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

Alexander Heinlein[1]

Workshop on Scientific Computing and Learning, University of Macau, China, August 17-18, 2023

[1]Delft University of Technology

Based on joint work with Victorita Dolean (University of Strathclyde & University Côte d'Azur) and Sid Mishra and Ben Moseley (ETH Zürich)

Artificial Neural Networks for Solving Ordinary and Partial Differential Equations

Isaac Elias Lagaris, Aristidis Likas, *Member, IEEE,* and Dimitrios I. Fotiadis

Published in IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 9, NO. 5, 1998.

Approach

Solve a general differential equation subject to boundary conditions

$$G(\boldsymbol{x}, \Psi(\boldsymbol{x}), \nabla\Psi(\boldsymbol{x}), \nabla^2\Psi(\boldsymbol{x})) = 0 \quad \text{in } \Omega$$

by solving an **optimization problem**

$$\min_{\boldsymbol{\theta}} \sum_{\boldsymbol{x}_i} G(\boldsymbol{x}_i, \Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}), \nabla\Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}), \nabla^2\Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}))^2$$

where $\Psi_t\left(\boldsymbol{x}, \boldsymbol{\theta}\right)$ is a **trial function**, $\boldsymbol{x}_i$ **sampling points inside the domain** Ω and $\boldsymbol{\theta}$ are **adjustable parameters**.

Construction of the trial functions

The trial functions **explicitly satisfy the boundary conditions**:

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

- N is a **feedforward neural network** with **trainable parameters** $\boldsymbol{\theta}$ and input $x \in \mathbb{R}^n$
- A and F are **fixed functions**, chosen s.t.:
  - **A satisfies the boundary conditions**
  - **F does not contribute to the boundary conditions**

Neural Networks for Solving Differential Equations

Approach

Solve a general differential equation subject to boundary conditions

$$G(\boldsymbol{x}, \Psi(\boldsymbol{x}), \nabla\Psi(\boldsymbol{x}), \nabla^2\Psi(\boldsymbol{x})) = 0 \quad \text{in } \Omega$$

by solving an **optimization problem**

$$\min_{\boldsymbol{\theta}} \sum_{\boldsymbol{x}_i} G(\boldsymbol{x}_i, \Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}), \nabla\Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}), \nabla^2\Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}))^2$$

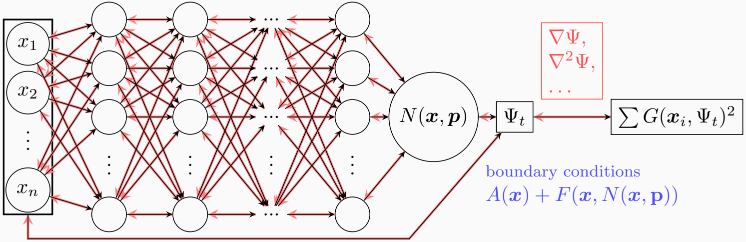
where $\Psi_t\left(\boldsymbol{x}, \boldsymbol{\theta}\right)$ is a **trial function**, $\boldsymbol{x}_i$ **sampling points inside the domain** Ω and $\boldsymbol{\theta}$ are **adjustable parameters**.

Construction of the trial functions

The trial functions **explicitly satisfy the boundary conditions**:

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

- N is a **feedforward neural network** with **trainable parameters** $\boldsymbol{\theta}$ and input $x \in \mathbb{R}^n$
- A and F are **fixed functions**, chosen s.t.:
  - **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**

$$\mathcal{N}[u](\boldsymbol{x}, \boldsymbol{t}) = f(\boldsymbol{x}, \boldsymbol{t}), \qquad (\boldsymbol{x}, \boldsymbol{t}) \in [0, T] \times \Omega \subset \mathbb{R}^d.$$

It is based on the approach by Lagaris et al. (1998). The main novelty of PINNs is the use of a **hybrid loss function**:

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

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

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

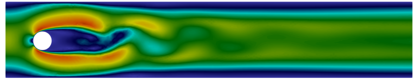
$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} \left(\mathcal{N}[u](\boldsymbol{x}_i, \boldsymbol{t}) - f(\boldsymbol{x}_i, \boldsymbol{t}_i) \right)^2.$$

