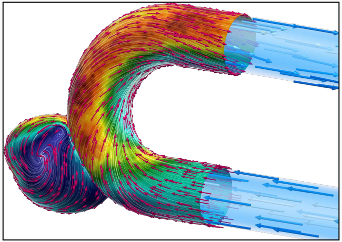
Domain Decomposition for Neural Networks

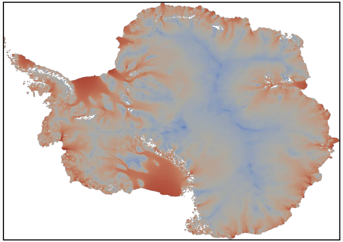
Alexander Heinlein[1]

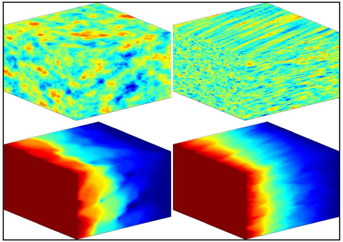
Scientific seminar, ENSEEIHT, Toulouse, France, May 22, 2025

[1]Delft 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

Outline

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

   Based on joint work with

Victorita Dolean	(Eindhoven University of Technology)
Siddhartha Mishra	(ETH Zürich)
Ben Moseley	(Imperial College London)

2. Domain decomposition for randomized neural networks

   Based on joint work with

Siddhartha Mishra	(ETH Zürich)
Yong Shang and **Fei Wang**	(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	(Sandia National Laboratories)
Corné Verburg	(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**

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

$$\Psi_t(0, \boldsymbol{\theta}) = \Psi_t(1, \boldsymbol{\theta}) = 0,$$

via a **collocation approach**:

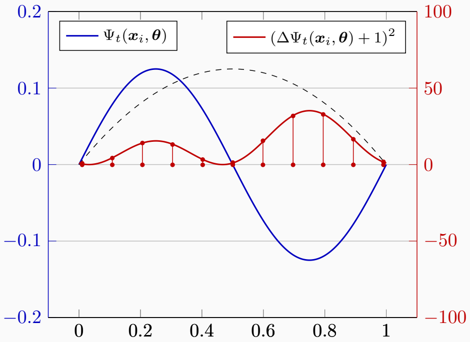
$$\min_{\boldsymbol{\theta}} \sum_{\boldsymbol{x}_i} \left(\Delta \Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}) + 1\right)^2$$

Boundary conditions ...

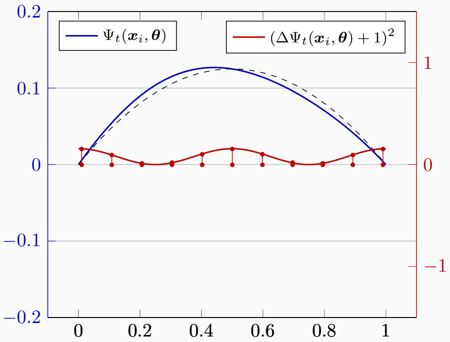
... can be **enforced explicitly** via the ansatz:

$$\Psi_t(\boldsymbol{x}, \boldsymbol{\theta}) = A(\boldsymbol{x}) + F(\boldsymbol{x}, \mathrm{NN}(\boldsymbol{x}, \boldsymbol{\theta}))$$

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



$(\Delta \Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}) + 1)^2 >> 0$



$(\Delta \Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}) + 1)^2 \approx 0$

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**:

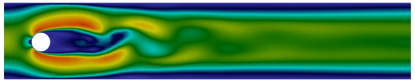
$$\mathcal{L}(\boldsymbol{\theta}) = \omega_{\text{data}} \mathcal{L}_{\text{data}}(\boldsymbol{\theta}) + \omega_{\text{PDE}} \mathcal{L}_{\text{PDE}}(\boldsymbol{\theta}),$$

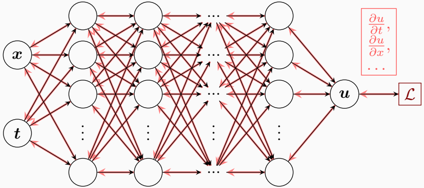
where ω_{data} and ω_{PDE} are **weights** and

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

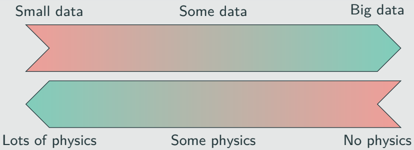
$$\mathcal{L}_{\text{PDE}}(\boldsymbol{\theta}) = \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} \left(\mathcal{N}[u](\mathbf{x}_i, \boldsymbol{\theta}) - f(\mathbf{x}_i) \right)^2.$$

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





Hybrid loss



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

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**

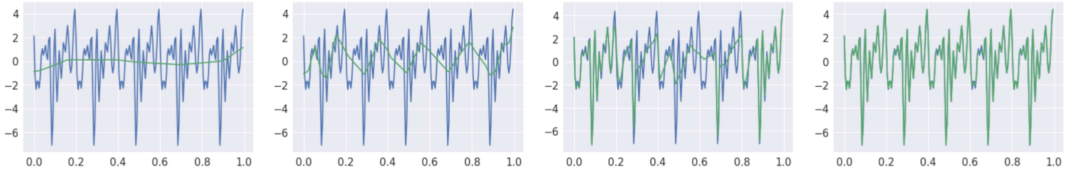
Estimate of the generalization error (Mishra and Molinaro (2022))

The generalization error (or total error) satisfies

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

- $\mathcal{E}_G = \mathcal{E}_G(\boldsymbol{X}, \theta) := \|\mathbf{u} - \mathbf{u}^*\|_V$ **general. error** (V Sobolev space, $\boldsymbol{X}$ 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"**



100 iterations | 1 000 iterations | 10 000 iterations | 80 000 iterations

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

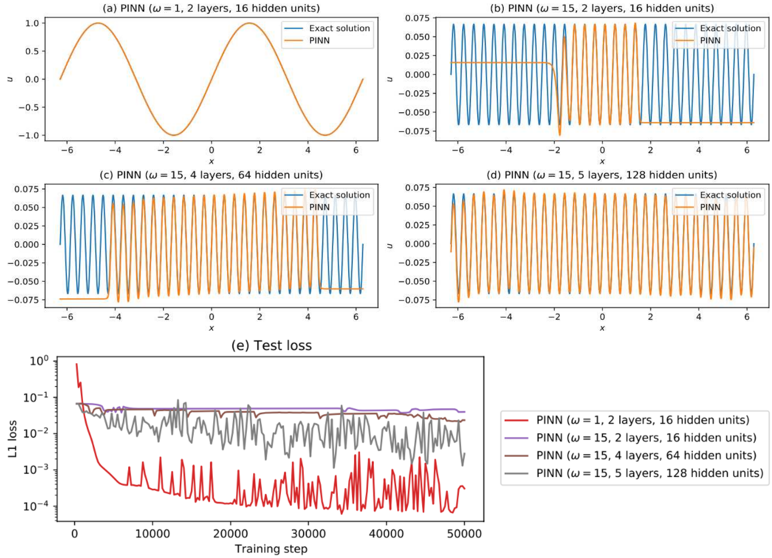
Related works: Cao et al. (2021), Wang, et al. (2022), Hong et al. (arXiv 2022), Xu et al (2024), . . .

