Domain Decomposition for Physics-Informed Neural Networks

Linear and Nonlinear Function Approximation and Operator Learning

Alexander Heinlein[1]

29th International Conference on Domain Decomposition Methods, July 23-27, 2025

[1]Delft University of Technology

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)

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

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

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

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

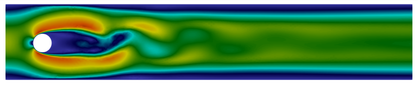
See also **Dissanayake and Phan-Thien (1994); Lagaris et al. (1998)**.

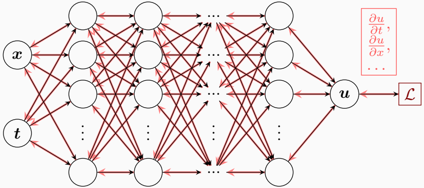
Advantages

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

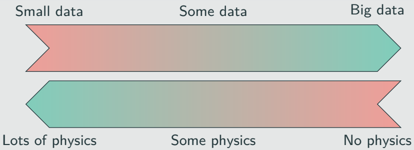
Drawbacks

- **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 $\mathcal{L}_{\text{data}}$

Scaling of PINNs for a Simple ODE Problem

Solve

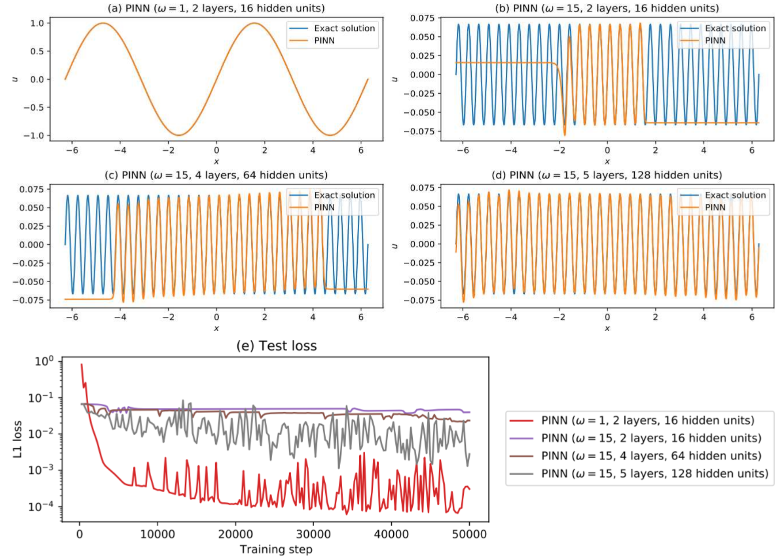
$$u' = \cos(\omega \boldsymbol{x}),$$
$$u(0) = 0,$$

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

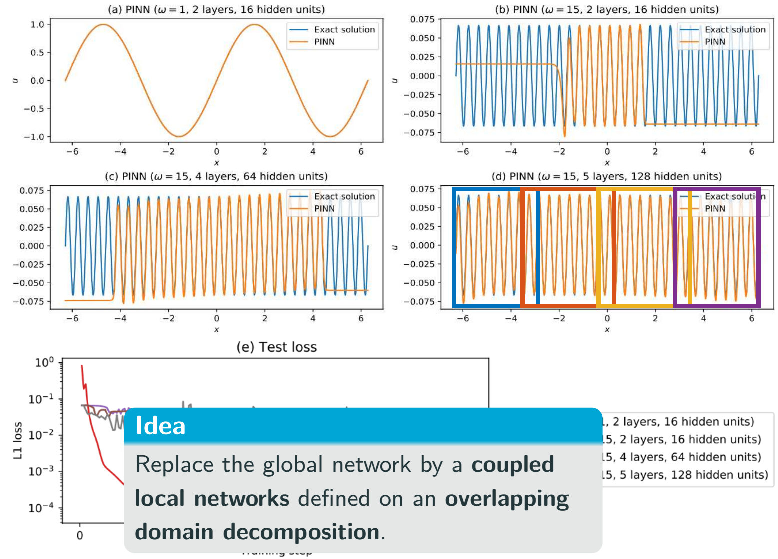
$$
\begin{aligned}
u' &= \cos(\omega \boldsymbol{x}), \\
u(0) &= 0,
\end{aligned}
$$