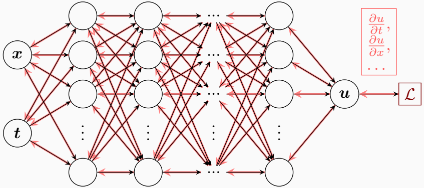
Advantages

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

Available Theoretical Results for PINNs – An Example

Mishra and Molinaro. *Estimates on the generalisation error of PINNs*, **2022**

Estimate of the generalization error

The generalization error (or total error) satisfies

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

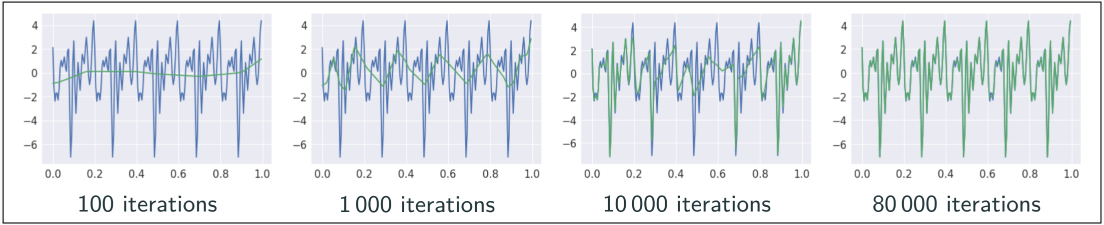
where

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

Rule of thumb:

“As long as the PINN is trained well, it also generalizes well”

- **Spectral bias: neural networks prioritize learning lower frequency functions** first irrespective of their amplitude



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

- Solving solutions on **large domains and/or with multiscale features** potentially requires **very large neural networks**.
- Training may **not sufficiently reduce the loss** or take **large numbers of iterations**.
- Significant **increase on the computational work**

Convergence analysis of PINNs via the **neural tangent kernel**: **Wang, Yu, Perdikaris,** *When and why PINNs fail to train: A neural tangent kernel perspective*, **JCP (2022)**

Solve

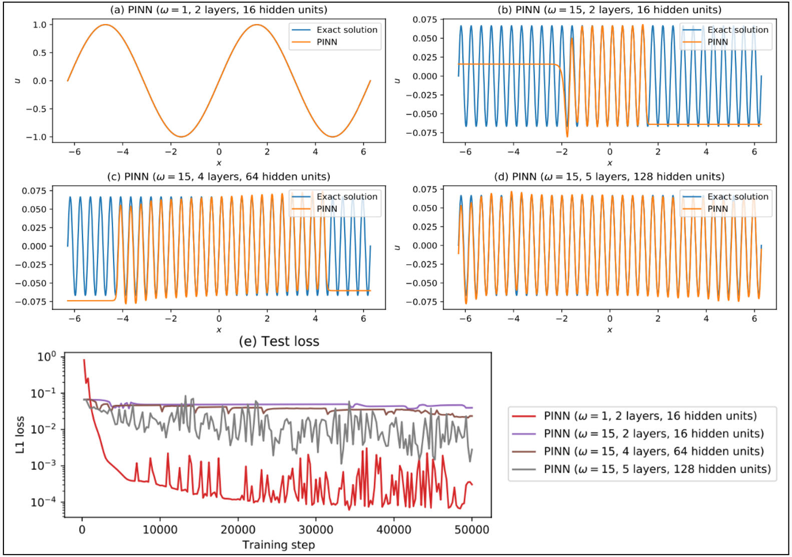
$$\begin{aligned} u' &= \cos(\omega \boldsymbol{x}), \\ u(0) &= 0, \end{aligned}$$

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

Scaling issues

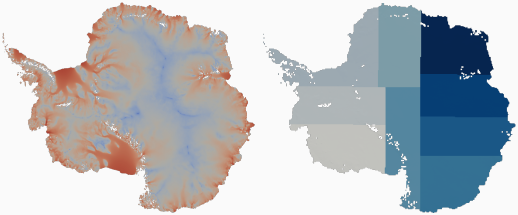
- Large computational domains
- Small frequencies

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



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

Domain Decomposition Methods



Images based on Heinlein, Perego, Rajamanickam (2022)

