Multi-level domain decomposition-based physics-informed neural networks

Alexander Heinlein[1]

28th International Conference on Domain Decomposition Methods (DD28), KAUST, Saudi Arabia, January 28 - February 1, 2024

[1]Delft University of Technology

Based on joint work with Damien Beecroft (University of Washington), Victorita Dolean (TU Eindhoven), Amanda A. Howard and Panos Stinis (Pacific Northwest National Laboratory), 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**

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

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

Construction of the trial functions

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

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

- N is a **feedforward neural network** with **trainable parameters** θ 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**

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**

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

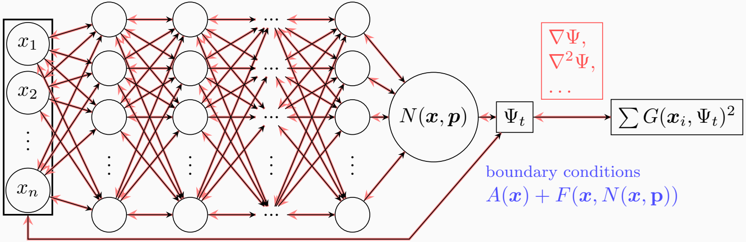
where $\Psi_t\left(\boldsymbol{x}, \boldsymbol{\theta}\right)$ is a **trial function**, $\boldsymbol{x}_i$ **sampling points inside the domain** Ω and $\boldsymbol{\theta}$ are **adjustable parameters**.

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**

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

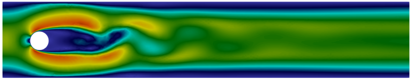
It is based on the approach by Lagaris et al. (1998). The main novelty of PINNs is the use of a **hybrid loss function**:

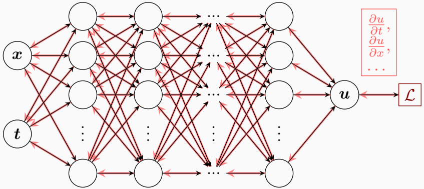
$$\mathcal{L} = \omega_{\text{data}} \mathcal{L}_{\text{data}} + \omega_{\text{PDE}} \mathcal{L}_{\text{PDE}},$$

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

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

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





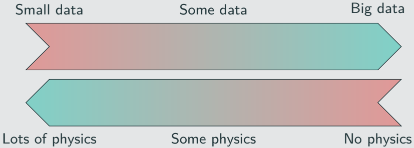
Advantages

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

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

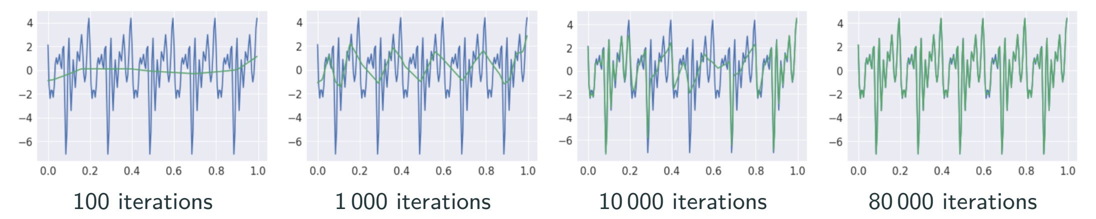
- $\mathcal{E}_G = \mathcal{E}_G(\theta; \boldsymbol{X}) := \|\mathbf{u} - \mathbf{u}^*\|_V$ (V Sobolev space, $\boldsymbol{X}$ training data set)
- $\mathcal{E}_T$ is the training error (l^p loss of the residual of the PDE)
- C_{PDE} and C_{quad} constants depending on the PDE resp. the quadrature
-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

Dependence on the choice of **activation functions**: **Hong et al. (arXiv 2022)**

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

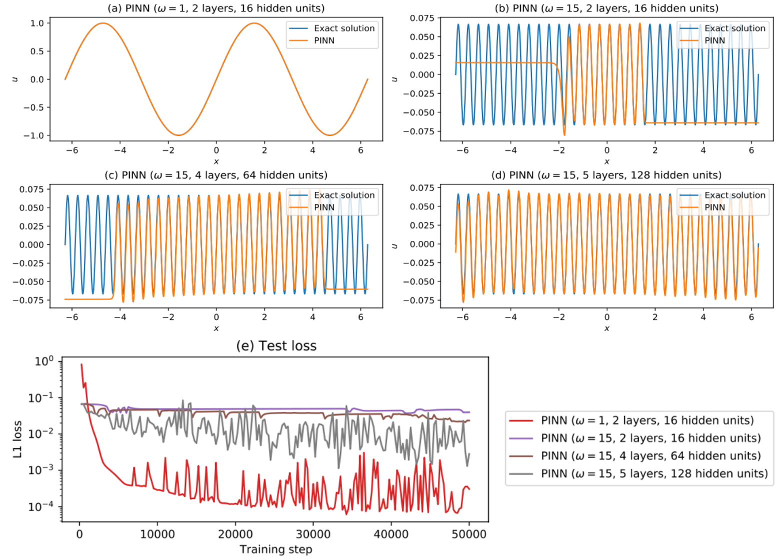
$$
\begin{aligned}
u' &= \cos(\omega \boldsymbol{x}), \\
u(0) &= 0,
\end{aligned}
$$