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)**



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

Idea

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

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 (acc. 2025 / 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

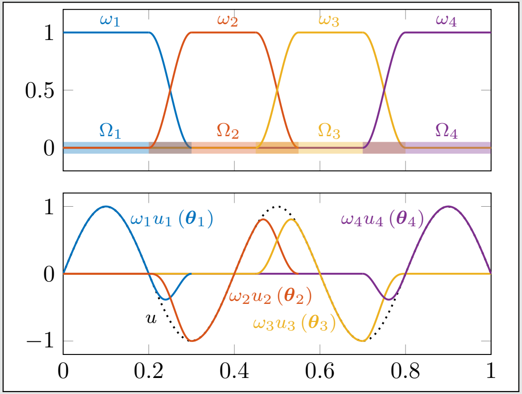
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

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

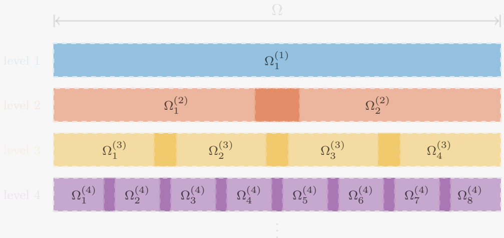
and the **loss function**

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



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**

$$u\left(\theta_1^{(1)}, \ldots, \theta_{J^{(L)}}^{(L)}\right) = \sum_{l=1}^{L} \sum_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)} \left(\theta_j^{(l)}\right)$$

and the **loss function**

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

Finite Basis Physics-Informed Neural Networks (FBPINNs)

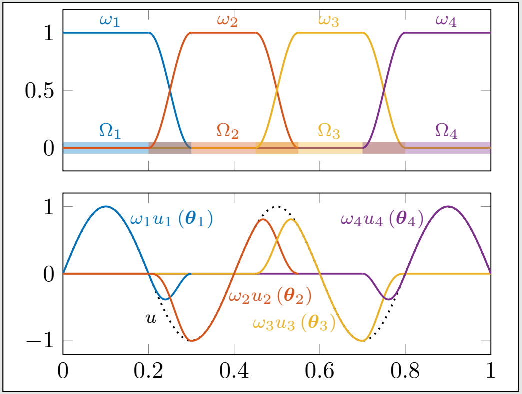
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

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

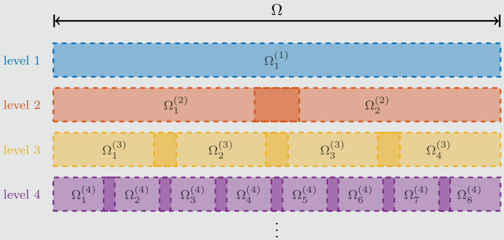
and the **loss function**

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



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**

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

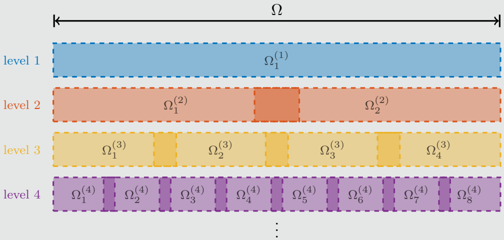
and the **loss function**

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

Finite Basis Physics-Informed Neural Networks (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**

$$u\left(\theta_1^{(1)}, \ldots, \theta_{J^{(L)}}^{(L)}\right) = \sum\nolimits_{l=1}^{L} \sum\nolimits_{j=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}\left(\theta_j^{(l)}\right)$$