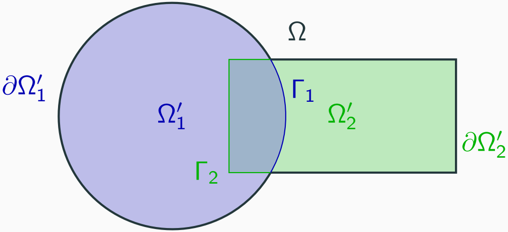
Historical remarks: The **alternating Schwarz method** is the earliest **domain decomposition method (DDM)**, which has been invented by **H. A. Schwarz** and published in **1870**:

- 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 overview**:

- **Machine Learning for adaptive BDDC, FETI–DP, and AGDSW**: Heinlein, Klawonn, Lanser, Weber (2019, 2020, 2021, 2021, 2021, 2022); Klawonn, Lanser, Weber (preprint 2022)
- **Domain decomposition for CNNs**: Gu, Zhang, Liu, Cai (2022); Lee, Park, Lee (2022); Klawonn, Lanser, Weber (arXiv 2023)
- **D3M**: Li, Tang, Wu, and Liao (2019)
- **DeepDDM**: Li, Xiang, Xu (2020); Mercier, Gratton, Boudier (arXiv 2021); Li, Wang, Cui, Xiang, Xu (2023); Sun, Xu, Yi (arXiv 2022, arXiv 2023)
- **FBPINNs**: Moseley, Markham, and Nissen-Meyer (2023); Dolean, Heinlein, Mishra, Moseley (accepted 2023, submitted 2023/arXiv:2306.05486)
- **Schwarz Domain Decomposition Algorithm for PINNs**: Kim, Yang (2022, arXiv 2022)
- **cPINNs**: Jagtap, Kharazmi, Karniadakis (2020)
- **XPINNs**: Jagtap, Karniadakis (2020)

An overview of the state-of-the-art in early 2021:

A. Heinlein, A. Klawonn, M. Lanser, J. Weber.
Combining machine learning and domain decomposition methods for the solution of partial differential equations — A review.
GAMM-Mitteilungen. 2021.

Finite Basis Physics-Informed Neural Networks (FBPINNs)

In the **finite basis physics informed neural network (FBPINNs) method** introduced in **Moseley, Markham, and Nissen-Meyer (2023)**, we solve the boundary value problem

$$\begin{aligned} \mathcal{N}[u](\mathbf{x}) &= f(\mathbf{x}), & \mathbf{x} \in \Omega \subset \mathbb{R}^d, \\ \mathcal{B}_k[u](\mathbf{x}) &= g_k(\mathbf{x}), & \mathbf{x} \in \Gamma_k \subset \partial\Omega. \end{aligned}$$

using the **PINN** approach and **hard enforcement of the boundary conditions**, similar to **Lagaris et al. (1998)**.

FBPINNs use the **network architecture**

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

and the **loss function**

$$\mathcal{L}(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J) = \frac{1}{N} \sum_{i=1}^{N} \Big(\mathcal{N}[\mathcal{C} \sum_{\mathbf{x}_i \in \Omega_j} \omega_j u_j](\mathbf{x}_i, \boldsymbol{\theta}_j) - f(\mathbf{x}_i) \Big)^2.$$

- **Overlapping DD**: $\Omega = \bigcup_{l=1}^{J} \Omega_j$
- **Window functions** ω_j with $\text{supp}(\omega_j) \subset \Omega_j$ and $\sum_{j=1}^{J} \omega_j \equiv 1$ on Ω

