



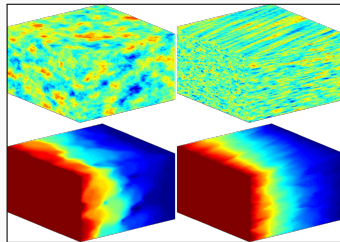
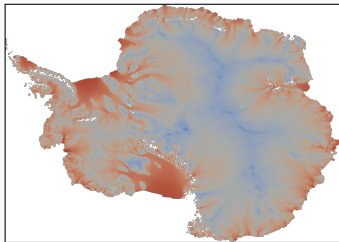
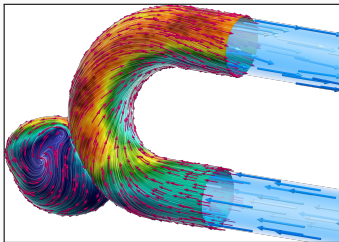
Localization and Efficient Training of Scientific Machine Learning Models

Alexander Heinlein¹

4th International Conference on Recent Advances in Fluid Mechanics and Nanoscience,
NIT Jalandhar, India, September 29–October 1, 2026

¹Delft University of Technology

Scientific Computing and Machine Learning



Numerical methods

Based on physical models

- + Robust and generalizable
- Require availability of mathematical models

Machine learning models

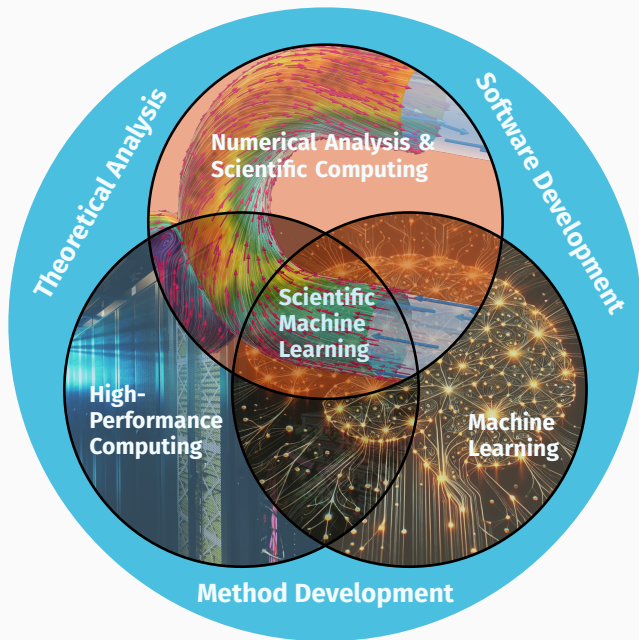
Driven by data

- + Do not require mathematical models
- Sensitive to data, limited extrapolation capabilities

Scientific machine learning

Combining the strengths and compensating the weaknesses of the individual approaches:

numerical methods	improve	machine learning techniques
machine learning techniques	assist	numerical methods



1 Physics-informed neural networks

2 Domain decomposition-based neural network architectures

Based on joint work with

Victorita Dolean

(Eindhoven University of Technology)

Siddhartha Mishra

(ETH Zürich)

Ben Moseley

(Imperial College London)

3 Enhanced training and extension to deep operator networks

Based on joint work with

Damien Beecroft

(University of Washington)

Amanda A. Howard and **Panos Stinis**

(Pacific Northwest National Laboratory)

Taniya Kapoor

(Wageningen University & Research)

Physics-informed neural networks

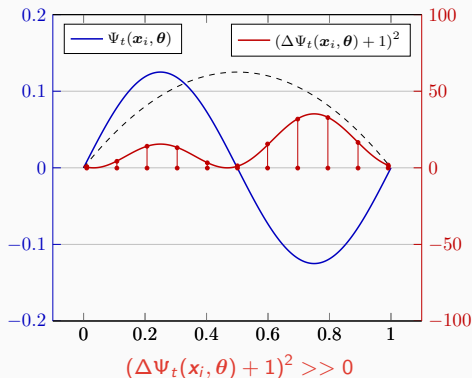
Physics-Informed Neural Networks (PINNs) — Idea

In **Lagaris et al. (1998)**, the authors solve the **boundary value problem**

$$\begin{aligned} -\Delta \Psi_t(\mathbf{x}, \theta) &= 1 \text{ on } [0, 1], \\ \Psi_t(0, \theta) &= \Psi_t(1, \theta) = 0, \end{aligned}$$

via a **collocation approach**:

$$\min_{\theta} \sum_{x_i} (\Delta \Psi_t(x_i, \theta) + 1)^2$$

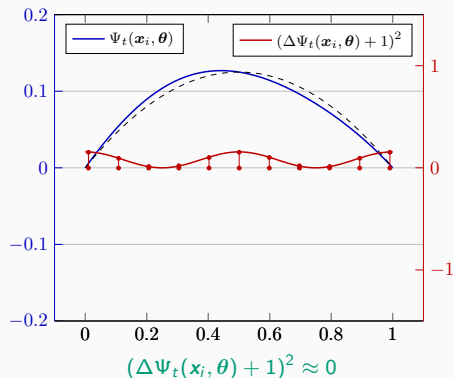


Boundary conditions ...

