



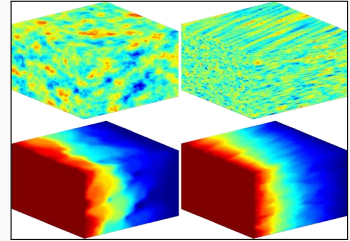
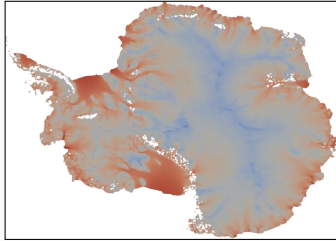
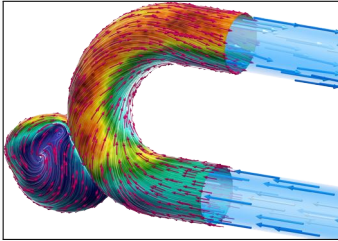
Domain Decomposition for Neural Networks

Alexander Heinlein¹

Scientific Computing Seminar, University of Kaiserslautern-Landau, Kaiserslautern, Germany, May 15, 2025

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 Multilevel domain decomposition-based architectures for physics-informed neural networks

Based on joint work with

Victorita Dolean
Siddhartha Mishra
Ben Moseley

(Eindhoven University of Technology)
(ETH Zürich)
(Imperial College London)

2 Domain decomposition for randomized neural networks

Based on joint work with

Siddhartha Mishra
Yong Shang and **Fei Wang**

(ETH Zürich)
(Xi'an Jiaotong University)

3 Domain decomposition-based physics-informed deep operator networks

Based on joint work with

Amanda A. Howard and **Panos Stinis**

(Pacific Northwest National Laboratory)

4 Domain decomposition-based image segmentation for high-resolution image segmentation on multiple GPUs

Based on joint work with

Eric Cyr
Corné Verburg

(Sandia National Laboratories)
(Delft University of Technology)

Multilevel domain decomposition-based architectures for physics-informed neural networks

Physics-Informed Neural Networks (PINNs) – Idea

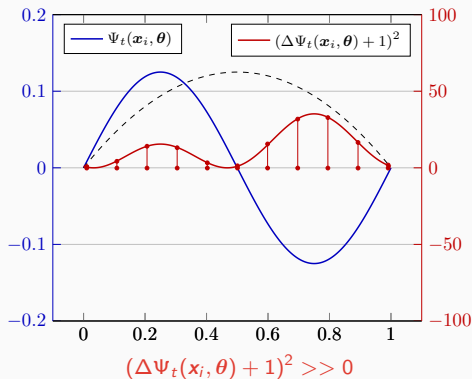
In **Lagaris et al. (1998)**, the authors solve the **boundary value problem**

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

$$\Psi_t(0, \theta) = \Psi_t(1, \theta) = 0,$$

via a **collocation approach**:

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

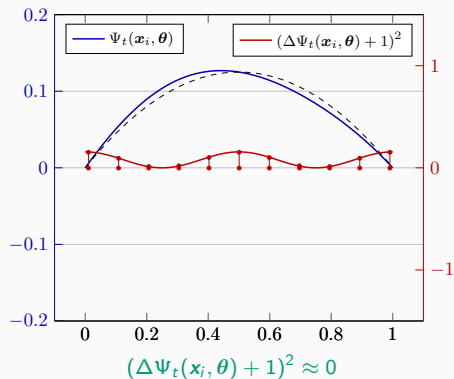


Boundary conditions ...

... 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

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

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

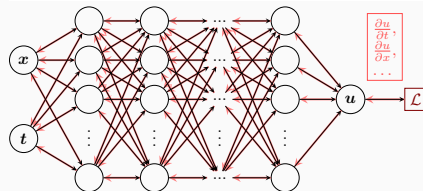
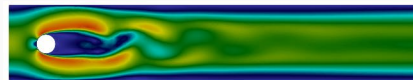
See also **Dissanayake and Phan-Thien (1994)**; **Lagaris et al. (1998)**.

Advantages

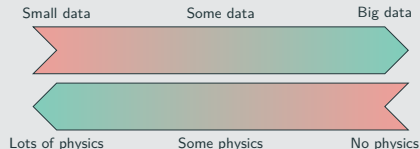
- **"Meshfree"**
- **Small data**
- **Generalization properties**
- **High-dimensional problems**
- **Inverse and parameterized problems**

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

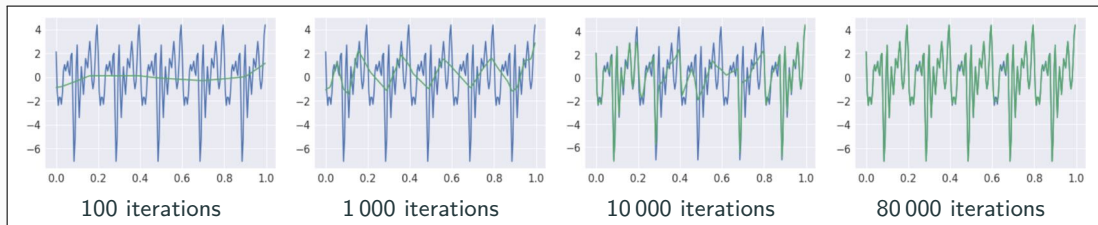
Estimate of the generalization error (Mishra and Molinaro (2022))

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), ...

Scaling of PINNs for a Simple ODE Problem

Solve

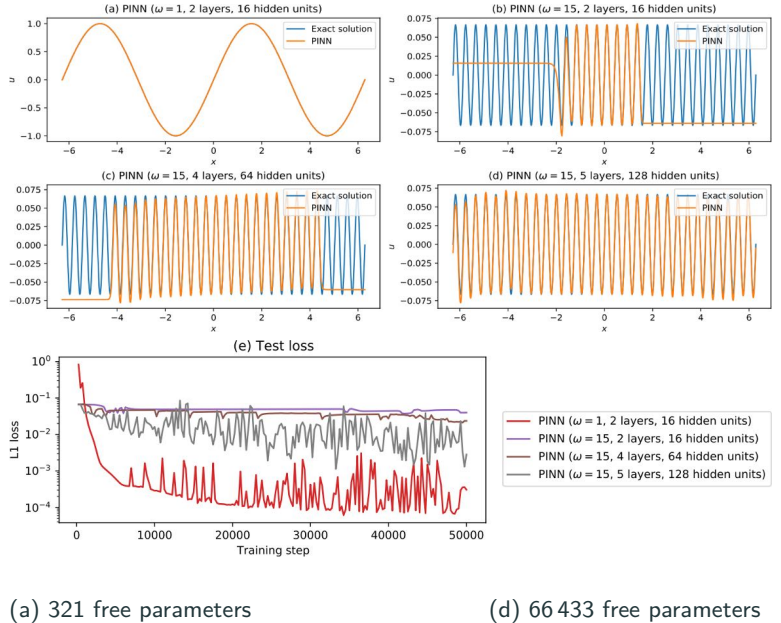
$$\begin{aligned}u' &= \cos(\omega x), \\ u(0) &= 0,\end{aligned}$$