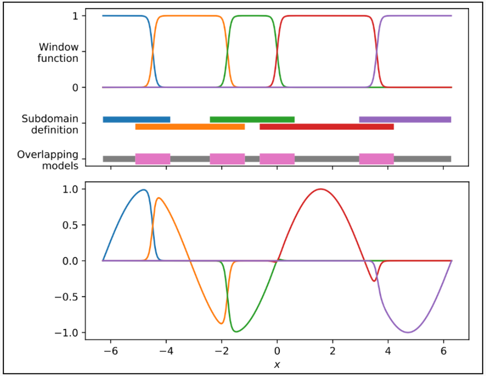
Hard enforcement of boundary conditions

Loss function

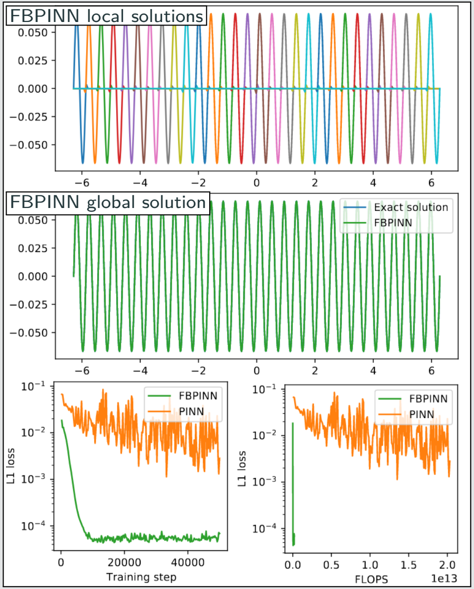
$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left(\mathcal{N}[\mathcal{C} u](\mathbf{x}_i, \boldsymbol{\theta}) - f(\mathbf{x}_i) \right)^2,$$

with constraining operator $\mathcal{C}$, which **explicitly enforces the boundary conditions**.

$\rightarrow$ Often **improves training performance**



PINN Vs FBPINN (Moseley et al. (2023))



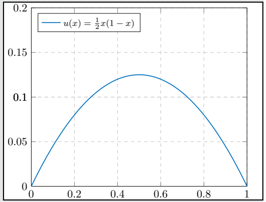
Scalability of FBPINNs

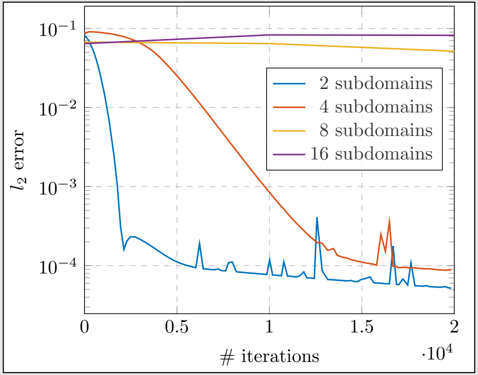
Consider the **simple boundary value problem**

$$-u'' = 1 \quad \text{in } [0,1],$$
$$u(0) = u(1) = 0,$$

which has the **solution**

$$u(x) = 1/2\, x(1-x).$$





Two-Level FBPINN Algorithm

Coarse correction and spectral bias

Questions:

- Scalability requires **global transport of information**. This can be done via **coarse global problem**.
- What does this mean in the **context of network training**?

Idea:

→ Learn **low frequencies** using a **small global network**, train **high frequencies** using **local networks**.

Two-level FBPINN network architecture:

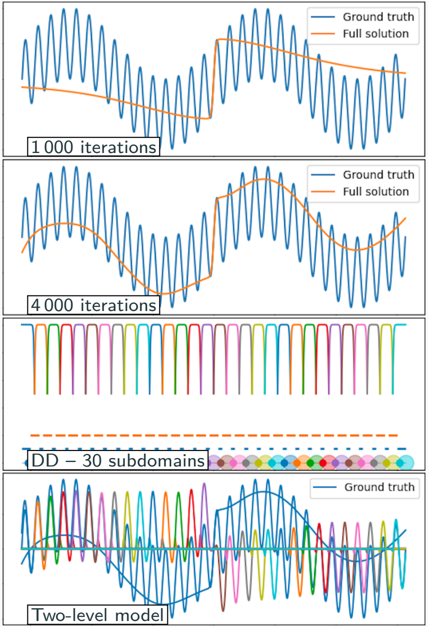
$$u(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J) = \mathcal{C}\left(u_0\left(\boldsymbol{\theta}_0\right) + \sum\nolimits_{j=1}^{J} \omega_j u_j\left(\boldsymbol{\theta}_j\right)\right)$$

Consider a **simple model problem** with **two frequencies**

$$\begin{cases} u' &= \omega_1 \cos(\omega_1 \boldsymbol{x}) + \omega_2 \cos(\omega_2 \boldsymbol{x}) \\ u(0) &= 0. \end{cases}$$

with $\omega_1 = 1$, $\omega_2 = 15$.

Cf. Dolean, Heinlein, Mishra, Moseley (accepted 2023).