... can be **enforced explicitly** via the ansatz:

$$\Psi_t(\mathbf{x}, \theta) = A(\mathbf{x}) + F(\mathbf{x}, \text{NN}(\mathbf{x}, \theta))$$

- A satisfies the boundary conditions
- F does not contribute to the boundary conditions



Physics-Informed Neural Networks (PINNs)

In the **physics-informed neural network (PINN)** approach introduced by **Raissi et al. (2019)**, a **neural network** is employed to **discretize a partial differential equation**

$$\mathcal{N}[u] = f, \quad \text{in } \Omega.$$

PINNs use a **hybrid loss function**:

$$\mathcal{L}(\theta) = \omega_{\text{data}} \mathcal{L}_{\text{data}}(\theta) + \omega_{\text{PDE}} \mathcal{L}_{\text{PDE}}(\theta),$$

where ω_{data} and ω_{PDE} are **weights** and

$$\mathcal{L}_{\text{data}}(\theta) = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} (u(\hat{\mathbf{x}}_i, \theta) - u_i)^2,$$

$$\mathcal{L}_{\text{PDE}}(\theta) = \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} (\mathcal{N}[u](\mathbf{x}_i, \theta) - f(\mathbf{x}_i))^2.$$

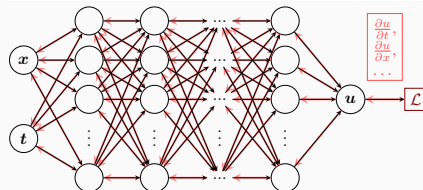
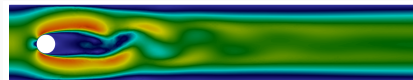
See also **Dissanayake and Phan-Thien (1994)**; **Lagaris et al. (1998)**.

Advantages

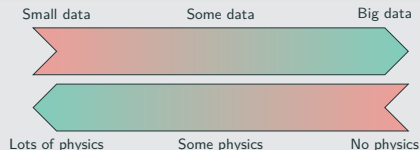
- **"Meshfree"**
- **Small data**
- **Generalization properties**
- **High-dimensional problems**
- **Inverse and parameterized problems**

Drawbacks

- **Training cost** and **robustness**
- **Convergence not well-understood**
- **Difficulties with scalability** and **multi-scale problems**



Hybrid loss



- **Known solution values** can be included in $\mathcal{L}_{\text{data}}$
- **Initial and boundary conditions** are also included in $\mathcal{L}_{\text{data}}$

Error Estimate & Spectral Bias

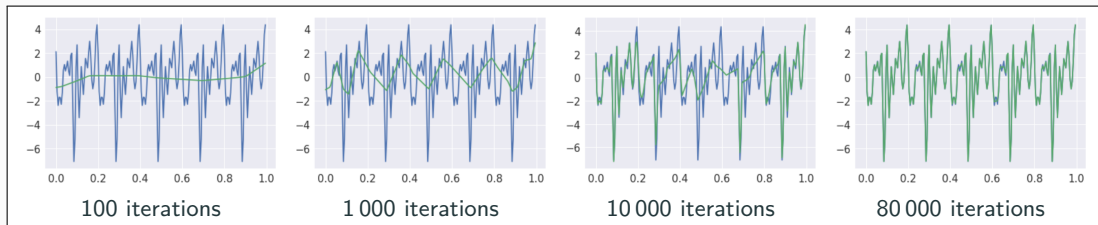
Estimate of the generalization error (Mishra and Molinaro (2022))

The generalization error (or total error) satisfies

$$\mathcal{E}_G \leq C_{\text{PDE}} \mathcal{E}_T + C_{\text{PDE}} C_{\text{quad}}^{1/p} N^{-\alpha/p}$$

- $\mathcal{E}_G = \mathcal{E}_G(\mathbf{X}, \theta) := \|\mathbf{u} - \mathbf{u}^*\|_V$ **general. error** (V Sobolev space, $\mathbf{X}$ training data set)
- $\mathcal{E}_T$ **training error** (L^p loss of the residual of the PDE)
- N **number of the training points** and α **convergence rate of the quadrature**
- C_{PDE} and C_{quad} **constants** depending on the **PDE**, **quadrature**, and **neural network**

*Rule of thumb: “As long as the PINN is **trained well**, it also **generalizes well**”*

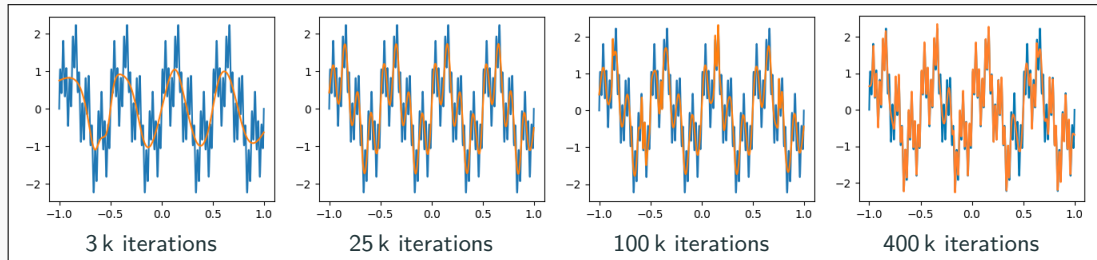


Rahaman et al., On the spectral bias of neural networks, ICML (2019)

Related works: Cao et al. (2021), Wang, et al. (2022), Hong et al. (arXiv 2022), Xu et al. (2024), ...

Spectral Bias of Neural Networks

Rahaman et al. (2019) observed the **spectral bias of neural networks**: during training, they **learn low-frequency functions faster than high-frequency functions**; see also Cao et al. (2021), Wang et al. (2022), Hong et al. (arXiv 2022), Xu et al. (2024), ...