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)



Scaling of PINNs for a Simple ODE Problem

Solve

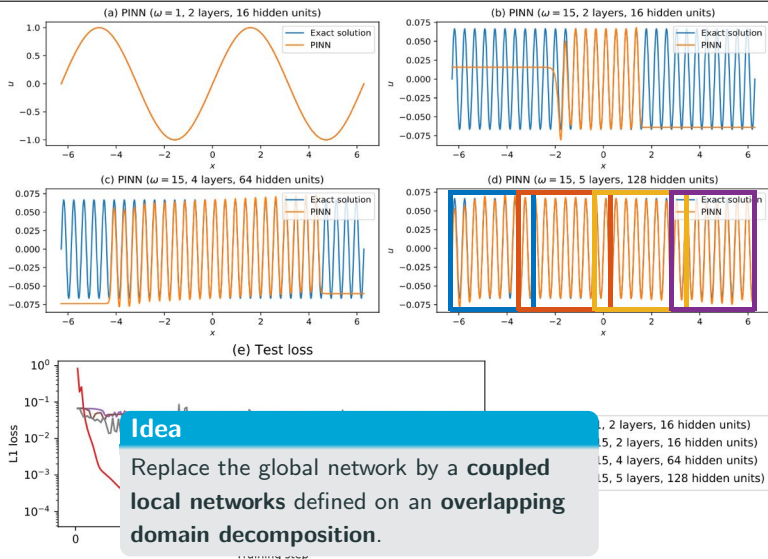
$$\begin{aligned}u' &= \cos(\omega x), \\ u(0) &= 0,\end{aligned}$$

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

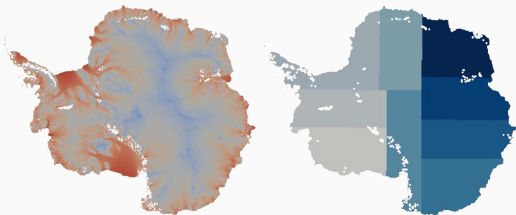
Scaling issues

- Large computational domains
- Small frequencies

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



Domain Decomposition Methods



Images based on Heinlein, Perego, Rajamanickam (2022)

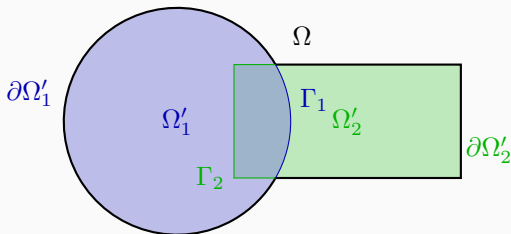
Historical remarks: The **alternating Schwarz method** is the earliest **domain decomposition method (DDM)**, which has been invented by **H. A. Schwarz** and published in **1870**:

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

Idea

Decomposing a large **global problem** into smaller **local problems**:

- **Better robustness** and **scalability** of numerical solvers
- **Improved computational efficiency**
- Introduce **parallelism**



A non-exhaustive literature overview:

- Machine Learning for adaptive BDDC, FETI-DP, and AGDSW: Heinlein, Klawonn, Lanser, Weber (2019, 2020, 2021, 2021, 2021, 2022); Klawonn, Lanser, Weber (2024)
- cPINNs, XPINNs: Jagtap, Kharazmi, Karniadakis (2020); Jagtap, Karniadakis (2020)
- Classical Schwarz iteration for PINNs or DeepRitz (D3M, DeepDDM, etc):: Li, Tang, Wu, and Liao (2019); Li, Xiang, Xu (2020); Mercier, Gratton, Boudier (arXiv 2021); Dolean, Heinlein, Mercier, Gratton (subm. 2024 / arXiv:2408.12198); Li, Wang, Cui, Xiang, Xu (2023); Sun, Xu, Yi (arXiv 2023, 2024); Kim, Yang (2023, 2024, 2024)
- FBPINNs, FBKANs: Moseley, Markham, Nissen-Meyer (2023); Dolean, H., Mishra, Moseley (2024, 2024); H., Howard, Beecroft, Stinis (2025); Howard, Jacob, Murphy, H., Stinis (arXiv 2024)
- DD for RaNNs, ELMS, Random Feature Method: Dong, Li (2021); Dang, Wang (2024); Sun, Dong, Wang (2024); Sun, Wang (2024); Chen, Chi, E, Yang (2022); Shang, H., Mishra, Wang (2025)
- DDMs for CNNs: Gu, Zhang, Liu, Cai (2022); Lee, Park, Lee (2022); Klawonn, Lanser, Weber (2024); Verburg, Heinlein, Cyr (2025)

An overview of the state-of-the-art 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)

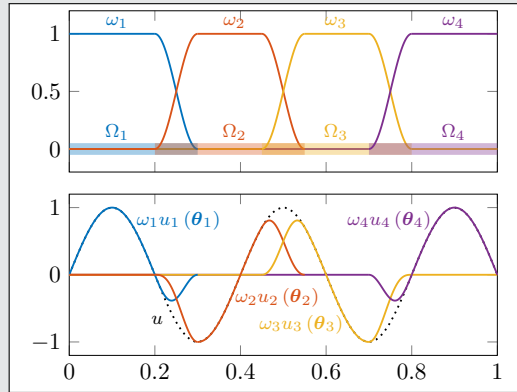
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

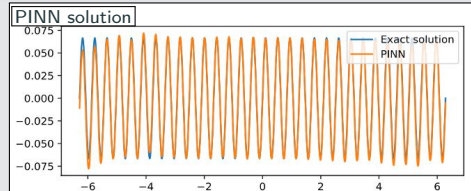
$$u(\theta_1, \dots, \theta_J) = \sum_{j=1}^J \omega_j u_j(\theta_j)$$

and the **loss function**

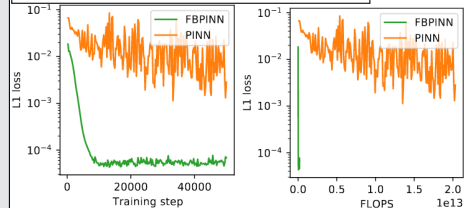
$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left(n \left[\sum_{\mathbf{x}_i \in \Omega_j} \omega_j u_j(\mathbf{x}_i, \theta_j) - f(\mathbf{x}_i) \right]^2 \right)$$



1D single-frequency problem



Moseley, Markham, Nissen-Meyer (2023)