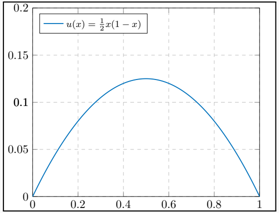


Consider, again, the **simple boundary value problem**

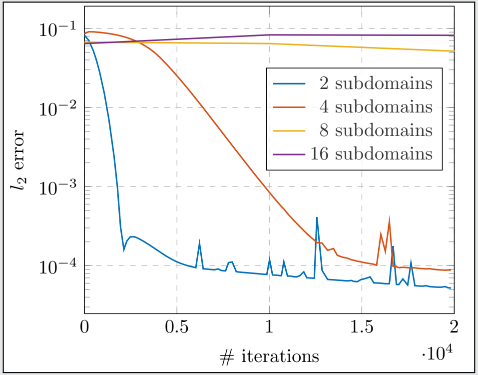
$$-u'' = 1 \quad \text{in } [0,1],$$
$$u(0) = u(1) = 0,$$

which has the **solution**

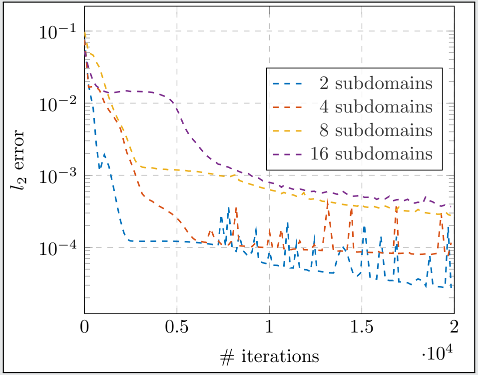
$$u(x) = 1/2\,x(1-x).$$



One-Level FBPINNs



Two-Level FBPINNs

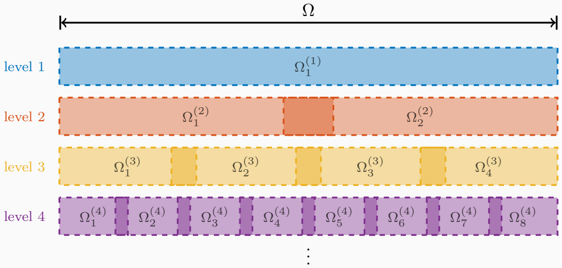


We introduce a **hierarchy of L overlapping domain decompositions**

$$\Omega = \bigcup_{j=1}^{J^{(l)}} \Omega_j^{(l)}$$

and corresponding window functions $\omega_j^{(l)}$ with

$\operatorname{supp}\left(\omega_j^{(l)}\right) \subset \Omega_j^{(l)}$ and $\sum_{j=1}^{J^{(l)}} \omega_j^{(l)} \equiv 1$ on Ω.



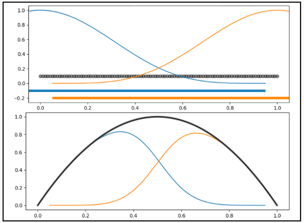
This yields the **L-level FBPINN algorithm**:

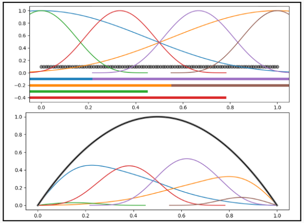
L-level network architecture

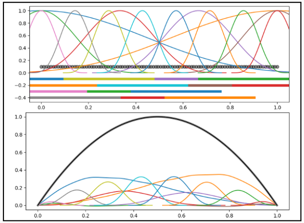
$$u\left(\theta_1^{(1)}, \ldots, \theta_{J^{(L)}}^{(L)}\right) = \mathcal{C}\left(\sum_{l=1}^{L} \sum_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}\left(\theta_j^{(l)}\right)\right)$$