For a mean-squared loss $\mathcal{C}(\theta) = \frac{1}{2N} \|\mathbf{r}(\theta)\|^2$ with error/PDE residual $\mathbf{r}(\theta)$ and Jacobian $\mathbf{J} = \frac{\partial \mathbf{r}(\theta)}{\partial \theta}$, gradient descent yields

$$\theta_{k+1} = \theta_k - \eta \nabla_{\theta} \mathcal{C}(\theta_k) = \theta_k - \eta/N \mathbf{J}_k^{\top} \mathbf{r}(\theta_k)$$

Linearization gives

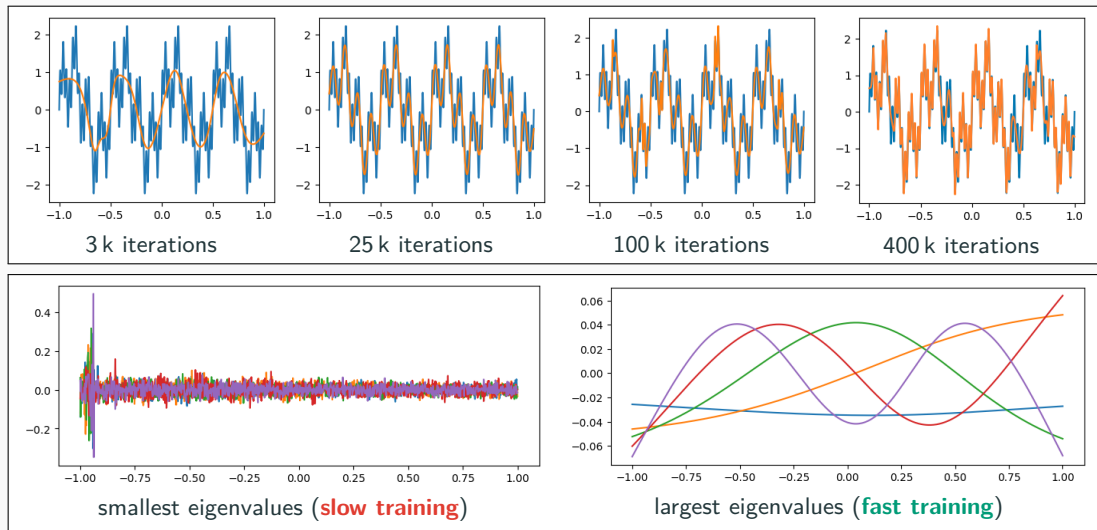
$$\mathbf{r}(\theta_{k+1}) \approx \mathbf{r}(\theta_k) + \mathbf{J}_k(\theta_{k+1} - \theta_k) = \mathbf{r}(\theta_k) - \eta/N \mathbf{J}_k \mathbf{J}_k^{\top} \mathbf{r}(\theta_k).$$

Here, $\Theta_k := 1/N \mathbf{J}_k \mathbf{J}_k^{\top}$ is the (finite-width) **neural tangent kernel (NTK)**. For **infinitely wide neural networks** or **infinitesimal step sizes**, we obtain a **constant NTK**, which yields the residual propagation:

$$\mathbf{r}_k \approx (\mathbf{I} - \eta \Theta)^k \mathbf{r}_0 = \mathbf{V}^{\top} (\mathbf{I} - \eta \Lambda)^k \mathbf{V} \mathbf{r}_0.$$

Spectral Bias of Neural Networks

Rahaman et al. (2019) observed the **spectral bias of neural networks**: during training, they **learn low-frequency functions faster than high-frequency functions**; see also Cao et al. (2021), Wang et al. (2022), Hong et al. (arXiv 2022), Xu et al. (2024), ...



Domain decomposition-based neural network architectures

Scaling of PINNs for a Simple ODE Problem

Solve

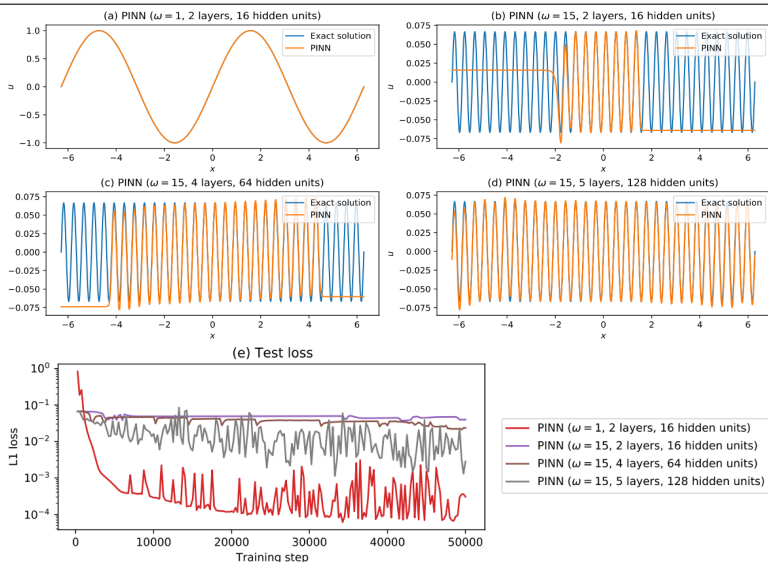
$$\begin{aligned}u' &= \cos(\omega x), \\ u(0) &= 0,\end{aligned}$$

for different values of ω
using **PINNs** with
varying network
capacities.