Finite Basis Physics-Informed Neural Networks (FBPINNs)

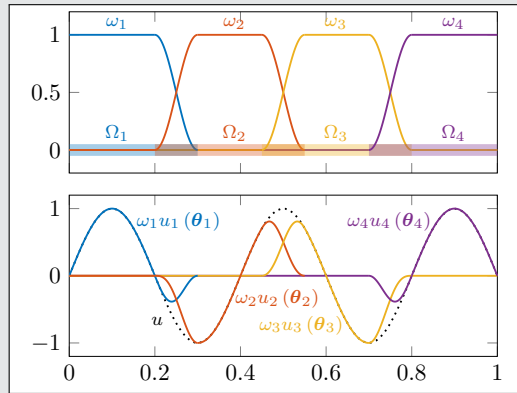
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

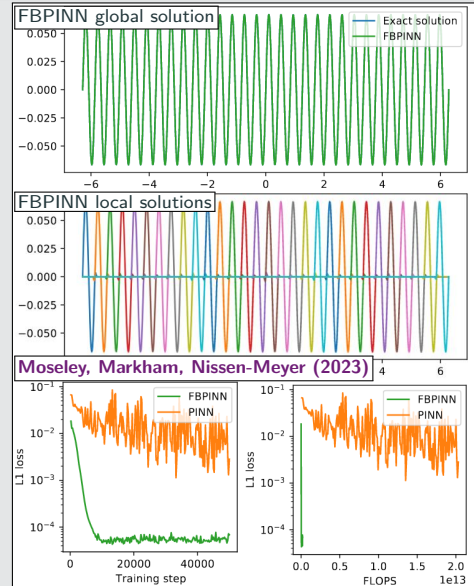
$$u(\theta_1, \dots, \theta_J) = \sum_{j=1}^J \omega_j u_j(\theta_j)$$

and the **loss function**

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

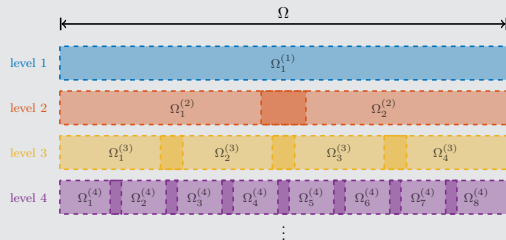


1D single-frequency problem



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**

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

and the **loss function**

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

Multi-Frequency Problem

Let us now consider the **two-dimensional multi-frequency Laplace boundary value problem**

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

$$u = 0 \quad \text{on } \partial\Omega,$$

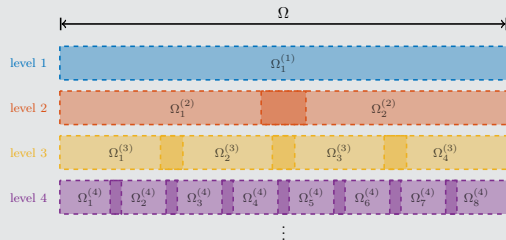
with $\omega_i = 2^i$.

For increasing values of n , we obtain the **analytical 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**

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

and the **loss function**

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

Multi-Frequency Problem

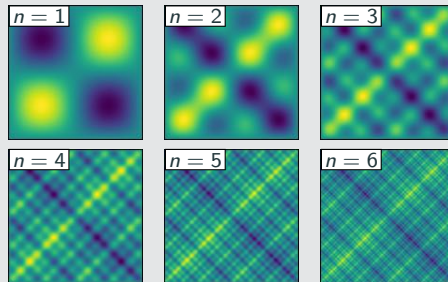
Let us now consider the **two-dimensional multi-frequency Laplace boundary value problem**

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

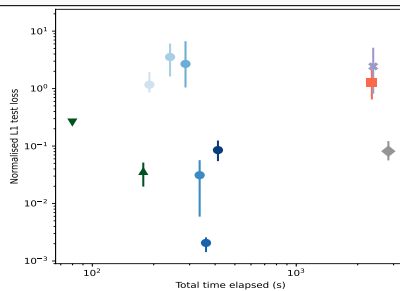
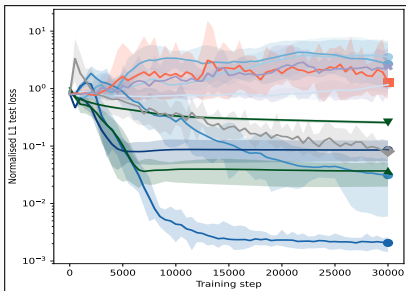
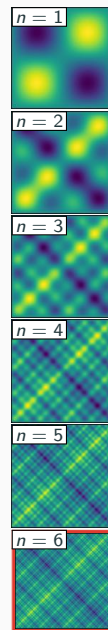
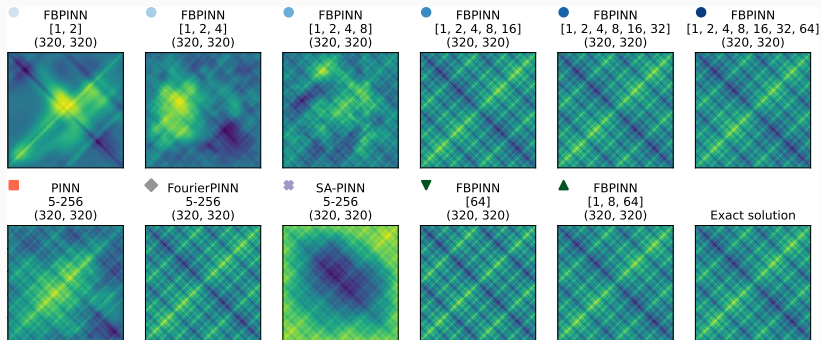
$$u = 0 \quad \text{on } \partial\Omega,$$

with $\omega_i = 2^i$.