and the **loss function**

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

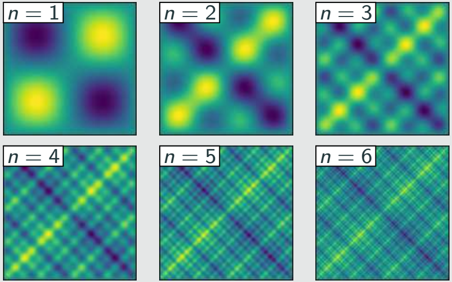
Multi-Frequency Problem

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

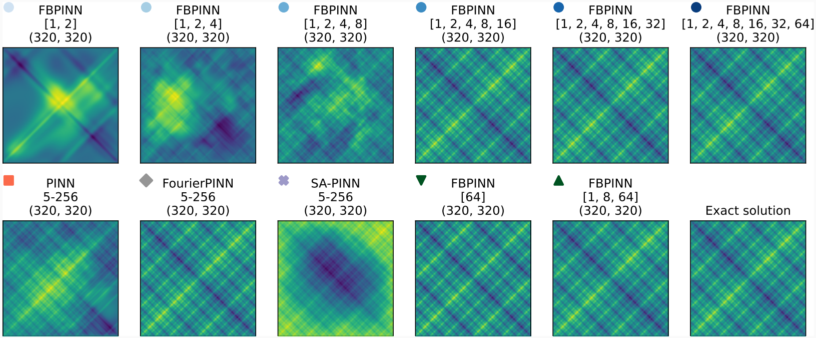
$$\begin{aligned} -\Delta u &= 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

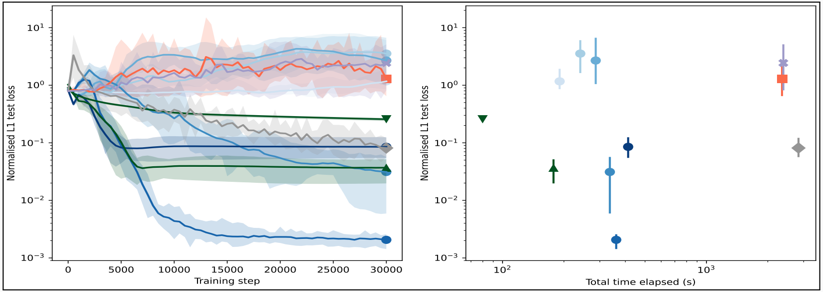
with $\omega_i = 2^i$.

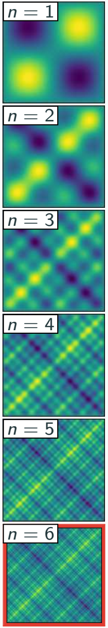
For increasing values of n, we obtain the **analytical solutions**:



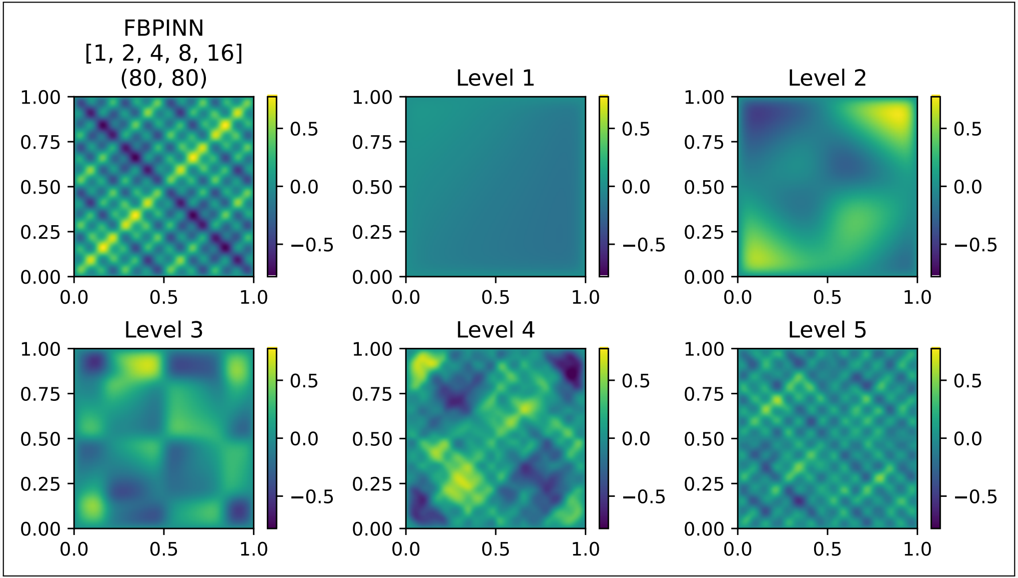
Multi-Level FBPINNs for a Multi-Frequency Problem – Strong Scaling