Scaling issues

- Large computational domains
- Small frequencies

Cf. Moseley, Markham, and
Nissen-Meyer (2023)



(a) 321 free parameters

(d) 66 433 free parameters

Scaling of PINNs for a Simple ODE Problem

Solve

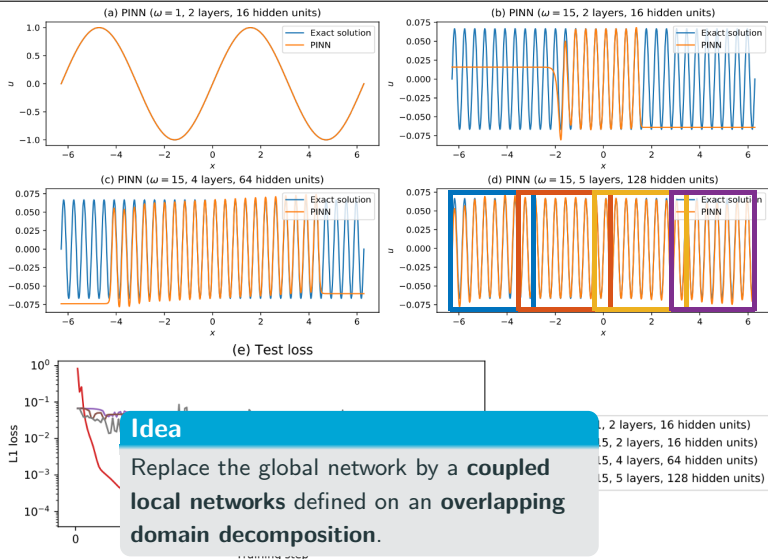
$$\begin{aligned}u' &= \cos(\omega x), \\ u(0) &= 0,\end{aligned}$$

for different values of ω
using **PINNs** with
varying network
capacities.

Scaling issues

- Large computational domains
- Small frequencies

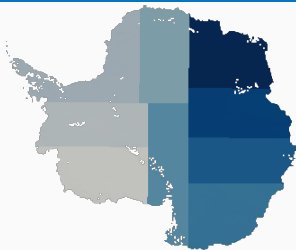
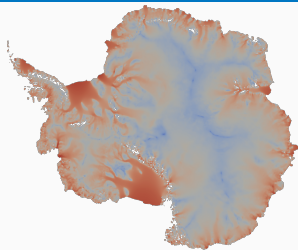
Cf. Moseley, Markham, and
Nissen-Meyer (2023)



(a) 321 free parameters

(d) 66 433 free parameters

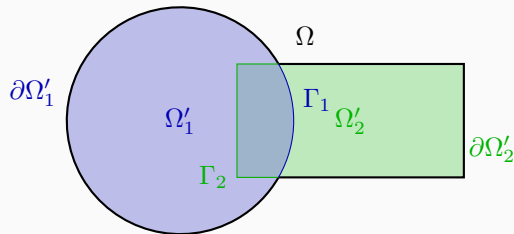
Domain Decomposition Methods



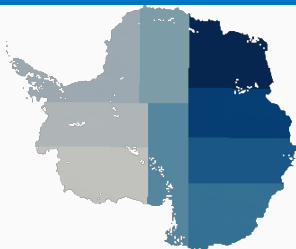
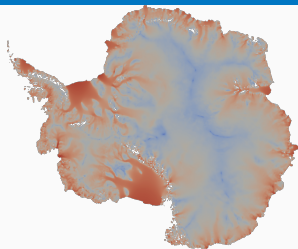
Graphics based on results from [Heinlein, Perego, Rajamanickam \(2022\)](#)

Historical remarks: The **alternating Schwarz method** is the earliest **domain decomposition method (DDM)**, which has been invented by **H. A. Schwarz** and published in **1870**:

- Schwarz used the algorithm to establish the **existence of harmonic functions** with prescribed boundary values on **regions with non-smooth boundaries**.



Domain Decomposition Methods



Graphics based on results from H.

Historical remarks: The

Schwarz method is the
decomposition method

been invented by **H. A.**

published in **1870**:

- Schwarz used the algorithm to establish the **existence of harmonic functions** with prescribed boundary values on **regions with non-smooth boundaries**.

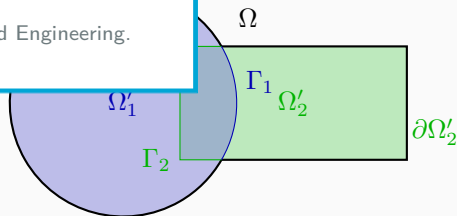
An overview of the state-of-the-art on domain decomposition and machine learning in 2024:



A. Klawonn, M. Lanser, J. Weber

Machine learning, domain decomposition methods – a survey

Computational Science and Engineering.
2024



Finite Basis Physics-Informed Neural Networks (FBPINNs)

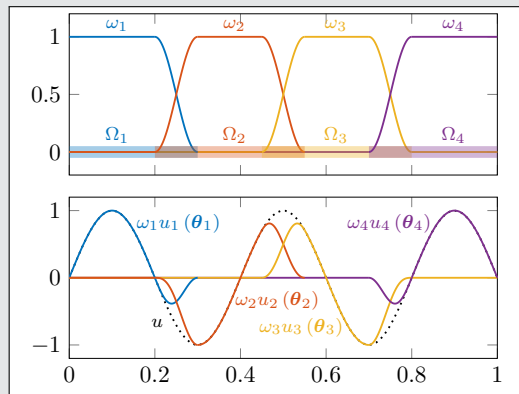
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

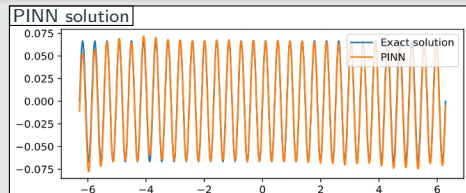
$$u(\theta_1, \dots, \theta_J) = \sum_{j=1}^J \omega_j u_j(\theta_j)$$

and the **loss function**

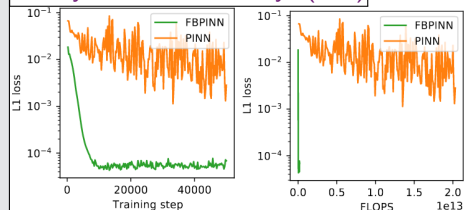
$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left(n \left[\sum_{\mathbf{x}_i \in \Omega_j} \omega_j u_j \right] (\mathbf{x}_i, \theta_j) - f(\mathbf{x}_i) \right)^2.$$