Scaling of PINNs for a Simple ODE Problem

Solve

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

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



(a) 321 free parameters (d) 66 433 free parameters

Scaling issues

- Large computational domains
- Small frequencies

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

Scaling of PINNs for a Simple ODE Problem

Solve

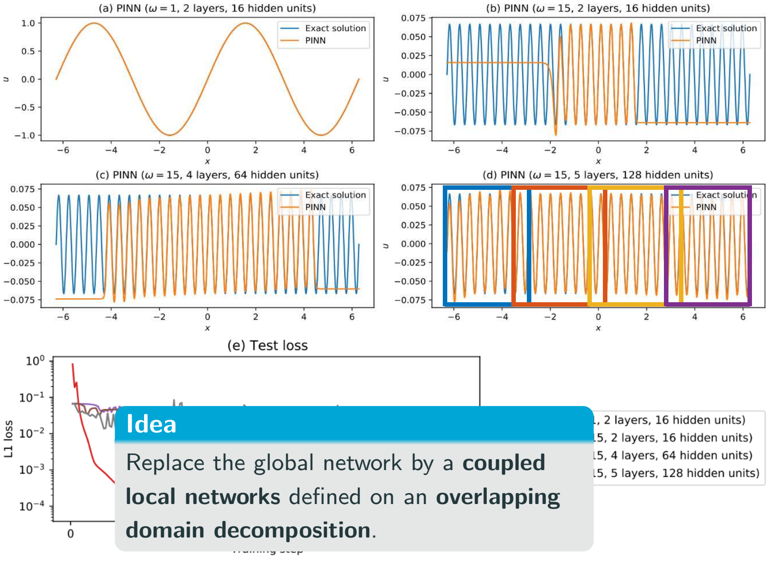
$$\begin{aligned} u' &= \cos(\omega \boldsymbol{x}), \\ u(0) &= 0, \end{aligned}$$

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)**

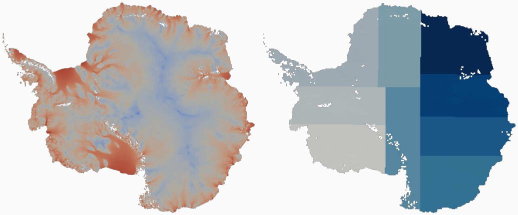


Idea

Replace the global network by a **coupled local networks** defined on an **overlapping domain decomposition**.

(a) 321 free parameters (d) 66 433 free parameters

Domain Decomposition Methods



Images based on **Heinlein, Perego, Rajamanickam (2022)**

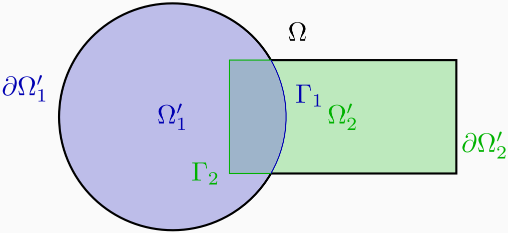
Historical remarks: The **alternating Schwarz method** is the earliest **domain decomposition method (DDM)**, which has been invented by **H. A. Schwarz** and published in **1870**:

- 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**



Domain Decomposition Methods and Machine Learning – Literature

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, Heinlein, Mishra, Moseley (2024, 2024); Heinlein, Howard, Beecroft, Stinis (2025); Howard, Jacob, Murphy, Heinlein, 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)

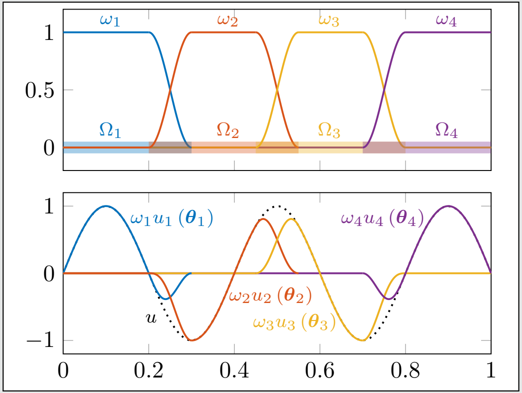
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

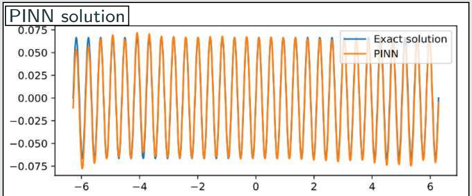
$$u(\theta_1, \ldots, \theta_J) = \sum_{j=1}^{J} \omega_j u_j(\theta_j)$$

and the **loss function**

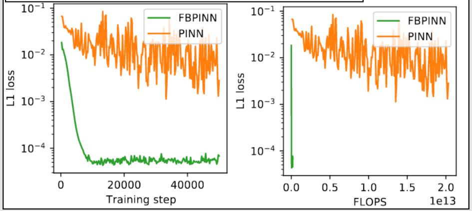
$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left(n[\sum_{x_i \in \Omega_j} \omega_j u_j](x_i, \theta_j) - f(x_i) \right)^2.$$



1D single-frequency problem



Moseley, Markham, Nissen-Meyer (2023)



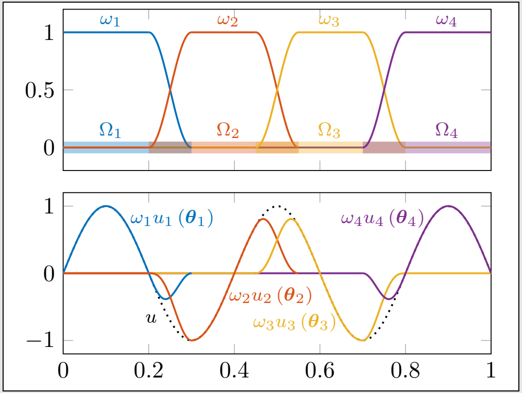
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