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

Solve

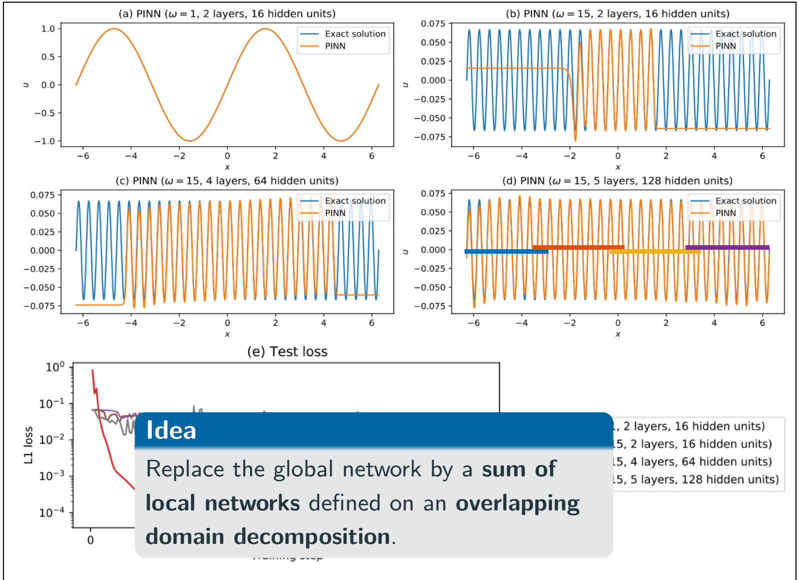
$$\begin{aligned} u' &= \cos(\omega \boldsymbol{x}), \\ u(0) &= 0, \end{aligned}$$

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 **sum of local networks** defined on an **overlapping domain decomposition**.

(a) 321 free parameters

(d) 66 433 free parameters

Machine Learning and Domain Decomposition Methods

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 (2024, subm. 2023 / arXiv:2306.05486); Heinlein, Howard, Beecroft, Stinis (subm. 2024 / arXiv:2401.07888)
- **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.

An overview of the state-of-the-art in the end of 2023:

A. Klawonn, M. Lanser, J. Weber

Machine learning and domain decomposition methods – a survey

arXiv:2312.14050. 2023

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

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

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

FBPINNs use the **network architecture**

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

and the **loss function**

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

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

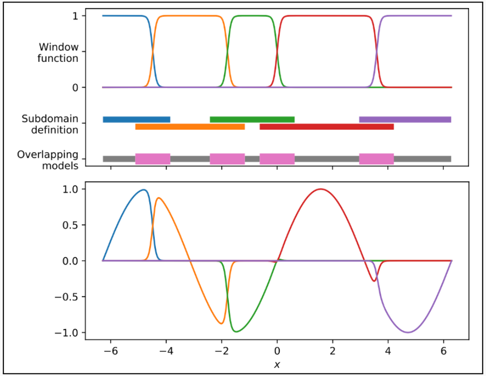
Hard enforcement of boundary conditions

Loss function

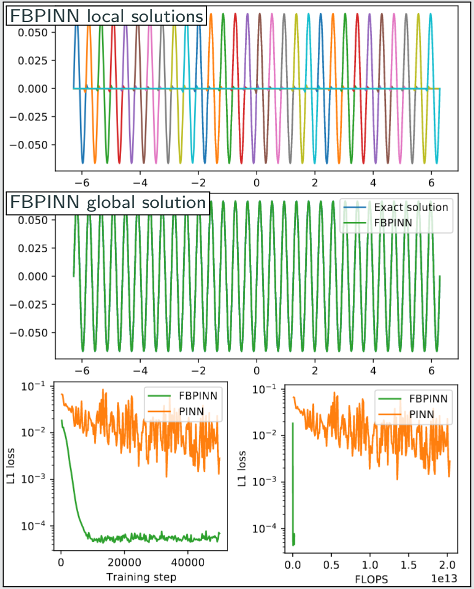
$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left(\mathcal{N}[\mathcal{C} u](\boldsymbol{x}_i, \boldsymbol{\theta}) - f(\boldsymbol{x}_i) \right)^2,$$

