}\big|_{\mathcal{P}}, f_i\big|_{\mathcal{P}}) - \mathcal{N}(\overline{u_{h_C}}\big|_{\mathcal{P}}, \bar{f}\big|_{\mathcal{P}}) \right| \lesssim \| u^i_{h_C}\big|_{\mathcal{P}} - \overline{u_{h_C}} \| + \| f_i\big|_{\mathcal{P}} - \bar{f} \|$$

Data error

How dense is the training data?

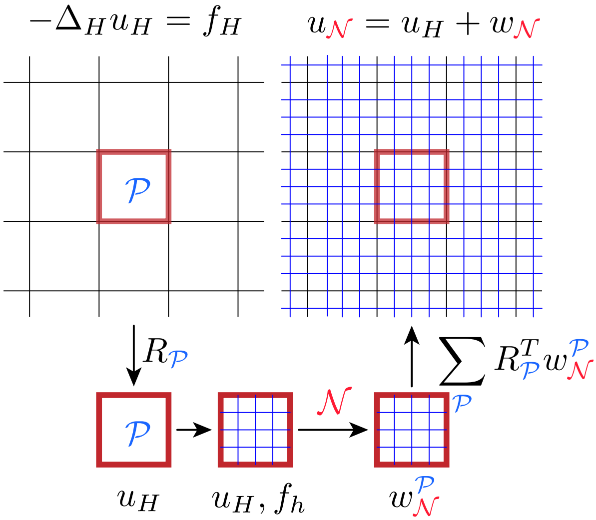
$$\min_{i,\mathcal{P}} \left\| f_i\big|_{\mathcal{P}} - \overline{f} \right\| < \epsilon_{\text{data}}$$

Optimization error

How good is the minimium?

$$\mathit{loss}(\mathcal{N}) - \inf_{\tilde{\mathcal{N}}} \mathit{loss}(\tilde{\mathcal{N}}) < \epsilon_{\text{opt}}$$

Hybrid finite element neural network solver - Laplace



1. Solve coarse finite element problem

$$(\nabla u_{h_C}, \nabla \phi_{h_C}) = (f, \phi_{h_C})$$

2. Restrict to local patch

$$u_{h_C} \to \{u_{h_C}|_{\mathcal{P}_i},\ i = 1, \dots, N_{\mathcal{P}}\}$$

3. Local neural network update

$$w|_{\mathcal{P}_i} = \mathcal{N}(u_{h_C}|_{\mathcal{P}_i}, f|_{\mathcal{P}_i})$$

4. Update global solution

$$u_{h_F}^{\mathcal{N}} = u_{h_C} + \sum \mathcal{I}_{\mathcal{P}_i} w|_{\mathcal{P}_i}$$

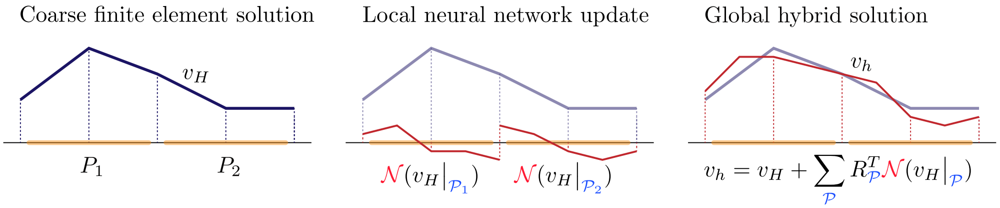
Training data from set of possible problems $\mathcal{F}_d = \{f_1, \dots, f_N\} \subset \mathcal{F}$

$$(\nabla u^i_{h_F}, \nabla \phi_{h_F}) = (f_i, \phi_{h_F}) \quad \forall \phi_{h_F} \in V_{h_F} \quad \Big| \quad (\nabla u^i_{h_C}, \nabla \phi_{h_C}) = (f_i, \phi_{h_C}) \quad \forall \phi_{h_C} \in V_{h_C}$$

Minimize

$$loss = \sum_{i,\mathcal{P}} \left\| u^i_{h_F}\big|_{\mathcal{P}} - \left(u^i_{h_C}\big|_{\mathcal{P}} + \mathcal{N}\left(u^i_{h_C}\big|_{\mathcal{P}}, f^i\big|_{\mathcal{P}}\right)\right)\right\|^2$$

Local network updates



Theorem (The simple one-patch case)

Let the network be trained such that

$$\sum_{f_i \in \mathcal{F}_{tr}} \|u_{h_F}^{f_i} - v_{\mathcal{N}}^{f_i}\|^2 \le \epsilon_{\mathcal{N}}^2$$

For any f it holds

$$\left\|\nabla\left(u - \left(u_{h_C} + \mathcal{N}(u_{h_C}, f)\right)\right)\right\| \le c_{\mathcal{N}}\left({h_F}^r + \min_{f_i \in \mathcal{F}_d} \|f - f_i\|_{H^{-1}}\right) + \epsilon_{\mathcal{N}}$$

The constant $c_{\mathcal{N}} > 0$ is (partially) computed a posteriori based on the trained network $\mathcal{N}$.