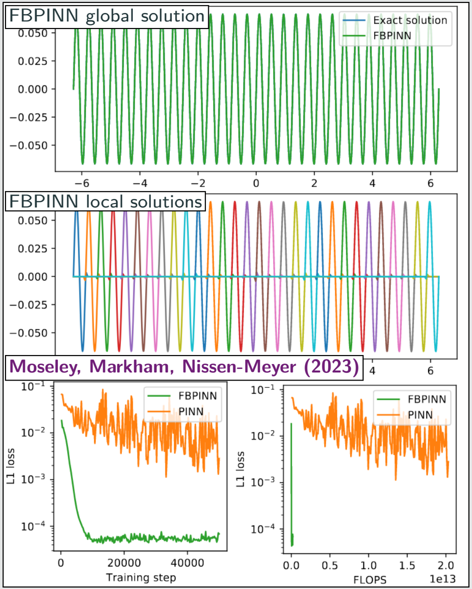
$$u(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J) = \sum_{j=1}^{J} \omega_j u_j(\boldsymbol{\theta}_j)$$

and the **loss function**

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

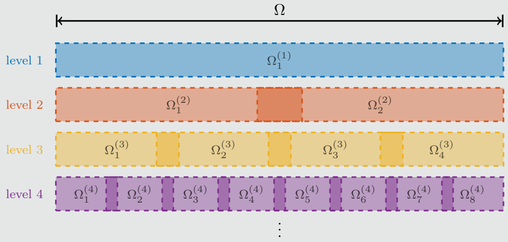


1D single-frequency problem



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**

$$u\left(\boldsymbol{\theta}_1^{(1)}, \ldots, \boldsymbol{\theta}_{J^{(L)}}^{(L)}\right) = \sum\nolimits_{l=1}^{L} \sum\nolimits_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}\left(\boldsymbol{\theta}_j^{(l)}\right)$$

and the **loss function**

$$\mathcal{L} = \frac{1}{N} \sum\nolimits_{i=1}^{N} \left(n\Big[\sum\nolimits_{\mathbf{x}_i \in \Omega_j^{(l)}} \omega_j^{(l)} u_j^{(l)}\Big](\mathbf{x}_i, \boldsymbol{\theta}_j^{(l)}) - f(\mathbf{x}_i)\right)^2.$$

Multi-Frequency Problem

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

$$\begin{aligned} -\Delta u &= 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

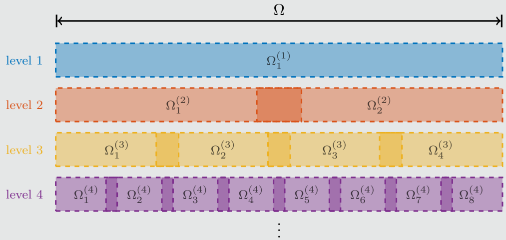
with $\omega_i = 2^i$.

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

Multi-Level FBPINNs

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**

$$u\left(\boldsymbol{\theta}_1^{(1)}, \ldots, \boldsymbol{\theta}_{J^{(L)}}^{(L)}\right) = \sum\nolimits_{l=1}^{L} \sum\nolimits_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}\left(\boldsymbol{\theta}_j^{(l)}\right)$$

and the **loss function**

$$\mathcal{L} = \frac{1}{N} \sum\nolimits_{i=1}^{N} \left(n\left[\sum\nolimits_{\mathbf{x}_i \in \Omega_j^{(l)}} \omega_j^{(l)} u_j^{(l)}\right](\mathbf{x}_i, \boldsymbol{\theta}_j^{(l)}) - f(\mathbf{x}_i)\right)^2.$$

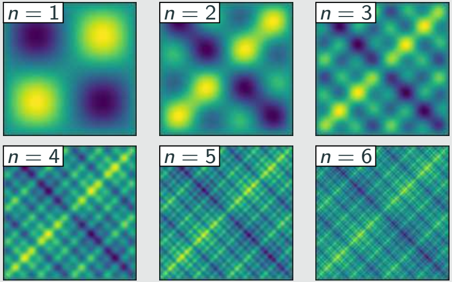
Multi-Frequency Problem

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

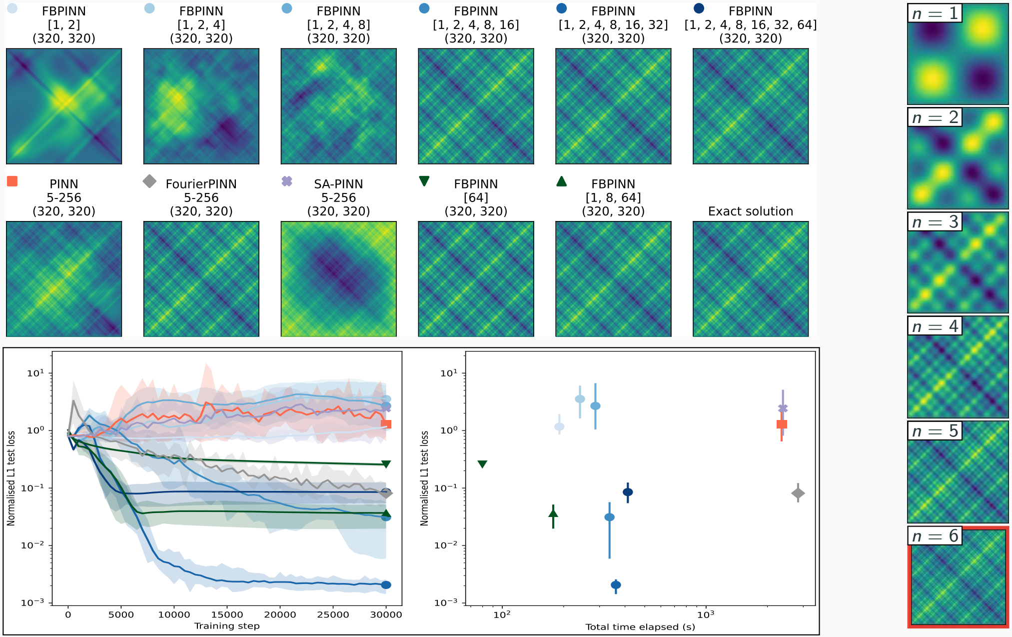
$$\begin{aligned} -\Delta u &= 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