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

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



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



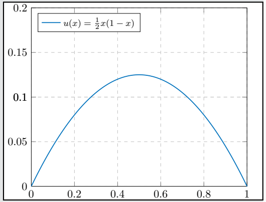
Scalability of FBPINNs

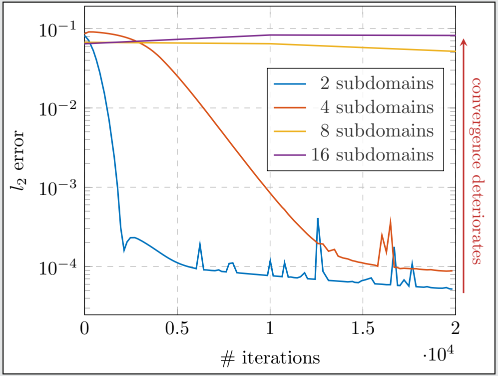
Consider the **simple boundary value problem**

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

which has the **solution**

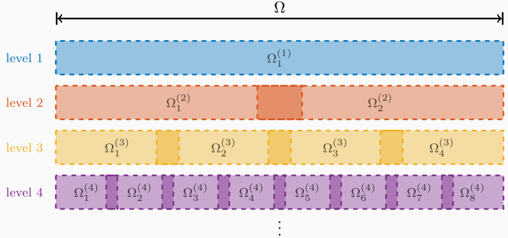
$$u(x) = 1/2\, x(1-x).$$





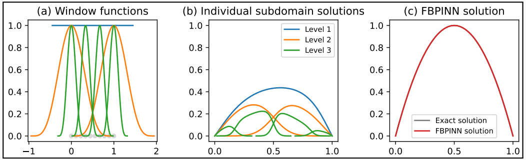
Multi-Level FBPINN Algorithm

Extension of FBPINNs to L levels; Cf. **Dolean, Heinlein, Mishra, Moseley (submitted 2023 / arXiv:2306.05486)**.



L-level network architecture

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



Multi-Frequency Problem

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

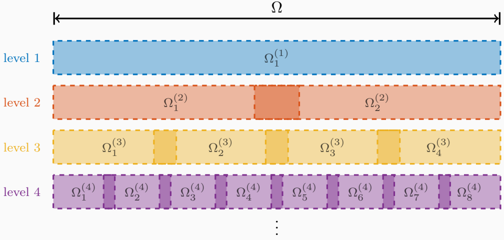
$$-\Delta u = 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) \quad \text{in } \Omega,$$
$$u = 0 \quad \text{on } \partial\Omega,$$

with $\omega_i = 2^i$.

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

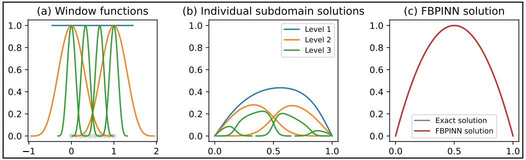
Multi-Level FBPINN Algorithm

Extension of FBPINNs to L levels; Cf. **Dolean, Heinlein, Mishra, Moseley (submitted 2023 / arXiv:2306.05486)**.