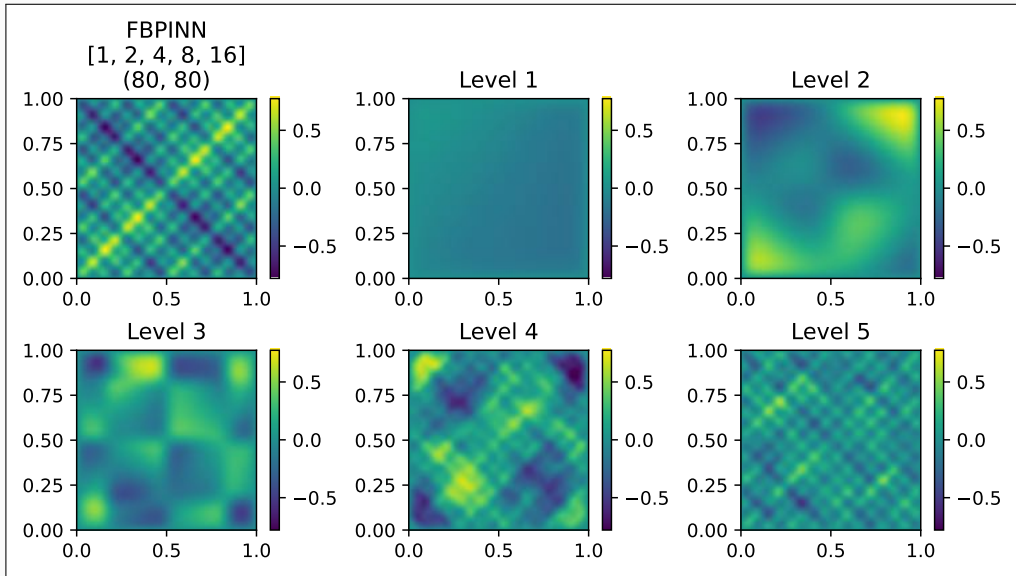
For increasing values of n , we obtain the **analytical solutions**:



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

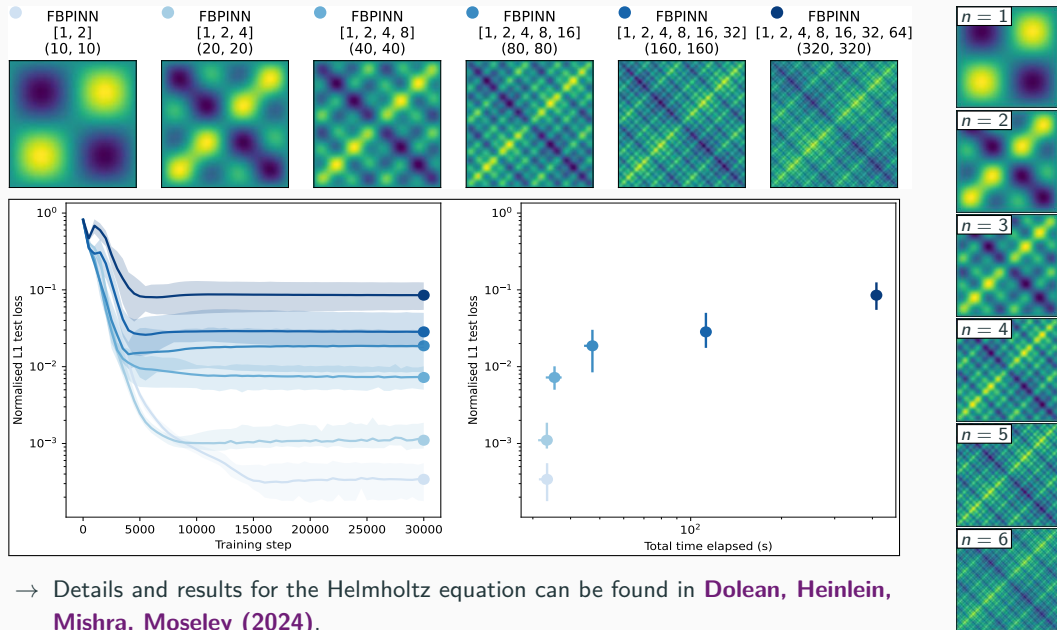


Multi-Frequency Problem – What the FBPINN Learns



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

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



→ Details and results for the Helmholtz equation can be found in [Dolean, Heinlein, Mishra, Moseley \(2024\)](#).

Domain decomposition for randomized neural networks

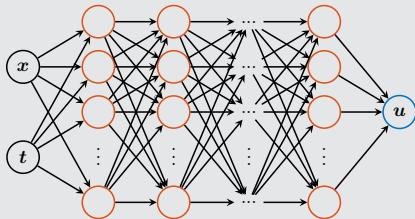
Randomized Neural Networks (RaNNs)

Neural networks

A standard **multilayer perceptron (MLP)** with L hidden layers is a **parametric** model of the form

$$u(\mathbf{x}, \theta) = F_{L+1}^{\mathbf{A}} \cdot F_L^{W_L, b_L} \circ \dots \circ F_1^{W_1, b_1}(\mathbf{x}),$$

where \mathbf{A} is **linear**, and the i th hidden layer is **nonlinear** $F_i^{W_i, b_i}(\mathbf{x}) = \sigma(\mathbf{W}_i \cdot \mathbf{x} + \mathbf{b}_i)$.



In order to optimize the loss function

$$\min_{\theta} \mathcal{L}(\theta),$$

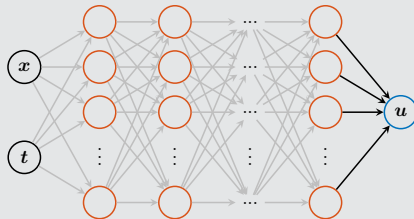
all parameters $\theta = (\mathbf{A}, \mathbf{W}_1, \mathbf{b}_1, \dots, \mathbf{W}_L, \mathbf{b}_L)$ are **trained**.

Randomized neural networks

In **randomized neural networks (RaNNs)** as introduced by **Pao and Takefuji (1992)**,

$$u(\mathbf{x}, \mathbf{A}) = F_{L+1}^{\mathbf{A}} \cdot F_L^{W_L, b_L} \circ \dots \circ F_1^{W_1, b_1}(\mathbf{x}),$$

the weights in the hidden layers are randomly initialized and **fixed**; only \mathbf{A} is trainable.



The model is **linear** with respect to the trainable parameters \mathbf{A} , and the optimization problem reads

$$\min_{\mathbf{A}} \mathcal{L}(\mathbf{A}).$$

This can **simplify the training process**.

Physics-Informed Randomized Neural Networks (PIRaNNs)

Physics-informed randomized neural networks (PIRaNNs) make use of the aforementioned linearization of the model with respect to the trainable parameters as well as the fact that RaNNs retain **universal approximation properties**, as shown in **Igel'nik and Pao (1995)**.

Consider a **linear differential operator** \mathcal{A} . Then, solving the PDE

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

using PIRaNNs yields the **linear equation system**

$$\mathcal{A}[u](\mathbf{x}_i) = f(\mathbf{x}_i), \quad i = 1, \dots, N_{\text{PDE}},$$

where N_{PDE} is the number of **collocation points**.

The resulting linear equation system

$$\mathbf{H}\mathbf{A} = \mathbf{f}$$

generally does **not have a unique solution**. In fact, \mathbf{H} is typically **rectangular**, **dense**, and **ill-conditioned**.

Solving the system using least squares corresponds to applying the **classical PINN loss function to the RaNN model** u . As we will see, this approach offers a **potentially efficient alternative**.