Loss function

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





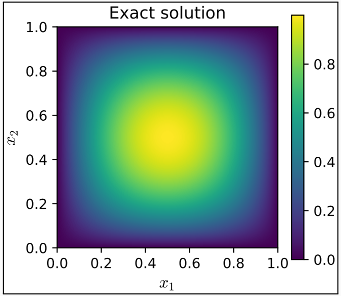


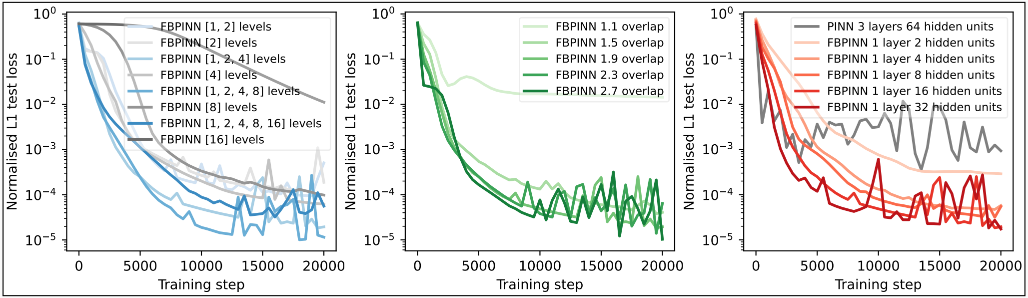
Let us consider the **simple two-dimensional boundary value problem**

$$\begin{aligned} -\Delta u &= 32(\boldsymbol{x}(1-\boldsymbol{x}) + \boldsymbol{y}(1-\boldsymbol{y})) && \text{in } \Omega = [0,1]^2, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

which has the **solution**

$$u(\boldsymbol{x},\boldsymbol{y}) = 16\,(\boldsymbol{x}(1-\boldsymbol{x})\boldsymbol{y}(1-\boldsymbol{y}))\,.$$





Cf. Dolean, Heinlein, Mishra, Moseley (submitted 2023/arXiv:2306.05486).

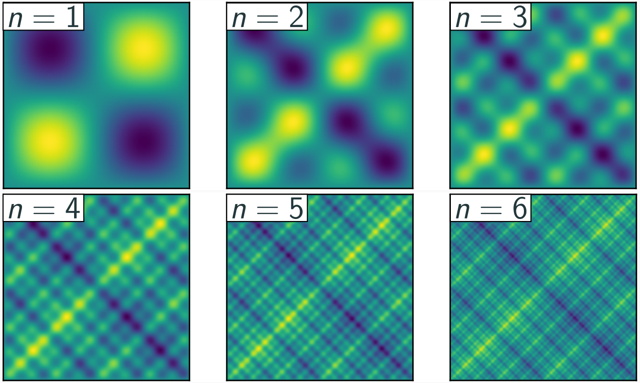
Multi-Frequency Problem

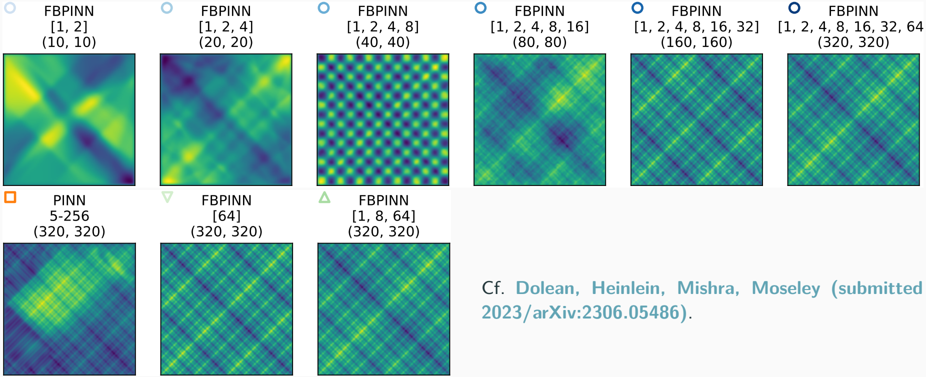
Let us now consider the **two-dimensional multi-frequency Laplace boundary value problem**

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

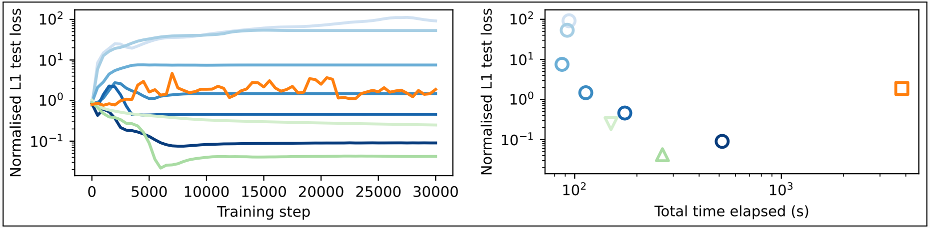
with $\omega_i = 2^i$.