1D single-frequency problem



Moseley, Markham, Nissen-Meyer (2023)



Finite Basis Physics-Informed Neural Networks (FBPINNs)

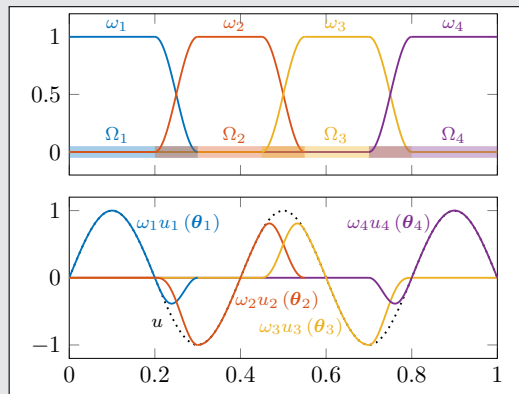
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

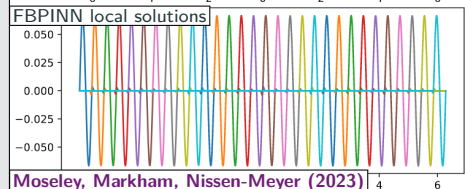
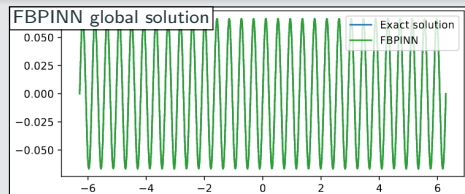
$$u(\theta_1, \dots, \theta_J) = \sum_{j=1}^J \omega_j u_j(\theta_j)$$

and the **loss function**

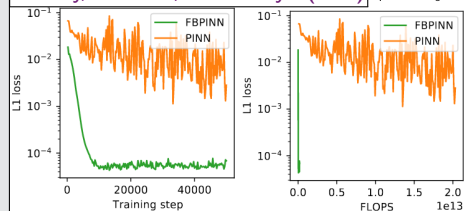
$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left(n \left[\sum_{\mathbf{x}_i \in \Omega_j} \omega_j u_j \right] (\mathbf{x}_i, \theta_j) - f(\mathbf{x}_i) \right)^2.$$



1D single-frequency problem

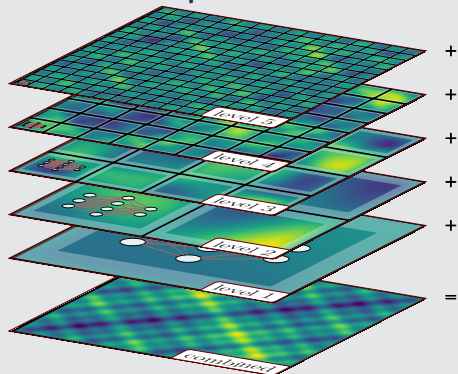


Moseley, Markham, Nissen-Meyer (2023)



Multi-level FBPINNs (ML-FBPINNs)

ML-FBPINNs (Dolean, Heinlein, Mishra, Moseley (2024)) are based on a **hierarchy of domain decompositions**:

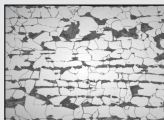


This yields the **network architecture**:

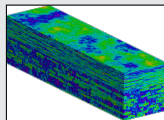
$$u(\theta_1^{(1)}, \dots, \theta_{j^{(L)}}^{(L)}) = \sum_{l=1}^L \sum_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}(\theta_j^{(l)})$$

Multiscale problems ...

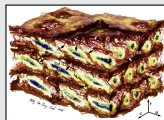
... appear in most areas of modern science and engineering:



Dual-phase steel;
fig. courtesy of
J. Schröder.



Groundwater flow;
cf. **Christie & Blunt**
(2001) (SPE10).



Arterial walls;
cf. **O'Connell et al.**
(2008).

Multi-frequency problem

Consider the **multi-frequency Laplace problem**

$$-\Delta u = 2 \sum_{i=1}^n (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y),$$

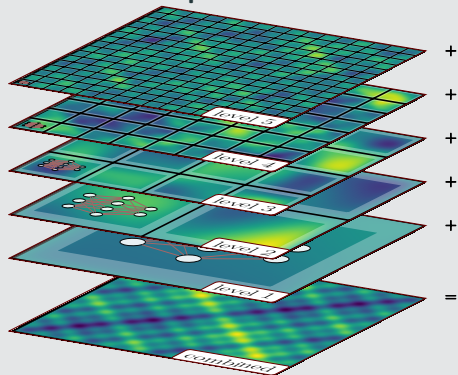
with homogeneous Dirichlet boundary conditions and $\omega_i = 2^i$.

For increasing values of n , we obtain the solutions:



Multi-level FBPINNs (ML-FBPINNs)

ML-FBPINNs (Dolean, Heinlein, Mishra, Moseley (2024)) are based on a hierarchy of domain decompositions:

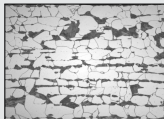


This yields the **network architecture**:

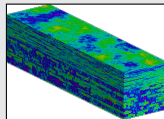
$$u(\theta_1^{(1)}, \dots, \theta_{j^{(L)}}^{(L)}) = \sum_{l=1}^L \sum_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}(\theta_j^{(l)})$$

Multiscale problems ...