Enforcement of boundary conditions

We construct u to explicitly satisfy BCs:

$$u(\mathbf{x}, \mathbf{A}) = G(\mathbf{x}) + L(\mathbf{x})\mathcal{N}(\mathbf{x}, \mathbf{A})$$

- \mathcal{N} is a **feedforward neural network** with **trainable parameters** \mathbf{A}
- G and L are **fixed functions**, chosen such that u satisfies the boundary conditions

Domain Decomposition-Based PIRaNNs

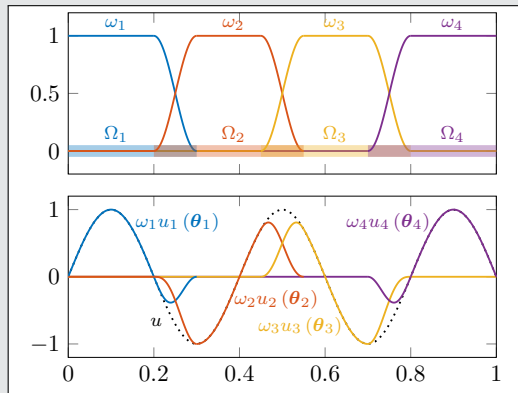
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

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**

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

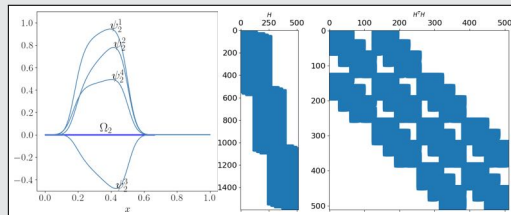


Domain decomposition for RaNNs

We employ the FBPINNs approach; cf. **Shang, Heinlein, Mishra, Wang (acc. 2025)**. This is closely related to the **random feature method (RFM)** by **Chen, Chi, E, Yang (2022)**. In particular, we solve

$$\mathcal{A} \left[\sum_{j=1}^J \omega_j u_j(\mathbf{A}_j) \right] (\mathbf{x}_i) = f(\mathbf{x}_i),$$

for $i = 1, \dots, N_{\text{PDE}}$; the boundary conditions are incorporated directly into the u_j .

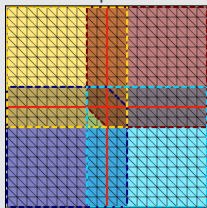


The hidden weights are randomly initialized, the resulting matrices \mathbf{H} and $\mathbf{H}^\top \mathbf{H}$ are block-sparse.

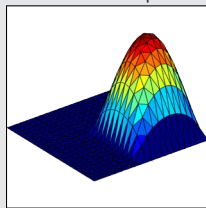
Preconditioning for Domain Decomposition-Based PIRaNNs

One-level Schwarz preconditioner

Overlap $\delta = 1h$



Solution of local problem



Based on an **overlapping domain decomposition**, we define a **one-level Schwarz operator** for $K := H^\top H$

$$M_{\text{OS-1}}^{-1} K = \sum_{i=1}^N R_i^\top K_i^{-1} R_i K,$$

where R_i and R_i^\top are restriction and prolongation operators corresponding to Ω'_i , and $K_i := R_i K R_i^\top$.

Here, the matrix K_i could be singular in which case we use a **pseudo inverse** K_i^+ instead of K_i^{-1} .

We also consider **restricted and scaled additive Schwarz preconditioners**; cf. **Cai, Sarkis (1999)**.

Singular Value Decomposition

As discussed before, on each subdomain Ω_j , the RaNN is

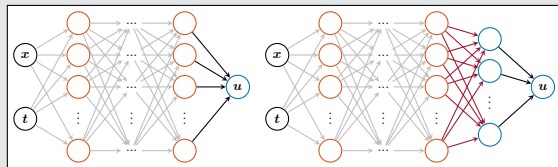
$$\begin{aligned} u_j(x, \mathbf{A}_j) &= F_{L+1}^{\mathbf{A}} \cdot F_L^{W_L, b_L} \circ \dots \circ F_1^{W_1, b_1}(x) \\ &= \mathbf{A}_j \begin{bmatrix} \Phi_1(x) & \dots & \Phi_k(x) \end{bmatrix}^\top, \end{aligned}$$

where k is the width of the last hidden layer and the Φ_l are the randomized basis functions.

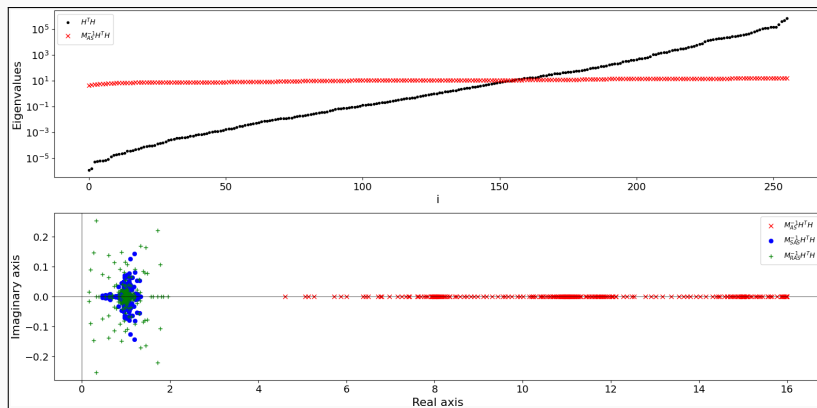
Consider a **reduced SVD** $\Phi = U \Sigma V^\top$, where the entries of the matrix are $\Phi_{i,l} = \Phi_l(x_i)$. Then, we consider

$$\hat{u}_j(x, \mathbf{A}_j) = \mathbf{A}_j \hat{\mathbf{V}}^\top \begin{bmatrix} \Phi_1(x) & \dots & \Phi_k(x) \end{bmatrix}^\top,$$

where $\hat{\mathbf{V}}^\top$ is obtained by omitting the right singular vectors corresponding to small singular values.