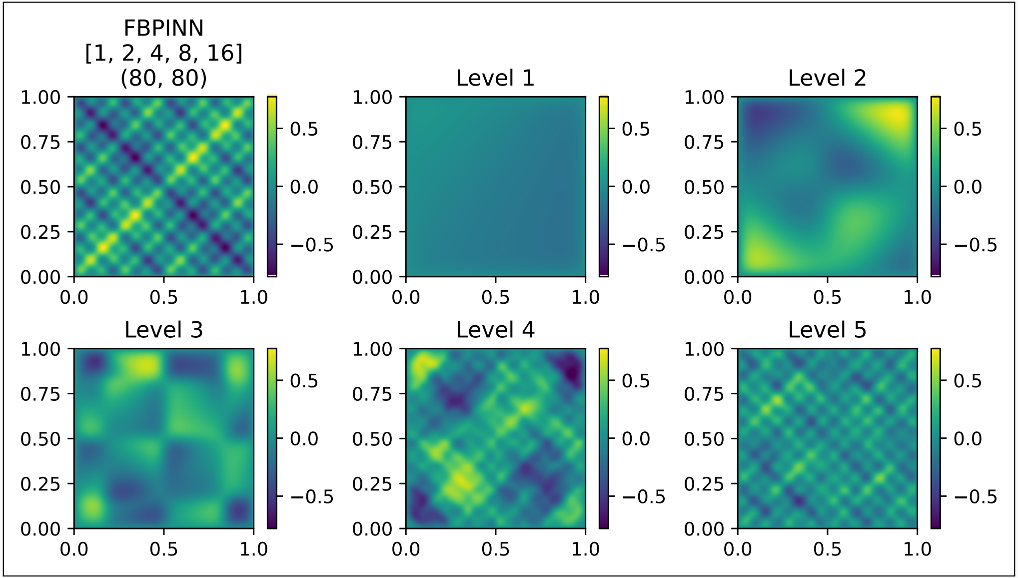
with $\omega_i = 2^i$.

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

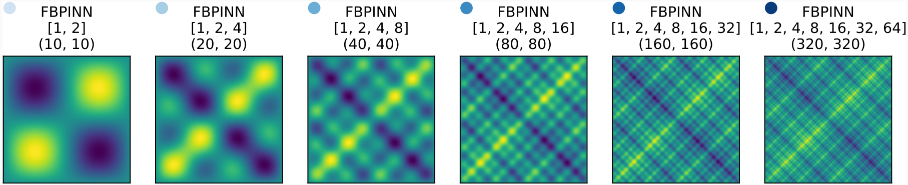


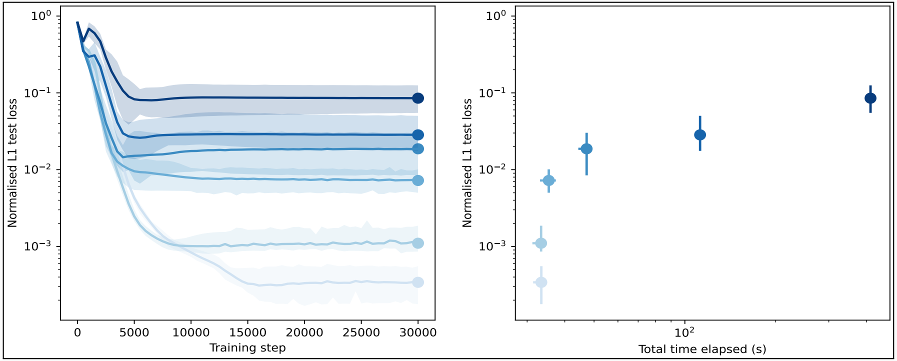
Multi-Level FBPINNs for a Multi-Frequency Problem – Strong Scaling

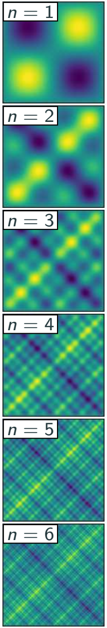




Cf. **Dolean, Heinlein, Mishra, Moseley (2024)**.







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

Domain decomposition for randomized neural networks

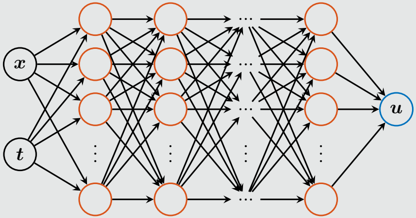
Randomized Neural Networks (RaNNs)

Neural networks

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

$$u(\boldsymbol{x}, \boldsymbol{\theta}) = F_{L+1}^{\boldsymbol{A}} \cdot F_L^{\boldsymbol{W}_L, \boldsymbol{b}_L} \circ \ldots \circ F_1^{\boldsymbol{W}_1, \boldsymbol{b}_1}(\boldsymbol{x}),$$

where $\boldsymbol{A}$ is linear, and the ith hidden layer is nonlinear $F_i^{\boldsymbol{W}_i, \boldsymbol{b}_i}(\boldsymbol{x}) = \sigma(\boldsymbol{W}_i \cdot \boldsymbol{x} + \boldsymbol{b}_i)$.



In order to optimize the loss function

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

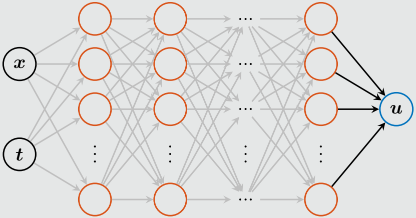
all parameters $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{W}_1, \boldsymbol{b}_1, \ldots, \boldsymbol{W}_L, \boldsymbol{b}_L)$ are **trained**.

Randomized neural networks

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

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

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



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

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

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 **Igelnik 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**

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

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

Enforcement of boundary conditions

We construct u to explicitly satisfy BCs:

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

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

The resulting linear equation system

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

generally does **not have a unique solution**. In fact, $\boldsymbol{H}$ 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**.

Domain Decomposition-Based PIRaNNs

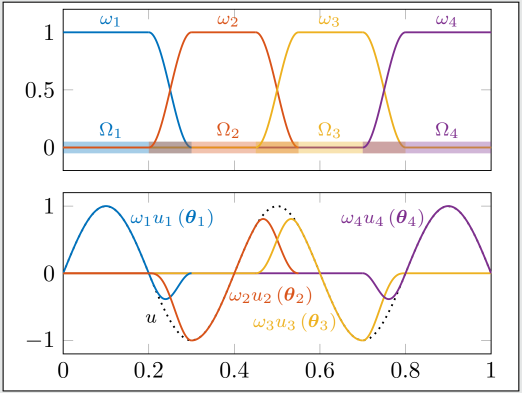
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

$$u(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J) = \sum_{j=1}^{J} \omega_j u_j(\boldsymbol{\theta}_j)$$

and the **loss function**

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \Big(n[\sum_{\boldsymbol{x}_i \in \Omega_j} \omega_j u_j](\boldsymbol{x}_i, \boldsymbol{\theta}_j) - f(\boldsymbol{x}_i) \Big)^2.$$

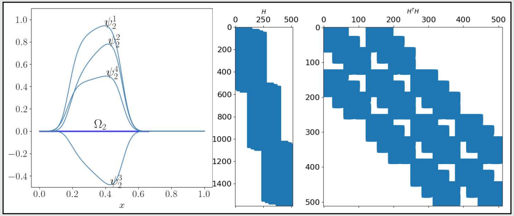


Domain decomposition for RaNNs

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

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

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

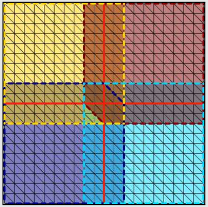


The hidden weights are randomly initialized, the resulting matrices $\boldsymbol{H}$ and $\boldsymbol{H}^\top \boldsymbol{H}$ are block-sparse.