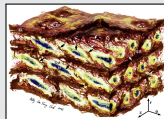
... appear in most areas of modern science and engineering:



Dual-phase steel;
fig. courtesy of
J. Schröder.



Groundwater flow;
cf. **Christie & Blunt**
(2001) (SPE10).



Arterial walls;
cf. **O'Connell et al.**
(2008).

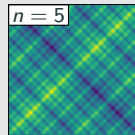
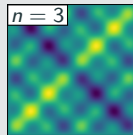
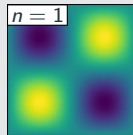
Multi-frequency problem

Consider the **multi-frequency Laplace problem**

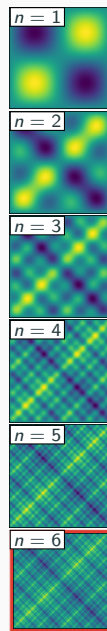
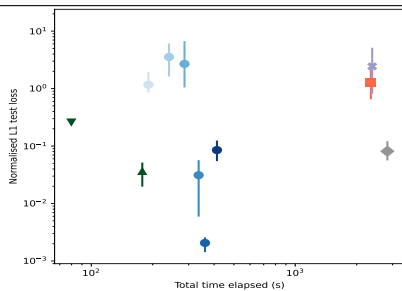
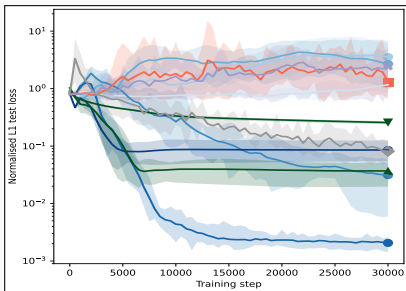
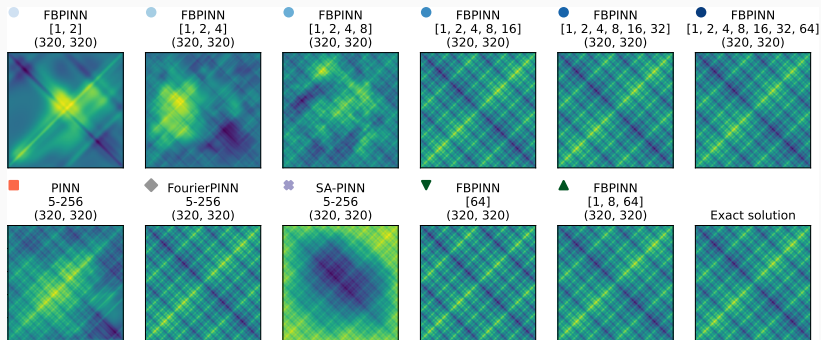
$$-\Delta u = 2 \sum_{i=1}^n (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y),$$

with homogeneous Dirichlet boundary conditions and $\omega_i = 2^i$.

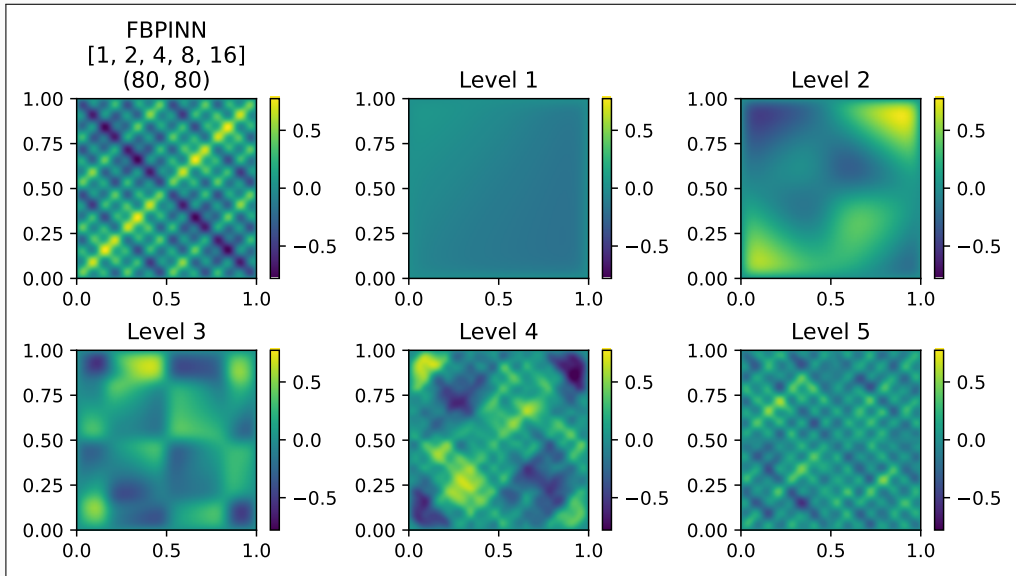
For increasing values of n , we obtain the **solutions**:



Multi-Level FBPINNs for a Multi-Frequency Problem – Strong Scaling



Multi-Frequency Problem – What the FBPINN Learns



Cf. [Dolean, Heinlein, Mishra, Moseley \(2024\)](#).

Enhanced training and extension to deep operator networks

Many popular optimizers are based on **gradient descent**:

$$\theta^{(m+1)} = \theta^{(m)} - \alpha \nabla_{\theta} \mathcal{L}(\theta^{(m)}),$$

where $\theta^{(m)}$ are the parameters at iteration m , α is the learning rate, and $\mathcal{L}$ is the loss function; a commonly used example is the **Adam** optimizer (**Kingma (2017)**).