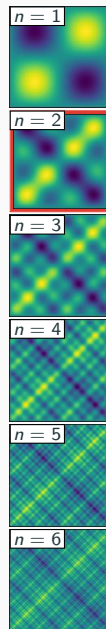


Results for the Multi-Frequency Problem ($n=2$)



	$M^{-1} = I$		$M^{-1} = M_{AS}^{-1}$		$M^{-1} = M_{RAS}^{-1}$		$M^{-1} = M_{SAS}^{-1}$	
	iter	e_{L^2}	iter	e_{L^2}	iter	e_{L^2}	iter	e_{L^2}
CG	> 2000	$1.95 \cdot 10^{-2}$	8	$5.03 \cdot 10^{-3}$	—	—	—	—
CGS	> 2000	$2.63 \cdot 10^{-2}$	4	$5.04 \cdot 10^{-3}$	24	$5.03 \cdot 10^{-3}$	6	$5.04 \cdot 10^{-3}$
BICG	> 2000	$1.03 \cdot 10^{-2}$	8	$5.08 \cdot 10^{-3}$	32	$5.05 \cdot 10^{-3}$	11	$5.09 \cdot 10^{-3}$
GMRES	> 2000	$8.68 \cdot 10^{-2}$	13	$5.07 \cdot 10^{-3}$	31	$5.06 \cdot 10^{-3}$	11	$5.08 \cdot 10^{-3}$

4×4 subdomains; DoF = 256; $N = 1600$; $\theta^0 \in \mathcal{U}(-1, 1)$; stop.: $\|\mathbf{M}^{-1} \mathbf{r}^k\|_{L^2} / \|\mathbf{M}^{-1} \mathbf{r}^0\|_{L^2} \leq 10^{-5}$



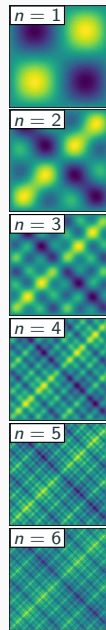
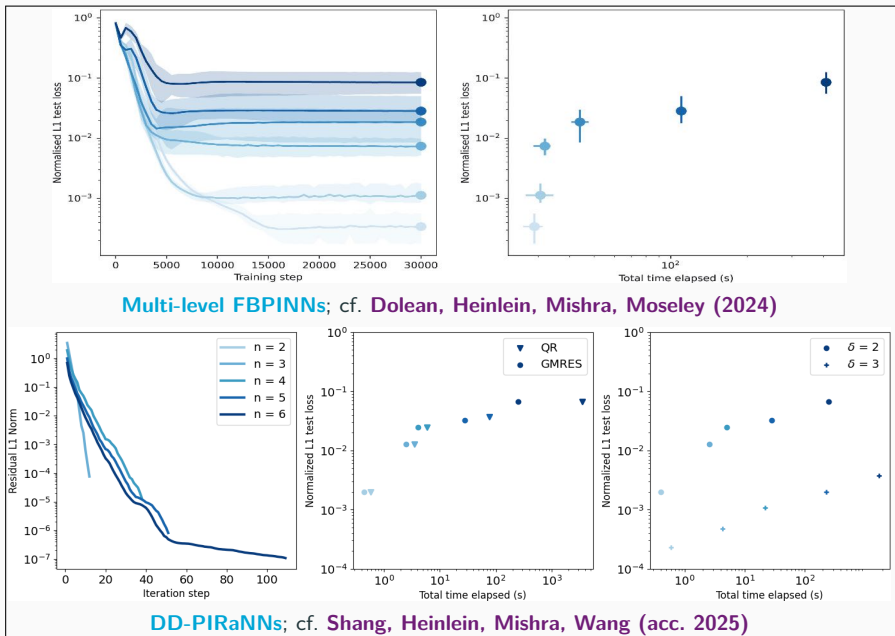
Results for the Multi-Frequency Problem ($n=2$) – Effect of the SVD

We now investigate the effect of omitting right singular vectors associated with singular values below a varying tolerance τ .

τ	DoF	M^{-1}	σ_{min}	σ_{max}	iter	e_{L^2}
10^{-4}	512	I	10^{-10}	10^6	> 2000	$3.72 \cdot 10^{-2}$
		M_{AS}^{-1}	10^{-6}	10^6	27	$5.46 \cdot 10^{-5}$
		M_{SAS}^{-1}	10^{-7}	10^5	30	$5.49 \cdot 10^{-5}$
10^{-3}	436	I	10^{-8}	10^5	> 2000	$3.75 \cdot 10^{-2}$
		M_{AS}^{-1}	10^{-5}	10^5	16	$1.28 \cdot 10^{-4}$
		M_{SAS}^{-1}	10^{-6}	10^4	18	$1.28 \cdot 10^{-4}$
10^{-2}	335	I	10^{-5}	10^5	> 2000	$4.51 \cdot 10^{-2}$
		M_{AS}^{-1}	10^{-3}	10^4	14	$7.14 \cdot 10^{-4}$
		M_{SAS}^{-1}	10^{-4}	10^3	13	$7.11 \cdot 10^{-4}$
10^{-1}	212	I	10^{-3}	10^6	> 2000	$5.01 \cdot 10^{-2}$
		M_{AS}^{-1}	10^{-2}	10^3	12	$7.13 \cdot 10^{-3}$
		M_{SAS}^{-1}	10^{-3}	10^2	11	$7.10 \cdot 10^{-3}$

4×4 subdomains; $N = 1600$; $\theta^0 \in \mathcal{U}(-1, 1)$; stop.: $\|\mathbf{M}^{-1}\mathbf{r}^k\|_{L^2}/\|\mathbf{M}^{-1}\mathbf{r}^0\|_{L^2} \leq 10^{-5}$

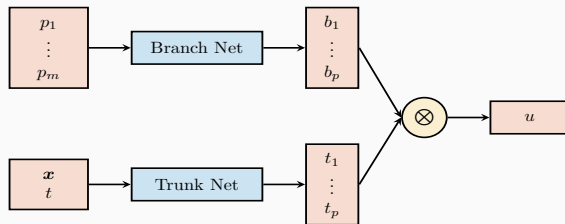
Results for the Multi-Frequency Problem



Domain decomposition-based physics-informed deep operator networks

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

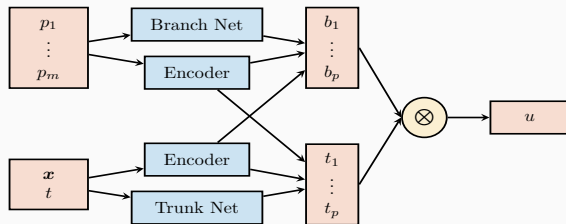
DeepONets are compatible with the PINN approach but **physics-informed DeepONets (PI-DeepONets)** are challenging to train.

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)**

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)**.



Modified architecture

In our numerical experiments, we employ the **modified DeepONet architecture** introduced in **Wang, Wang, and Perdikaris (2022)**.