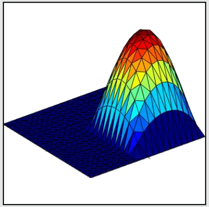
Preconditioning for Domain Decomposition-Based PIRaNNs

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 $\boldsymbol{K} := \boldsymbol{H}^\top \boldsymbol{H}$

$$\boldsymbol{M}_{\text{OS-1}}^{-1}\boldsymbol{K} = \sum\nolimits_{i=1}^{N} \boldsymbol{R}_i^\top \boldsymbol{K}_i^{-1} \boldsymbol{R}_i \boldsymbol{K},$$

where $\boldsymbol{R}_i$ and $\boldsymbol{R}_i^\top$ are restriction and prolongation operators corresponding to Ω_i', and $\boldsymbol{K}_i := \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^\top$.

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

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

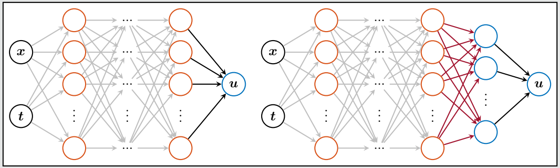
$$\begin{aligned} u_j(\boldsymbol{x}, \boldsymbol{A}_j) &= F_{L+1}^{\boldsymbol{A}} \cdot F_L^{W_L, b_L} \circ \ldots \circ F_1^{W_1, b_1}(\boldsymbol{x}) \\ &= \boldsymbol{A}_j \begin{bmatrix} \Phi_1(\boldsymbol{x}) & \cdots & \Phi_k(\boldsymbol{x}) \end{bmatrix}^\top, \end{aligned}$$

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

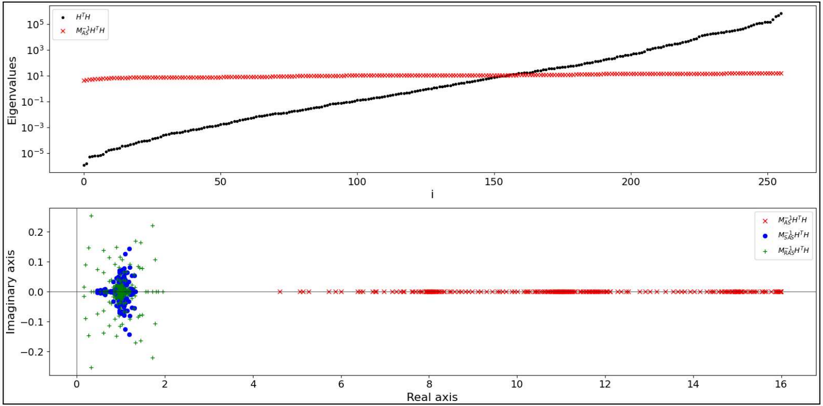
Consider a **reduced SVD** $\boldsymbol{\Phi} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, where the entries of the matrix are $\boldsymbol{\Phi}_{i,l} = \Phi_l(\boldsymbol{x}_i)$. Then, we consider

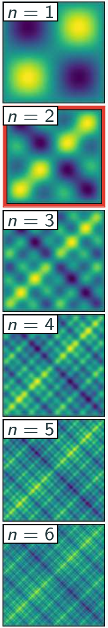
$$\hat{u}_j(\boldsymbol{x}, \boldsymbol{A}_j) = \boldsymbol{A}_j \hat{\boldsymbol{V}}^\top \begin{bmatrix} \Phi_1(\boldsymbol{x}) & \cdots & \Phi_k(\boldsymbol{x}) \end{bmatrix}^\top,$$

where $\hat{\boldsymbol{V}}^\top$ 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 $=256$; $N=1600$; $\boldsymbol{\theta}^0\in\mathcal{U}(-1,1)$; stop.: $\|\boldsymbol{M}^{-1}\boldsymbol{r}^k\|_{L^2}/\|\boldsymbol{M}^{-1}\boldsymbol{r}^0\|_{L^2}\leq 10^{-5}$

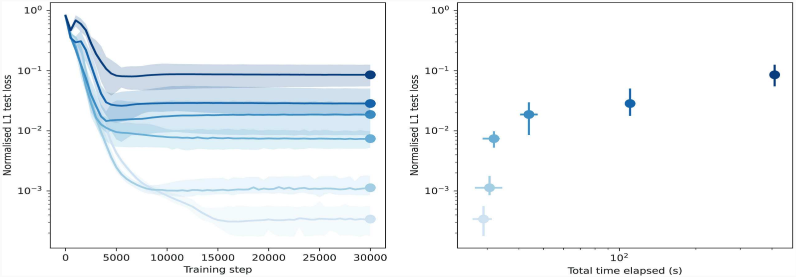
Results for the Multi-Frequency Problem (n=2) – Effect of the SVD

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

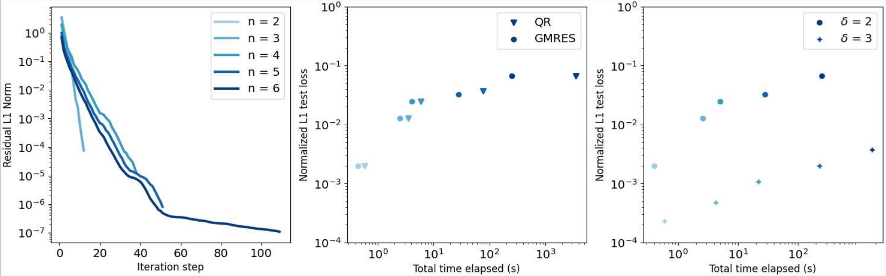
τ	DoF	M^{-1}	σ_{min}	σ_{max}	iter	e_{L^2}
10^{-4}	512	I	10^{-10}	10^{6}	> 2000	$3.72 \cdot 10^{-2}$
		M_{AS}^{-1}	10^{-6}	10^{6}	27	$5.46 \cdot 10^{-5}$
		M_{SAS}^{-1}	10^{-7}	10^{5}	30	$5.49 \cdot 10^{-5}$
10^{-3}	436	I	10^{-8}	10^{5}	> 2000	$3.75 \cdot 10^{-2}$
		M_{AS}^{-1}	10^{-5}	10^{5}	16	$1.28 \cdot 10^{-4}$
		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; $N = 1600$; $\boldsymbol{\theta}^0 \in \mathcal{U}(-1,1)$; stop.: $\|\boldsymbol{M}^{-1}\boldsymbol{r}^k\|_{L^2} / \|\boldsymbol{M}^{-1}\boldsymbol{r}^0\|_{L^2} \leq 10^{-5}$