The **Gauss–Newton method** is based on **Newton’s method** but uses the **Gramian** G as an **approximate Jacobian**:

$$\theta^{(m+1)} = \theta^{(m)} - \alpha \mathbf{G}^+(\theta^{(m)}) \nabla_{\theta} \mathcal{L}(\theta^{(m)}),$$

where $G^+(\theta)$ is the pseudoinverse of G , as it may **not** be **invertible** but **only symmetric positive semidefinite**. Therefore, we may regularize G and consider $G + \mu I$ instead, making the matrix **symmetric positive definite**.

Mean squared error (MSE) loss

$$\mathbf{G}(\theta)_{ij} = \sum_{\mathbf{x}_i \in \Omega} (\partial_{\theta_i} u_{\theta}(\mathbf{x}_i)) (\partial_{\theta_j} u_{\theta}(\mathbf{x}_i))$$

Physics-informed loss function

$$\mathbf{G}(\theta)_{ij} = \sum_{\mathbf{x}_i \in \Omega} \partial_{\theta_i} \mathcal{N}[u]_{\theta}(\mathbf{x}_i) \partial_{\theta_j} \mathcal{N}[u]_{\theta}(\mathbf{x}_i)$$

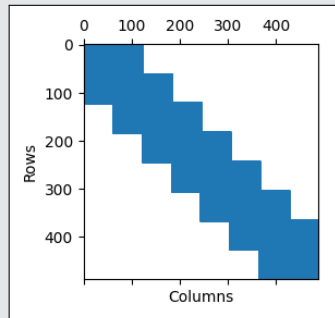
See, for instance, **Müller and Zeinhofer (2023)** and **Cai et al. (arXiv 2024)**.

Results for the ODE Problem

Sparsity

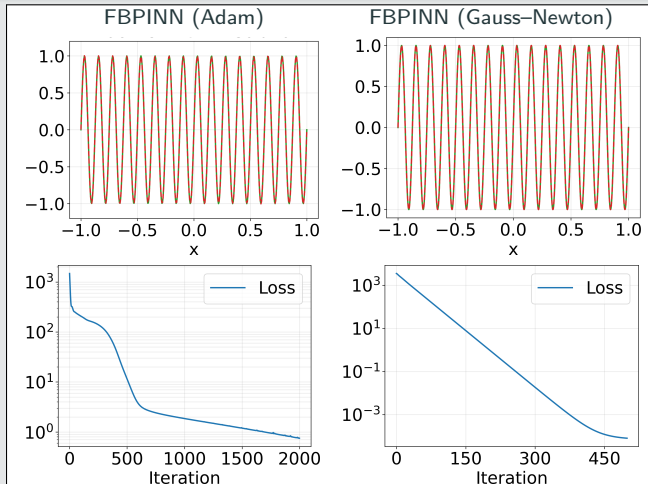
The domain decomposition introduces sparsity in G :

- 8 subdomains
- coupling blocks due to overlap between subdomains



Cf. Heinlein and Kapoor (acc. 2026).

Comparison of Adam and Gauss-Newton training



FBPINN with	Adam	Gauss-Newton
test MSE	7.8×10^{-3}	8.0×10^{-4}

Results for a Two-Dimensional Helmholtz Problem

Problem setup

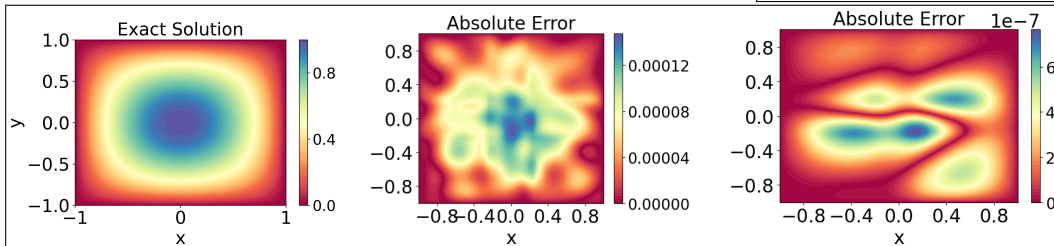
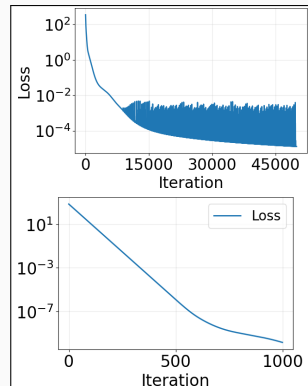
We consider a **low-wavenumber two-dimensional Helmholtz** problem on $\Omega = [-1, 1]^2$:

$$\Delta u(x, y) + k^2 u(x, y) = f(x, y), \text{ in } \Omega,$$

with homogeneous Dirichlet boundary conditions. For $k = 1$,

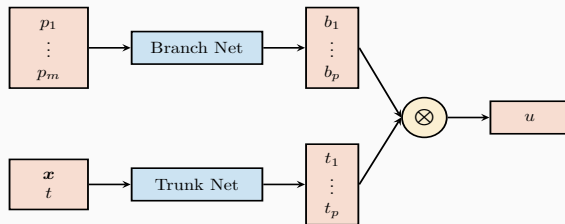
$$f(x, y) = -4 + 2(x^2 + y^2) + (1 - x^2)(1 - y^2).$$

which yields the analytical solution $u(x, y) = (1 - x^2)(1 - y^2)$.



Deep Operator Networks (DeepONets / DONs)

Neural operators learn operators between function spaces using neural networks. Here, we learn the **solution operator** of a initial-boundary value problem parametrized with $p_1, \dots, p_m$ using **DeepONets** as introduced in **Lu et al. (2021)**.