- Kapustsin, Kaya, Richter. *A hybrid finite element/neural network solver and its application to the poisson problem*, PAMM 2023

1) Quality of training data

We insert the fine solution $u^f_{h_F} \in V_{h_F}$ for f

$$(\nabla u^f_{h_F}, \nabla \phi_{h_F}) = (f, \phi_{h_F}) \quad \forall \phi_{h_F} \in V_{h_F}$$

and estimate with *standard finite element estimates*

$$\|\nabla(u - u_{\mathcal{N}})\| \le \|\nabla(u - u^f_{h_F})\| + \|\nabla(u^f_{h_F} - u_{\mathcal{N}})\|$$

$$\le \underline{c{h_F}^r \|f\|_{H^{r-1}(\Omega)}} + \boxed{\|\nabla(u^f_{h_F} - u_{\mathcal{N}})\|}$$

2) Density of Training Data

We insert the resolved solution $u^i_{h_F}$ for $f_i \in \mathcal{F}_d$ closest to $f \in \mathcal{F}$

$$(\nabla u_{h_F}, \nabla \phi_{h_F}) = (f, \phi_{h_F})$$
$$(\nabla u^i_{h_F}, \nabla \phi_{h_F}) = (f_i, \phi_{h_F})$$

and estimate with *standard elliptic stability estimates*

$$\|\nabla(u_{h_F} - u_{\mathcal{N}})\| \leq \|\nabla(u_{h_F} - u^i_{h_F})\| + \|\nabla(u^i_{h_F} - u_{\mathcal{N}})\|$$

$$\leq \underline{\|f - f_i\|_{H^{-1}(\Omega)}} + \boxed{\|\nabla(u^i_{h_F} - u_{\mathcal{N}})\|}$$

3) Network Approximation and Optimization Error

We insert the neural network solution for the training point v_h^i

$$u_{\mathcal{N}} = u_{h_C} + \mathcal{N}(u_{h_C}, f)$$
$$u_{\mathcal{N}}^i = u_{h_C}^i + \mathcal{N}(u_{h_C}^i, f_i)$$

and estimate

$$\|\nabla(u_{h_F}^i - u_{\mathcal{N}})\| \le \|\nabla(u_{h_F}^i - u_{\mathcal{N}}^i)\| + \|\nabla(u_{\mathcal{N}}{}^i - v_{\mathcal{N}}\|$$

$$\le \epsilon_{\mathcal{N}} + \boxed{\|\nabla(u_{\mathcal{N}}^i - u_{\mathcal{N}})\|}$$

4) Generalizability = Network Stability

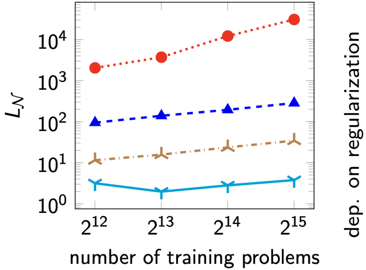
$$\|\nabla(u_{\mathcal{N}}{}^i - u_{\mathcal{N}})\| = \Big\|\nabla\Big((u_{h_C} + \mathcal{N}(u_{h_C}, f)) - (u^i_{h_C} + \mathcal{N}(u^i_{h_C}, f_i))\Big)\Big\|$$

$$\leq \underbrace{\|\nabla(u_{h_C} - u^i_{h_C})\|}_{\leq \|f - f_i\|_{H^{-1}(\Omega)}} + \boxed{\|\nabla(\mathcal{N}(u_{h_C}, f) - \mathcal{N}(u^i_{h_C}, f_i))\|}$$

A posteriori estimation of the Networks Lipschitz constant

$$\|\nabla(\mathcal{N}(u_{h_C}, f) - \mathcal{N}(u^i_{h_C}, f_i))\| \\ \leq L_{\mathcal{N}}(\|\nabla(u_{h_C} - u^i_{h_C})\| + \|f - f_i\|)$$

$$\boxed{\|\nabla(v - v_{\mathcal{N}})\| \leq c_{\mathcal{N}}\Big(h_F{}^r + \min_{f_i \in \mathcal{F}_d} \|f - f_i\|_{H^{-1}}\Big) + \epsilon_{\mathcal{N}}}$$