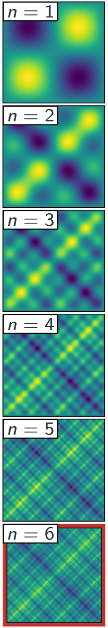
For increasing values of n, we obtain the **analytical solutions**:



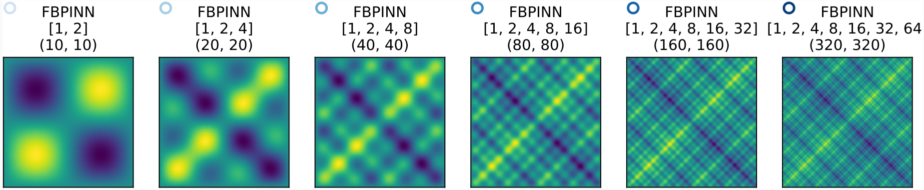


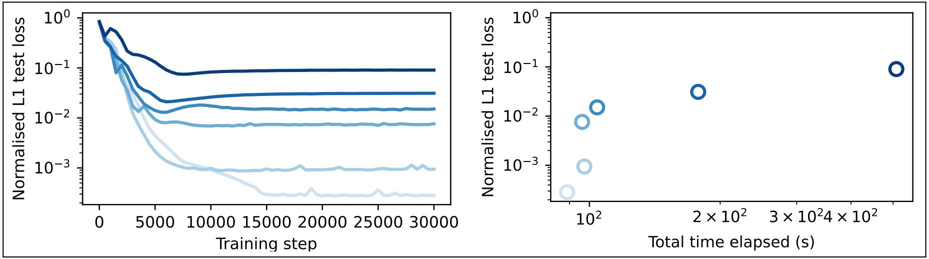
Cf. Dolean, Heinlein, Mishra, Moseley (submitted 2023/arXiv:2306.05486).

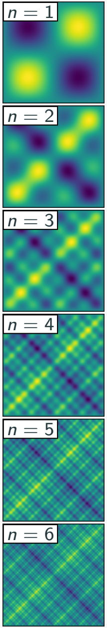




Multi-Level FBPINNs for a Multi-Frequency Problem – Weak Scaling





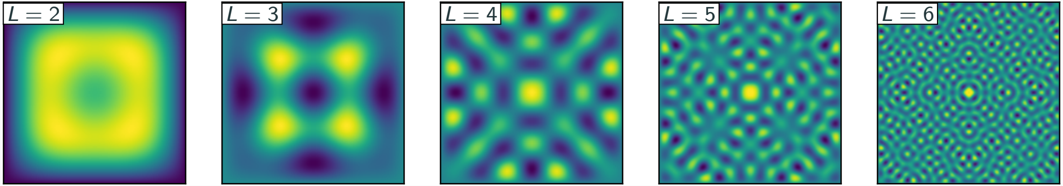


Cf. **Dolean, Heinlein, Mishra, Moseley (submitted 2023/arXiv:2306.05486)**.

Finally, let us consider the **two-dimensional Helmholtz boundary value problem**

$$\Delta u - k^2 u = f \quad \text{in } \Omega = [0,1]^2,$$
$$u = 0 \quad \text{on } \partial\Omega,$$
$$f(\mathbf{x}) = e^{-\frac{1}{2}(\|\mathbf{x}-0.5\|/\sigma)^2}.$$

With $k = 2^L\pi/1.6$ and $\sigma = 0.8/2^L$, we obtain the **solutions**:



Multilevel FBPINNs – 2D Helmholtz Problem

Let us consider the **two-dimensional Helmholtz boundary value problem**

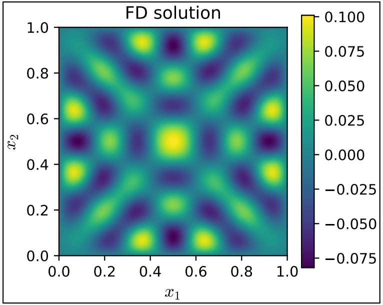
$$\Delta u - k^2 u = f \quad \text{in } \Omega = [0,1]^2,$$