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

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

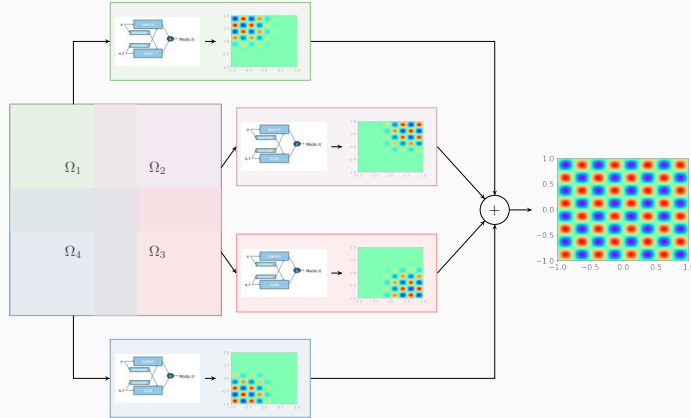
Physics-informed DeepONets

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

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

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

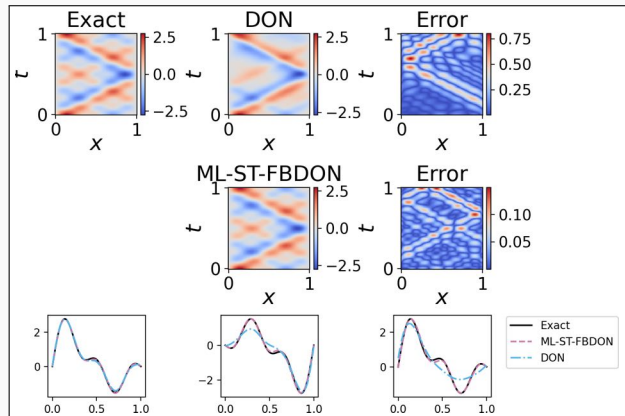
$$\text{Solution: } s(x, t) = \sum_{n=1}^5 b_n \sin(n\pi x) \cos(n\pi\sqrt{2}t)$$

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.

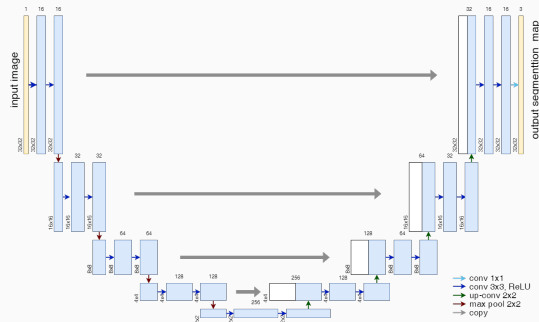
DeepONet	0.30 ± 0.11
ML-ST-FBDON ([1, 4, 8, 16] subd.)	0.05 ± 0.03
ML-FBDON ([1, 4, 8, 16] subd.)	0.08 ± 0.04

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

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

Domain decomposition-based image segmentation for high-resolution image segmentation on multiple GPUs

Memory Requirements for CNN Training

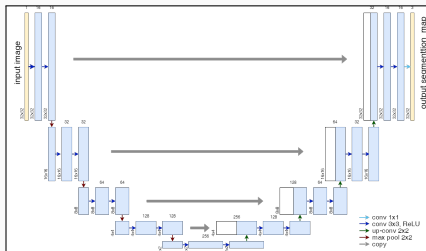


- As an example for a **convolutional neural network (CNN)**, we employ the **U-Net architecture** introduced in **Ronneberger, Fischer, and Brox (2015)**.
- The U-Net yields **state-of-the-art accuracy** in **semantic image segmentation** and other **image-to-image tasks**.

Below: memory consumption for training on a single 1024×1024 image.

name	size	# channels		mem. feature maps		mem. weights	
		input	output	# of values	MB	# of values	MB
input block	1 024	3	64	268 M	1 024.0	38 848	0.148
encoder block 1	512	64	128	167 M	704.0	221 696	0.846
encoder block 2	256	128	256	84 M	352.0	885 760	3.379
encoder block 3	128	256	512	42 M	176.0	3 540 992	13.508
encoder block 4	64	512	1 024	21 M	88.0	14 159 872	54.016
decoder block 1	64	1,024	512	50 M	192.0	9 177 088	35.008
decoder block 2	128	512	256	101 M	384.0	2 294 784	8.754
decoder block 3	256	256	128	201 M	768.0	573 952	2.189
decoder block 4	512	128	64	402 M	1 536.0	143 616	0.548
output block	1 024	64	3	3.1 M	12.0	195	0.001

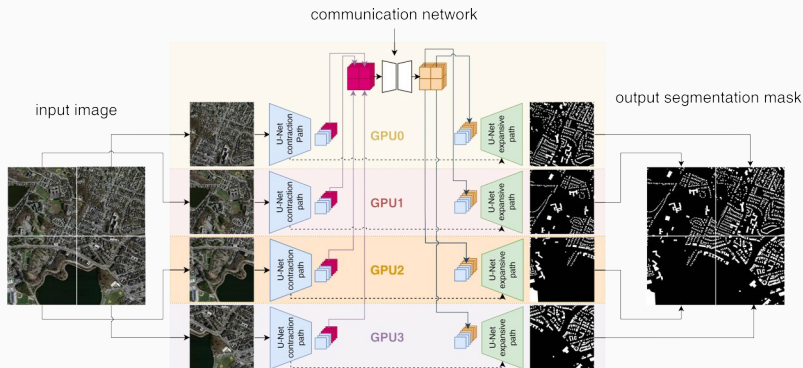
Domain Decomposition-Based U-Net Architecture



name	mem. feature maps		mem. weights	
	# of values	MB	# of values	MB
input block	268 M	1 024.0	38 848	0.148
encoder blocks	314 M	1 320	18 M	72
decoder blocks	754 M	3880	12 M	47
output block	3.1 M	12.0	195	0.001

Most memory in the **U-Net** is used by **feature maps**, not weights
 → **Decompose** feature maps to **distribute memory consumption**.

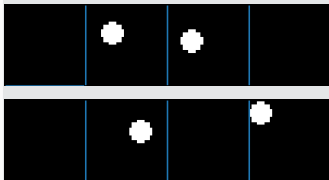
Cf. **Verburg, Heinlein, Cyr (2025)**.



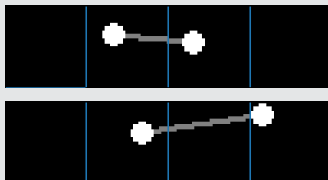
Results – Synthetic Data Set

Task: Connect two dots via a line segment

Input

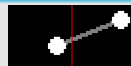


Target (segmentation mask)



Result: Communication

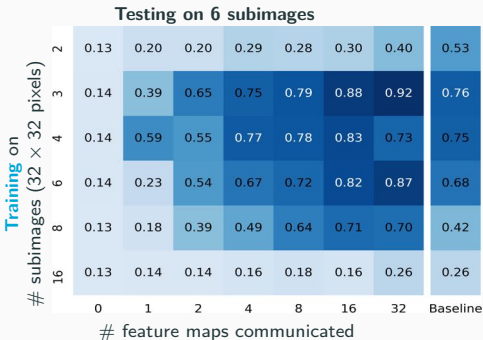
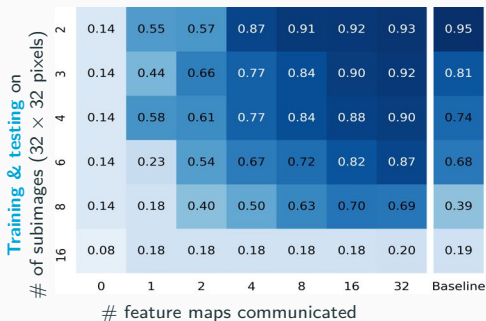
True mask



Pred. (no comm.)