Multi-Frequency Problem – What the FBPINN Learns



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

Domain decomposition for randomized neural networks

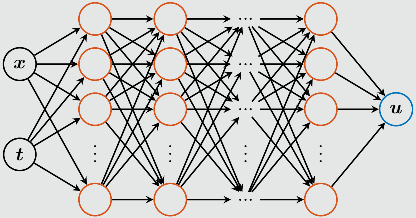
Physics-Informed Randomized Neural Networks (PIRaNNs)

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

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

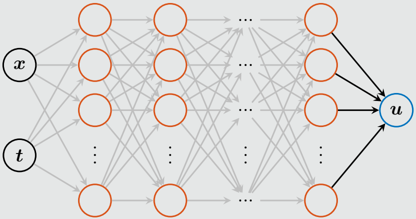
all parameters $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{W}_1, \boldsymbol{b}_1, \ldots, \boldsymbol{W}_L, \boldsymbol{b}_L)$ are **trained**.

Randomized neural networks

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

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

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**.

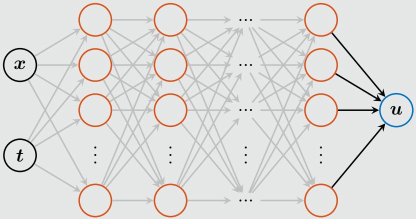
Physics-Informed Randomized Neural Networks (PIRaNNs)

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

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

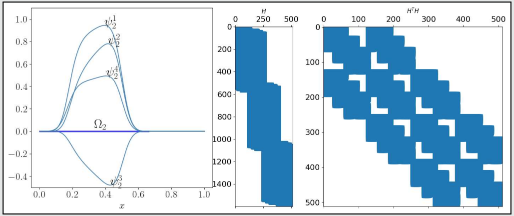
This can **simplify the training process**.

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

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

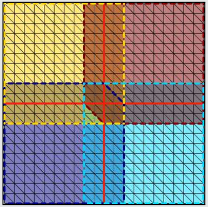
for $i = 1, \ldots, N_{\text{PDE}}$; the boundary condtions are incorporated directly into the u_j.

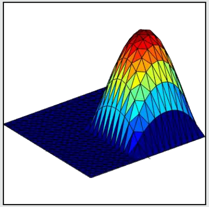


The hidden weights are randomly initialized, the resulting matrices $\boldsymbol{H}$ and $\boldsymbol{H}^\top \boldsymbol{H}$ are block-sparse.

Preconditioning for Domain Decomposition-Based PIRaNNs

One-level Schwarz preconditioner





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

$$M_{\text{OS-1}}^{-1} \boldsymbol{K} = \sum_{i=1}^{N} \boldsymbol{R}_i^\top \boldsymbol{K}_i^{-1} \boldsymbol{R}_i \boldsymbol{K},$$