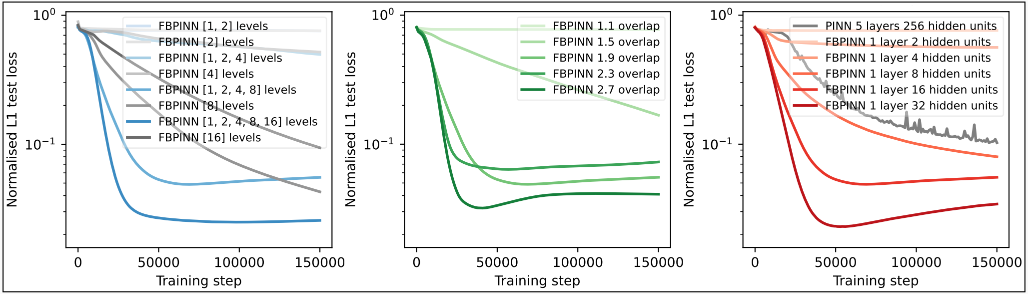
$$u = 0 \quad \text{on } \partial\Omega,$$

$$f(\mathbf{x}) = e^{-\frac{1}{2}(\|\mathbf{x}-0.5\|/\sigma)^2}.$$

with $k = 2^4\pi/1.6$ and $\sigma = 0.8/2^4$.

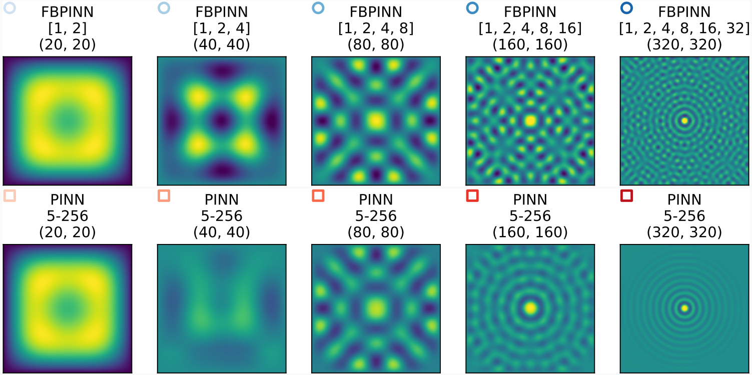
We compute a **reference solution** using **finite differences** with a 5-point stencil on a 320×320 grid.

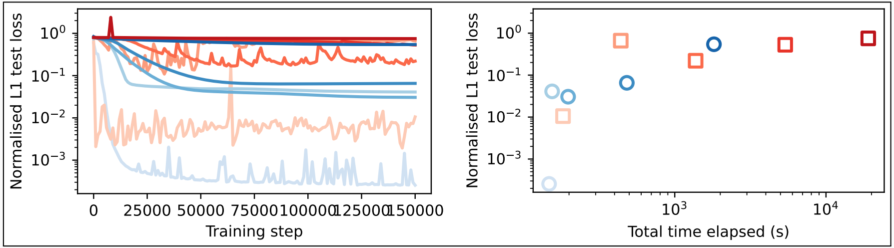


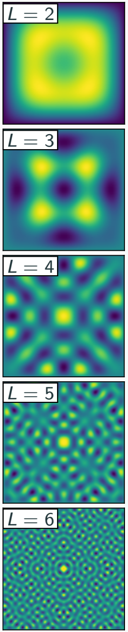


Cf. Dolean, Heinlein, Mishra, Moseley (submitted 2023/arXiv:2306.05486).

Multi-Level FBPINNs for the Helmholtz Problem – Weak Scaling







PINNs

- **Training of PINNs is often problematic** when:
  - **scaling to large domains / high frequency solutions**
  - **multiple loss terms** have to be balanced
- **Convergence of PINNs has yet to be understood better**

(Multilevel) FBPINNs

- **Schwarz domain decomposition approaches 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**.

Outlook

- Investigate, e.g.,
  - more complex / realistic geometries and boundary conditions
  - unstructured domain decompositions
  - three dimensional problems (already possible in the implementation)
- Theoretical convergence analysis

Thank you for your attention!