Pred. (comm.)



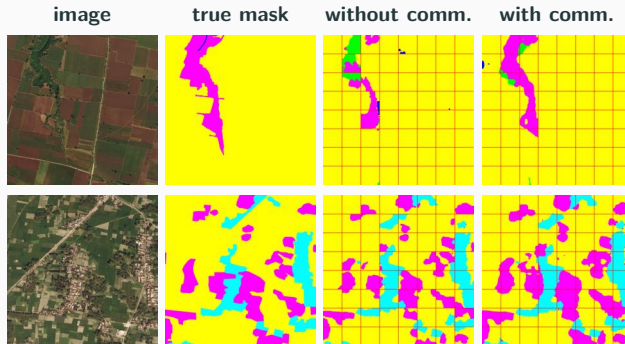
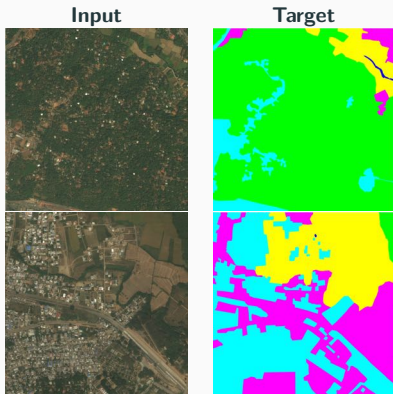
DeepGlobe 2018 Satellite Image Data Set (Demir et al. (2018))

class	pixel count	proportion
urban	642.4M	9.35 %
agriculture	3898.0M	56.76 %
rangeland	701.1M	10.21 %
forest	944.4M	13.75 %
water	256.9M	3.74 %
barren	421.8M	6.14 %
unknown	3.0M	0.04 %

Avoiding overfitting

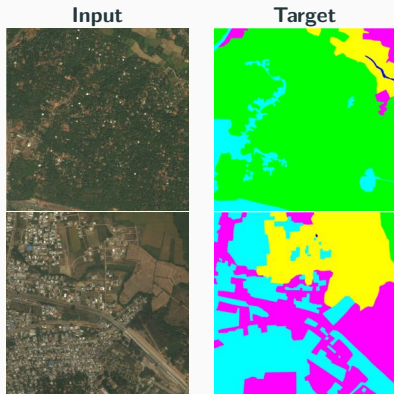
The data set includes **only 803 images**. To **avoid overfitting**, we

- apply **batch normalization**, use **random dropout** layers and **data augmentation**, and
- **initialize the encoder** using the **ResNet-18** (He, Zhang, Ren, and Sun (2016))



DeepGlobe 2018 Satellite Image Data Set (Demir et al. (2018))

class	pixel count	proportion
urban	642.4M	9.35 %
agriculture	3898.0M	56.76 %
rangeland	701.1M	10.21 %
forest	944.4M	13.75 %
water	256.9M	3.74 %
barren	421.8M	6.14 %
unknown	3.0M	0.04 %



Avoiding overfitting

The data set includes **only 803 images**. To **avoid overfitting**, we

- apply **batch normalization**, use **random dropout** layers and **data augmentation**, and
- **initialize the encoder** using the **ResNet-18** (He, Zhang, Ren, and Sun (2016))



Scientific Machine Learning in Academia and Beyond:

From Theory to Real-World Impact (in Industry)

- **Dates:** June 17, 2025, 12.30–17.30
- **Location:** Crowne Plaza Hotel, Utrecht
- **Lunch & networking:** 12.30–13.30; closing discussion & drinks to follow.
- An afternoon with **talks, case studies, and lively discussions** on advancing scientific machine learning from theory to real-world deployment — tackling core challenges like *uncertainty quantification, data assimilation, graph-based modelling, and operator learning*.
- **Confirmed plenary speakers:**
 - [Max Welling](#) (UvA, CUSP AI)
 - [Stefan Kurz](#) (ETH Zürich & Bosch)
 - [Koen Strien](#) (Ignition Computing)
 - [Maxim Pisarenko](#) (ASML)
 - [Jan Willem van de Meent](#) (UvA)



**Computational
Science NL**



CWI Research Semester Programme:

Bridging Numerical Analysis and Scientific Machine Learning: Advances and Applications

Co-organizers: Victorita Dolean (TU/e), Alexander Heinlein (TU Delft), Benjamin Sanderse (CWI), Jemima Tabbart (TU/e), Tristan van Leeuwen (CWI)

- **Autumn School** (October 27–31, 2025):
 - [Chris Budd](#) (University of Bath)
 - [Ben Moseley](#) (Imperial College London)
 - [Gabriele Steidl](#) (Technische Universität Berlin)
 - [Andrew Stuart](#) (California Institute of Technology)
 - [Andrea Walther](#) (Humboldt-Universität zu Berlin)
 - [Ricardo Baptista](#) (University of Toronto)
- **Workshop** (December 1–3, 2025):
 - 3 days with plenary talks (academia & industry) and an industry panel
 - Confirmed plenary speakers:
 - [Marta d'Elia](#) (Atomic Machines)
 - [Benjamin Peherstorfer](#) (New York University)
 - [Andreas Roskopf](#) (Fraunhofer Institute)



Centrum Wiskunde & Informatica



Join us for inspiring talks, hands-on sessions, and industry collaboration!

Summary

Multilevel Finite Basis Physics Informed Neural Networks (ML-FBPINNs)

- Schwarz domain decomposition architectures **improve the scalability of PINNs** to large domains / high frequencies, **keeping the complexity of the local networks low**.
- As classical domain decomposition methods, **one-level FBPINNs** are **not scalable to large numbers of subdomains**; multilevel FBPINNs **enable scalability**.

Extensions to Stacking Multifidelity PINNs, RaNNs, and DeepONets

- Multifidelity stacking PINNs with FBPINNs improve **accuracy and efficiency** for time-dependent problems.
- RaNNs reduce computational cost but face **ill-conditioning**, mitigated by **Schwarz preconditioning** and **SVD**.
- DeepONets provide **efficient predictions** for parametrized problems but struggle with multiscale problems. Domain decomposition **improves scalability and performance**.
- Novel DDU-Net approach **decouples the training on the sub-images**, allowing us to **distribute the memory load** among multiple GPUs. It **limits communication** to deepest level of the U-Net architecture using a **communication network**.

Thank you for your attention!



Topical Activity
Group