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

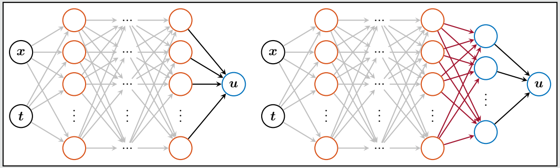
$$u_j(\boldsymbol{x}, \boldsymbol{A}_j) = F_{L+1}^{\boldsymbol{A}} \cdot F_L^{W_L, b_L} \circ \ldots \circ F_1^{W_1, b_1}(\boldsymbol{x})$$
$$= \boldsymbol{A}_j \begin{bmatrix} \Phi_1(\boldsymbol{x}) & \cdots & \Phi_k(\boldsymbol{x}) \end{bmatrix}^\top,$$

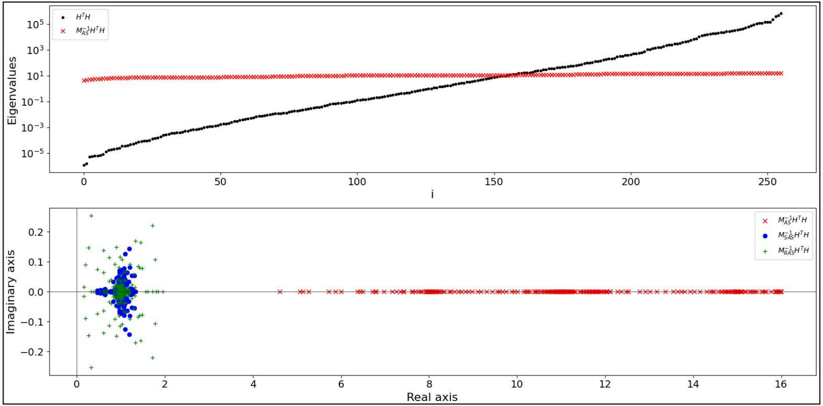
where k is the width of the last hidden layer and the Φ_l are the randomized basis functions.

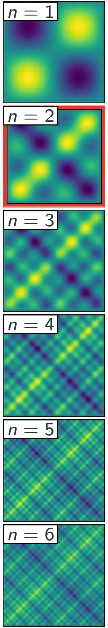
Consider a **reduced SVD** $\boldsymbol{\Phi} = \boldsymbol{U\Sigma V}^\top$, where the entries of the matrix are $\boldsymbol{\Phi}_{i,l} = \Phi_l(\boldsymbol{x}_i)$. Then, we consider

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

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



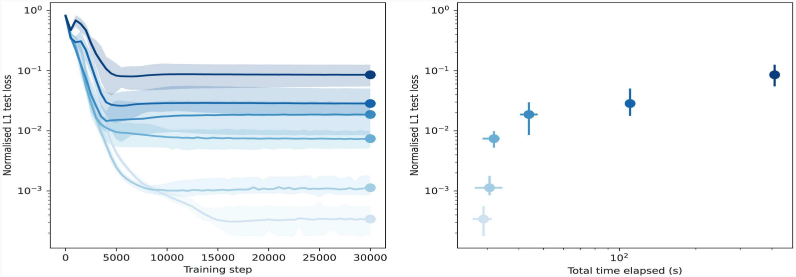




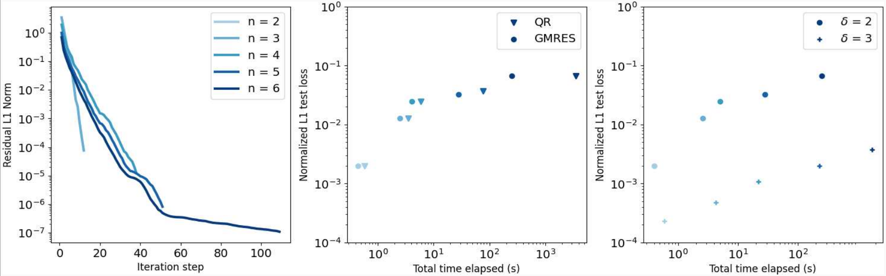
	$M^{-1} = I$		$M^{-1} = M_{AS}^{-1}$		$M^{-1} = M_{RAS}^{-1}$		$M^{-1} = M_{SAS}^{-1}$	
	iter	e_{L^2}	iter	e_{L^2}	iter	e_{L^2}	iter	e_{L^2}
CG	$>$ 2000	$1.95\cdot10^{-2}$	8	$5.03\cdot10^{-3}$	—	—	—	—
CGS	$>$ 2000	$2.63\cdot10^{-2}$	4	$5.04\cdot10^{-3}$	24	$5.03\cdot10^{-3}$	6	$5.04\cdot10^{-3}$
BICG	$>$ 2000	$1.03\cdot10^{-2}$	8	$5.08\cdot10^{-3}$	32	$5.05\cdot10^{-3}$	11	$5.09\cdot10^{-3}$
GMRES	$>$ 2000	$8.68\cdot10^{-2}$	13	$5.07\cdot10^{-3}$	31	$5.06\cdot10^{-3}$	11	$5.08\cdot10^{-3}$