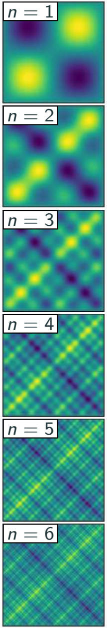
Results for the Multi-Frequency Problem



Multi-level FBPINNs; cf. **Dolean, Heinlein, Mishra, Moseley (2024)**

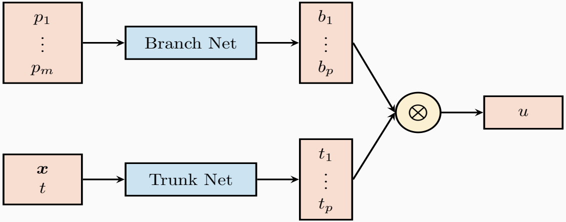


DD-PIRaNNs; cf. **Shang, Heinlein, Mishra, Wang (2025)**



Domain decomposition-based physics-informed deep operator networks

Neural operators learn operators between function spaces using neural networks. Here, we learn the **solution operator** of a initial-boundary value problem parametrized with $p_1, \ldots, p_m$ 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

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

Physics-informed DeepONets

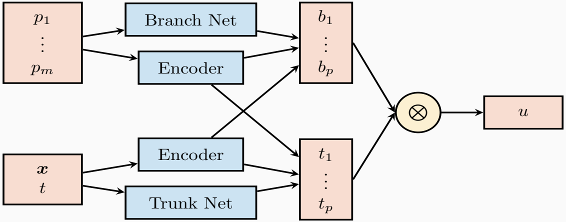
DeepONets are **compatible with the PINN approach** but **physics-informed DeepONets (PI-DeepONets)** are challenging to train.

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 $p_1, \ldots, p_m$ 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

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

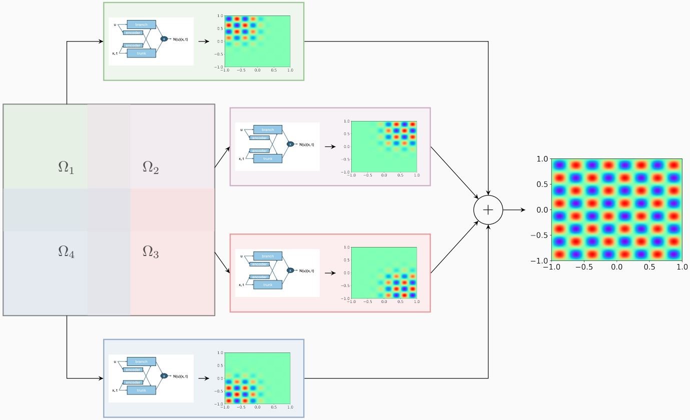
Physics-informed DeepONets

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

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

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

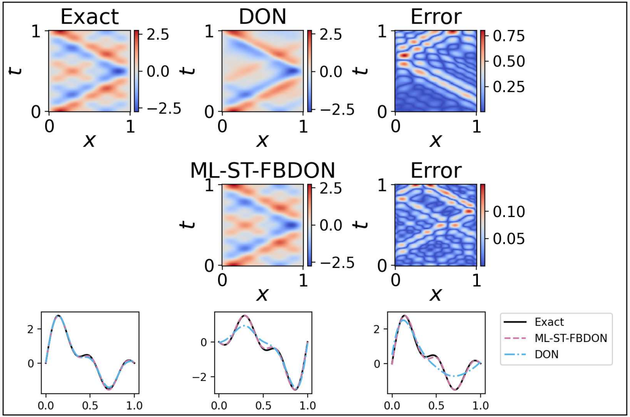
Solution: $s(x,t) = \sum_{n=1}^{5} b_n \sin(n\pi x)\cos(n\pi\sqrt{2}t)$

Parametrization

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

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

Training on 1 000 random configurations.

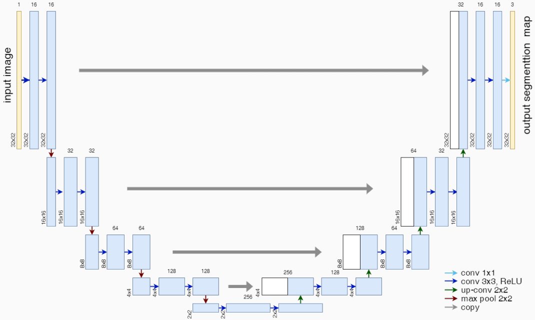


Mean rel. l_2 error on 100 config.	
DeepONet	0.30 ± 0.11
ML-ST-FBDON ([1, 4, 8, 16] subd.)	0.05 ± 0.03
ML-FBDON ([1, 4, 8, 16] subd.)	0.08 ± 0.04

$\rightarrow$ 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

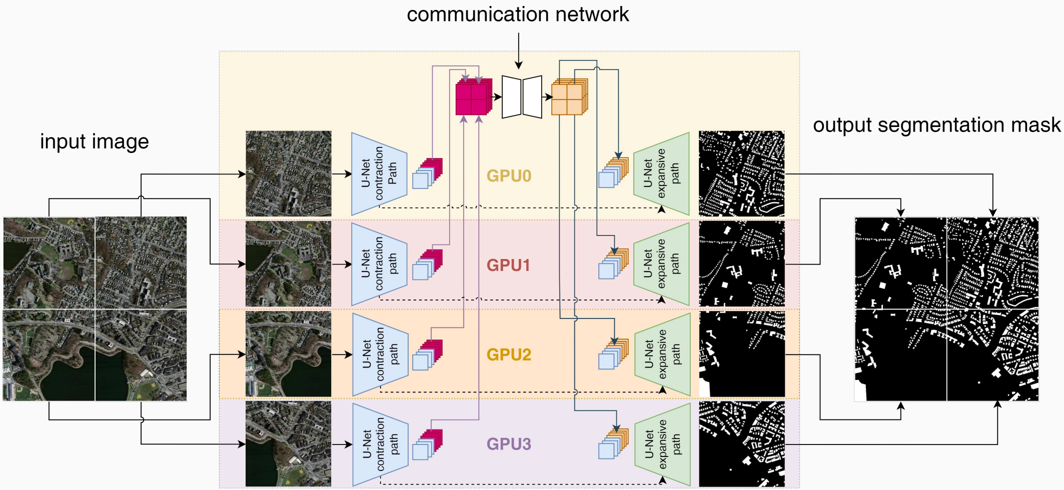


- As an example for a **convolutional neural network (CNN)**, we employ the **U-Net architecture** introduced in **Ronneberger, Fischer, and Brox (2015)**.
- The U-Net yields **state-of-the-art accuracy in semantic image segmentation** and other **image-to-image tasks**.