Single-layer case

The DeepONet architecture is based on the **single-layer case** analyzed in **Chen and Chen (1995)**. In particular, the authors show **universal approximation properties for continuous operators**.

The architecture is based on the following ansatz for presenting the parametrized solution

$$u_{(p_1, \dots, p_m)}(\mathbf{x}, t) = \sum_{i=1}^p \underbrace{b_i(p_1, \dots, p_m)}_{\text{branch}} \cdot \underbrace{t_i(\mathbf{x}, t)}_{\text{trunk}}$$

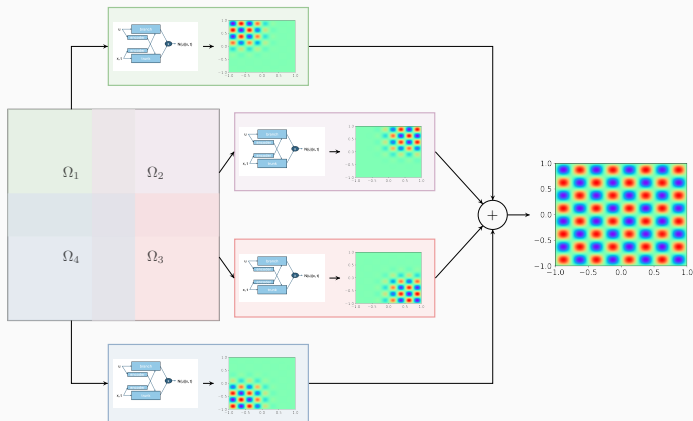
Physics-informed DeepONets

DeepONets are compatible with the PINN approach but **physics-informed DeepONets (PI-DeepONets)** are challenging to train.

Other operator learning approaches

- **FNOs**: **Li et al. (2021)**
- **PCA-Net**: **Bhattacharya et al. (2021)**
- **Random features**: **Nelsen and Stuart (2021)**
- **CNOs**: **Raonić et al. (2023)**

Finite Basis DeepONets (FBDONs)



Howard, Heinlein, Stinis (in prep.)

Variants:

Shared-trunk FBDONs (ST-FBDONs)

The trunk net learns spatio-temporal basis functions. In ST-FBDONs, we use the **same trunk network for all subdomains**.

Stacking FBDONs

Combination of the **stacking multifidelity approach** with FBDONs.

Heinlein, Howard, Beecroft, Stinis (2025)

Wave equation

$$\frac{d^2 s}{dt^2} = 2 \frac{d^2 s}{dx^2}, \quad (x, t) \in [0, 1]^2$$

$$s_t(x, 0) = 0, x \in [0, 1], \quad s(0, t) = s(1, t) = 0,$$

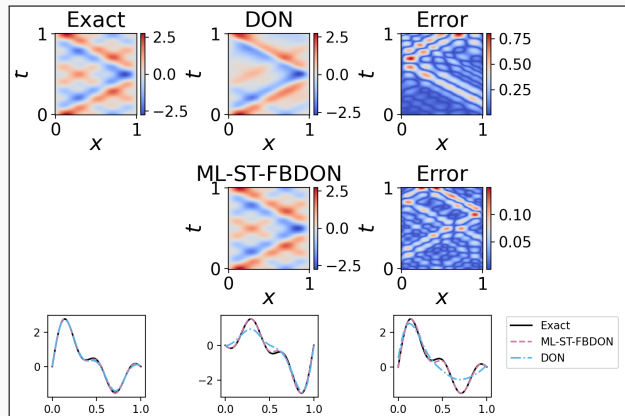
$$\text{Solution: } s(x, t) = \sum_{n=1}^5 b_n \sin(n\pi x) \cos(n\pi\sqrt{2}t)$$

Parametrization

Initial conditions for s parametrized by $b = (b_1, \dots, b_5)$ (normally distributed):

$$s(x, 0) = \sum_{n=1}^5 b_n \sin(n\pi x) \quad x \in [0, 1]$$

Training on 1000 random configurations.



Mean rel. l_2 error on 100 config.

DeepONet	0.30 ± 0.11
ML-ST-FBDON ([1, 4, 8, 16] subd.)	0.05 ± 0.03
ML-FBDON ([1, 4, 8, 16] subd.)	0.08 ± 0.04

→ Sharing the trunk network does not only save in the number of parameters but even yields **better performance**

Cf. Howard, Heinlein, Stinis (in prep.)

4TU.AMI – SRI “Bridging Numerical Analysis and Machine Learning”

UNIVERSITY
OF TWENTE.



Christoph
Brune



Silke Glas



Matthias
Schlottbom

 **TU Delft**
Delft
University of
Technology



Francesca
Bartolucci



Alexander
Heinlein



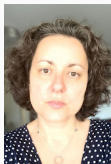
Matthias
Möller



Deepesh
Toshniwal

4TU+.AMI

TU/e EINDHOVEN
UNIVERSITY OF
TECHNOLOGY



Victorita
Dolean



Wil
Schilders



Jemima
Tabcart



Karen
Veroy-Grepl

 **WAGENINGEN**
UNIVERSITY & RESEARCH



Xiaodong
Cheng



Taniya
Kapoor

PINNs: localization & multilevel methods

- Neural networks can solve differential equations, but **scalability and conditioning are central bottlenecks**.
- Schwarz domain decomposition **improves scalability**; multilevel variants enable multiscale problems and scalability to many subdomains.

Training & extension to DeepONets

- **Training** efficiency and robustness improve with **second-order optimization methods**, such as Gauß–Newton methods.
- **DeepONets**: **efficient operator learning** for parametrized problems, but training can exhibit **spectral bias**; domain decomposition can **improve performance**.

Thank you for your attention!