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

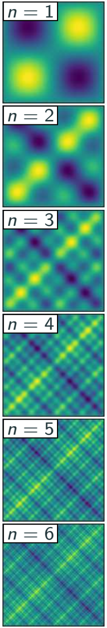
Results for the Multi-Frequency Problem



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



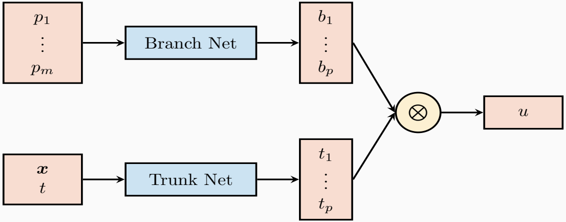
DD-PIRaNNs; cf. **Shang, Heinlein, Mishra, Wang (2025)**



Domain decomposition-based physics-informed deep operator networks

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)**.



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

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

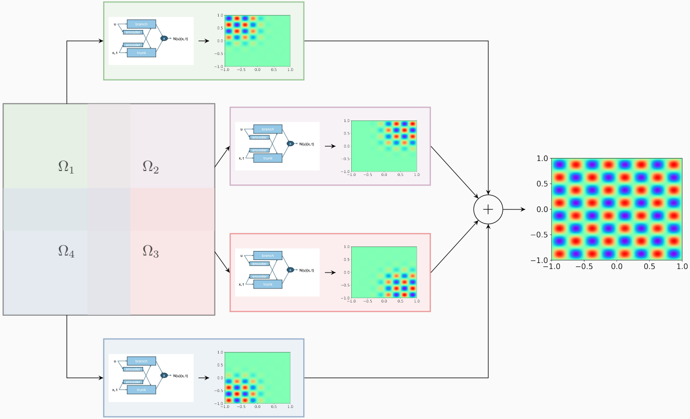
Physics-informed DeepONets

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)**

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

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

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

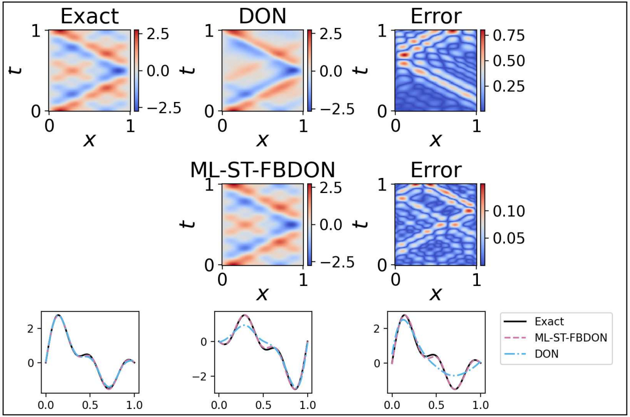
Solution: $s(x,t) = \sum_{n=1}^{5} b_n \sin(n\pi x)\cos\left(n\pi\sqrt{2}t\right)$

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

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

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

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.

Thank you for your attention!

Topical Activity Group

Scientific Machine Learning