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

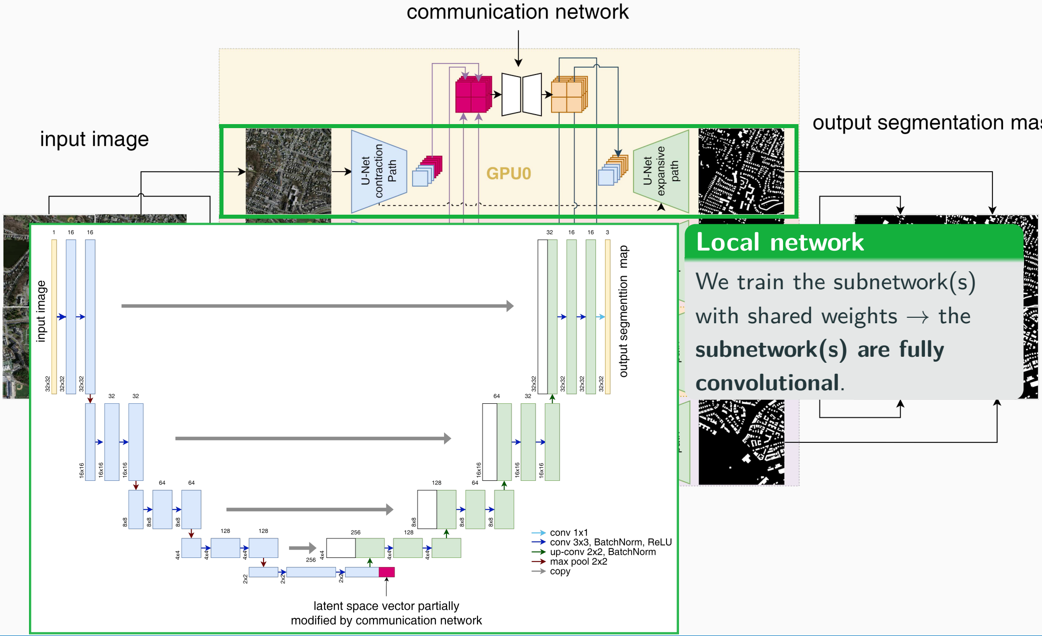
name	size	# channels		mem. feature maps		mem. weights	
		input	output	# of values	MB	# of values	MB
input block	1 024	3	64	268 M	**1 024.0**	38 848	**0.148**
encoder block 1	512	64	128	167 M	**704.0**	221 696	**0.846**
encoder block 2	256	128	256	84 M	**352.0**	885 760	**3.379**
encoder block 3	128	256	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**

Decomposing the U-Net



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

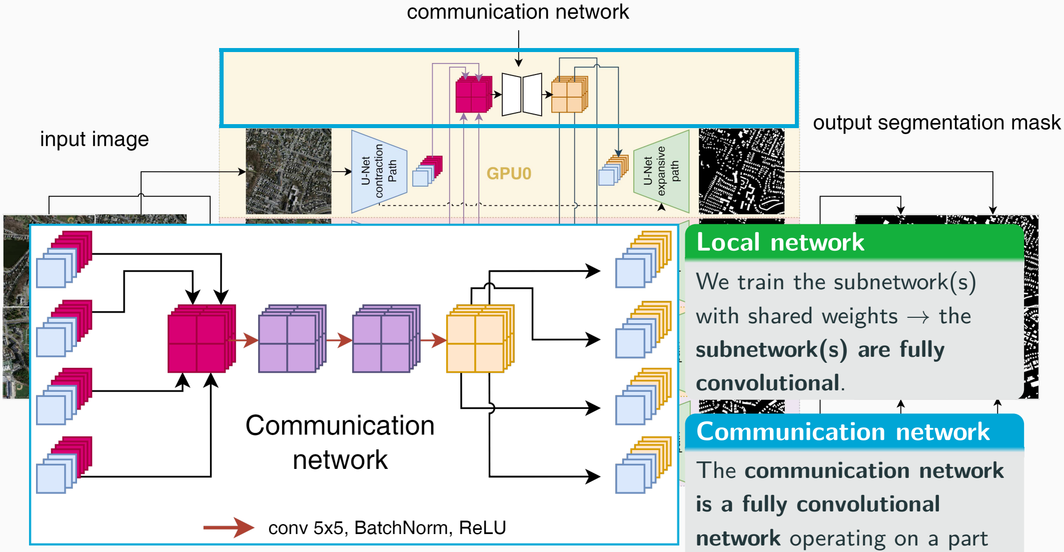
Decomposing the U-Net



Local network

We train the subnetwork(s) with shared weights → the **subnetwork(s) are fully convolutional**.

Decomposing the U-Net



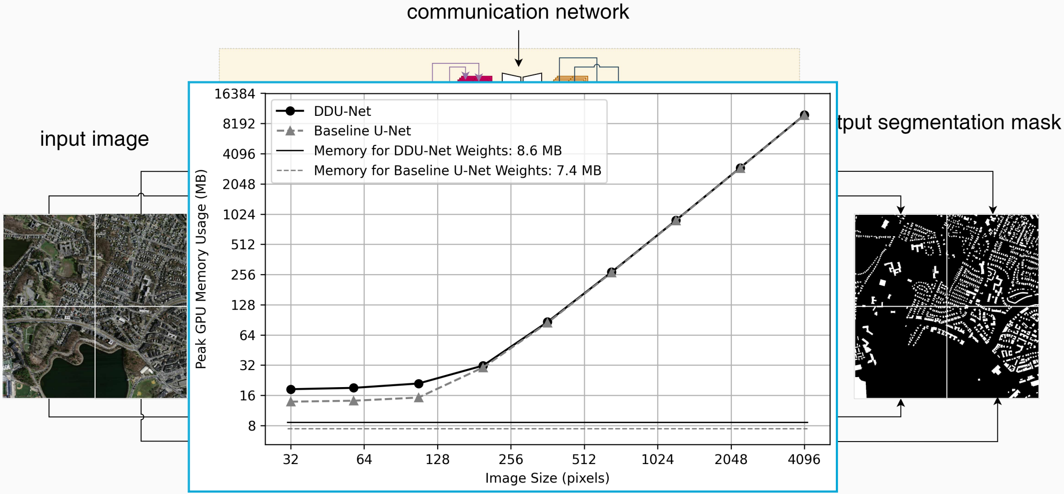
Local network

We train the subnetwork(s) with shared weights → the **subnetwork(s) are fully convolutional**.