The local approach - Patches

Instead of inserting one global fine solution

$$\|\nabla(u-u_{\mathcal{N}})\| \le \|\nabla(u-u^f_{h_F})\| + \|\nabla(u^f_{h_F}-u_{\mathcal{N}})\|$$

we can/must now insert separate local fine-scale solution on each patch

$$u^{f,\mathcal{P}}_{h_F} := u^f_{h_C} + w^{f,\mathcal{P}}_{h_F}: \qquad (\nabla(u^f_{h_C} + w^{f,\mathcal{P}}_{h_F}), \nabla\phi^{\mathcal{P}}_{h_F})_{\mathcal{P}} = (f, \phi^{\mathcal{P}}_{h_F})_{\mathcal{P}} \quad \forall \phi^{\mathcal{P}}_{h_F} \in V_{h_F}(\mathcal{P})$$

Interior Error Esitmates

Use Local Finite Element Estimates for $\mathcal{P} \subset\subset \tilde{\mathcal{P}} \subset\subset \Omega$

$$\|\nabla(u - u_{h_F,\mathcal{P}})\| \lesssim \inf_{v_{h_F} \in V_{h_F}(\tilde{\mathcal{P}})} \|\nabla(u - v_{h_F})\|_{\tilde{\mathcal{P}}} + d(\mathcal{P}) \|u - u_{h_F,\mathcal{P}}\|_{H^{1-r}(\tilde{\mathcal{P}})}$$

- Nitsche, Schatz. *Interior estimates for Ritz-Galerkin methods*, Mathematics of Computation, 1974
  - But, standard estimates give

$$d(\mathcal{P}) \approx \frac{1}{\operatorname{dist}(\mathcal{P}, \tilde{\mathcal{P}})^r}$$

  - All is fine, if patches do not depend on mesh size (fixed number of patches)

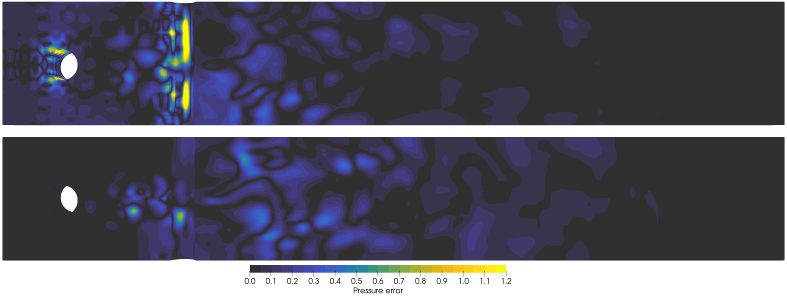
$$\|\nabla(v - v_{\mathcal{N}})\| \le c_{\mathcal{N}} \Big(\boxed{{h_C}^{2r}} + {h_F}^r + \min_{f_i \in \mathcal{F}_d} \|f - f_i\|_{H^{-1}} \Big) + \epsilon_{\mathcal{N}}$$

  - But, numerical evidence shows optimal results for

$$diam(\mathcal{P}) = O({h_C}^d)$$

Summary

- Hybridization of finite elements and deep neural networks boosts efficiency
- (Robust) analysis is possible for simple equations
- Sharp estimates for the generalization (Lipschitz) constant $c(\mathcal{N})$ still needed
- Still fighting with the local case



https://numerics.ovgu.de

Literature

- N. Margenberg, R. Jendersie, C. Lessig, and T. Richter. *Dnn-mg: A hybrid neural network/finite element method with applications to 3d simulations of the Navier-Stokes equations*, CMAME, 2023
- P. Minakowski and T. Richter. *A priori and a posteriori error estimates for the deep ritz method applied to the Laplace and Stokes problem*, Journal of Computational and Applied Mathematics, 2023
- U. Kapustsin, U. Kaya, and T. Richter. *A hybrid finite element/neural network solver and its application to the poisson problem*, PAMM 2023
- N. Margenberg, D. Hartmann, C. Lessig, and T. Richter. *A neural network multigrid solver for the Navier-Stokes equations*, Journal of Computational Physics 2022
- N. Margenberg, C. Lessig, and T. Richter. *Structure preservation for the deep neural network multigrid solver*, ETNA 2021
- R. Becker, M. Braack, D. Meidner, T. Richter, B. Vexler. *Gascoigne 3D - Adaptive Multigrid Finite Element Library*, `www.gascoigne.de`