Communication network

The **communication network is a fully convolutional network** operating on a part of the coarse feature maps.

Decomposing the U-Net

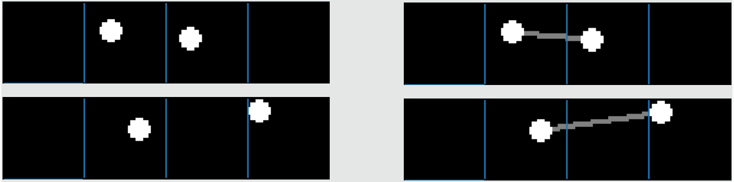


- Distribution of feature maps results in **significant reduction of memory usage on a single GPU**
- **Moderate additional memory usage** due to the **communication network**

Results – Synthetic Data Set

Task: Connect two dots via a line segment

Input | Target (segmentation mask)

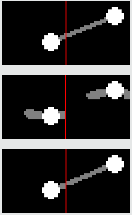


Result: Communication

True mask

Pred. (no comm.)

Pred. (comm.)



Training & testing on # of subimages (32 × 32 pixels) vs. # feature maps communicated

# of subimages	0	1	2	4	8	16	32	Baseline
2	0.14	0.55	0.57	0.87	0.91	0.92	0.93	0.95
3	0.14	0.44	0.66	0.77	0.84	0.90	0.92	0.81
4	0.14	0.58	0.61	0.77	0.84	0.88	0.90	0.74
6	0.14	0.23	0.54	0.67	0.72	0.82	0.87	0.68
8	0.14	0.18	0.40	0.50	0.63	0.70	0.69	0.39
16	0.08	0.18	0.18	0.18	0.18	0.18	0.20	0.19

Testing on 6 subimages

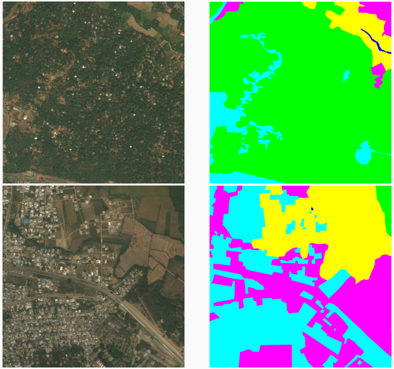
Training on # subimages (32 × 32 pixels) vs. # feature maps communicated

# subimages	0	1	2	4	8	16	32	Baseline
2	0.13	0.20	0.20	0.29	0.28	0.30	0.40	0.53
3	0.14	0.39	0.65	0.75	0.79	0.88	0.92	0.76
4	0.14	0.59	0.55	0.77	0.78	0.83	0.73	0.75
6	0.14	0.23	0.54	0.67	0.72	0.82	0.87	0.68
8	0.13	0.18	0.39	0.49	0.64	0.71	0.70	0.42
16	0.13	0.14	0.14	0.16	0.18	0.16	0.26	0.26

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 %

Input Target

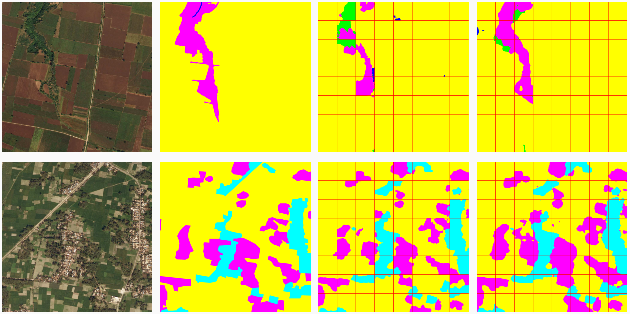


Avoiding overfitting

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

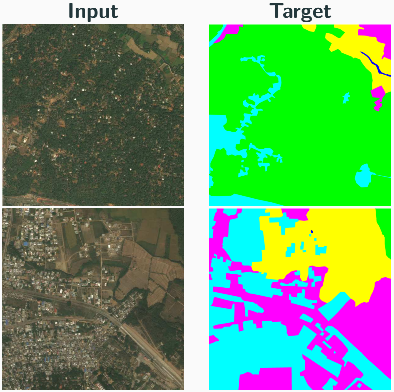
- apply **batch normalization**, use **random dropout** layers and **data augmentation**, and
- **initialize the encoder** using the **ResNet-18** (**He, Zhang, Ren, and Sun (2016)**)

image true mask without comm. with comm.



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

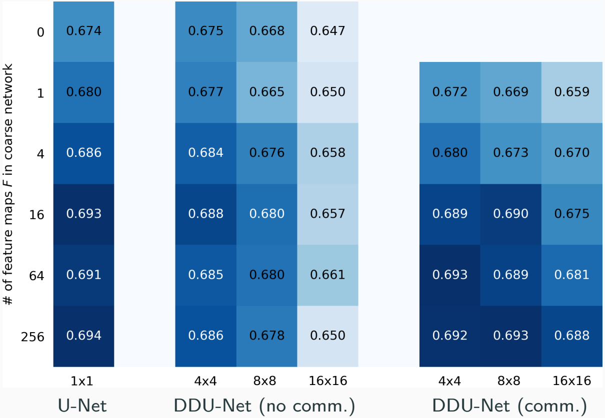
class	pixel count	proportion
urban	642.4M	9.35 %
agriculture	3898.0M	56.76 %
rangeland	701.1M	10.21 %
forest	944.4M	13.75 %
water	256.9M	3.74 %
barren	421.8M	6.14 %
unknown	3.0M	0.04 %



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)**)



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 Tabbeart (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

- **Schwarz domain decomposition architectures improve the scalability of PINNs** to **large domains / high frequencies**, **keeping the complexity of the local networks low**.

Extensions to RaNNs and DeepONets

- **RaNNs** reduce computational cost but also face **ill-conditioning**, mitigated by **Schwarz preconditioning** and **SVD**.
- **DeepONets** provide **efficient predictions** for **parametrized problems**. **Domain decomposition improves scalability and performance**.

Domain decomposition for Image Segmentation with CNNs

- **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

Scientific Machine Learning