L-level network architecture

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



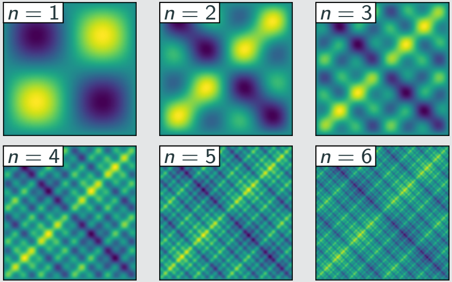
Multi-Frequency Problem

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

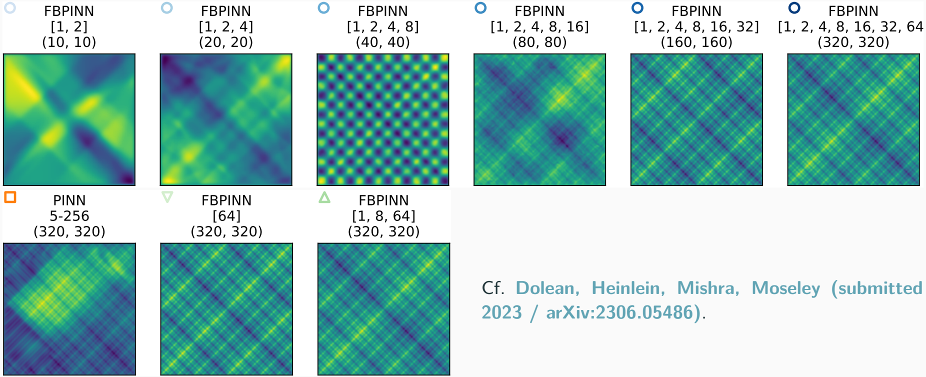
$$\begin{aligned} -\Delta u &= 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

with $\omega_i = 2^i$.

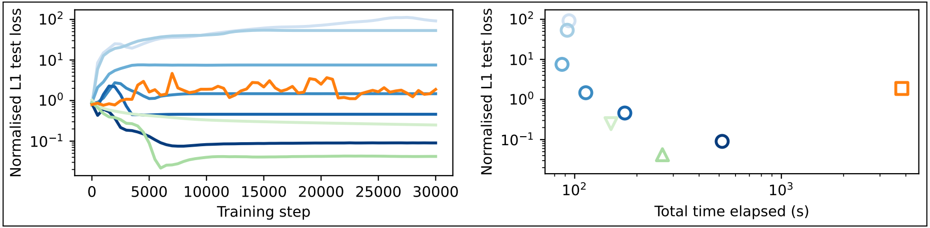
For increasing values of n, we obtain the **analytical solutions**:

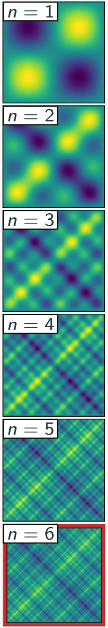


Multi-Level FBPINNs for a Multi-Frequency Problem – Strong Scaling

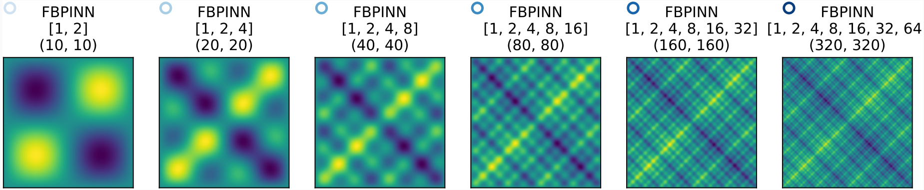


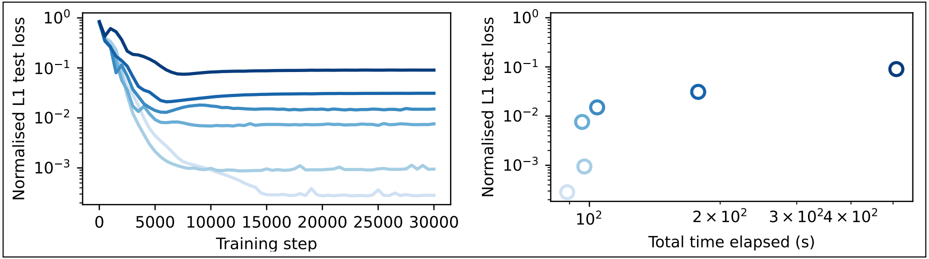
Cf. Dolean, Heinlein, Mishra, Moseley (submitted 2023 / arXiv:2306.05486).

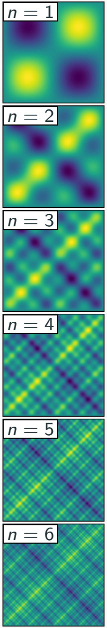




Multi-Level FBPINNs for a Multi-Frequency Problem – Weak Scaling







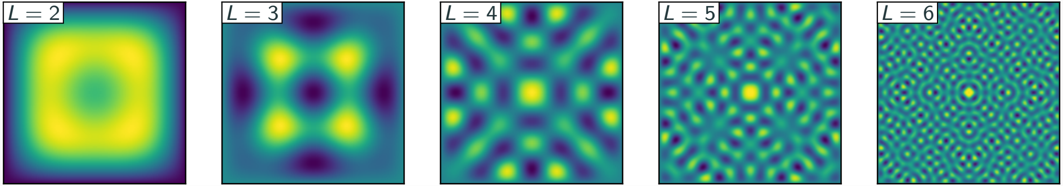
- Ongoing: **analysis and improvement of the convergence**

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

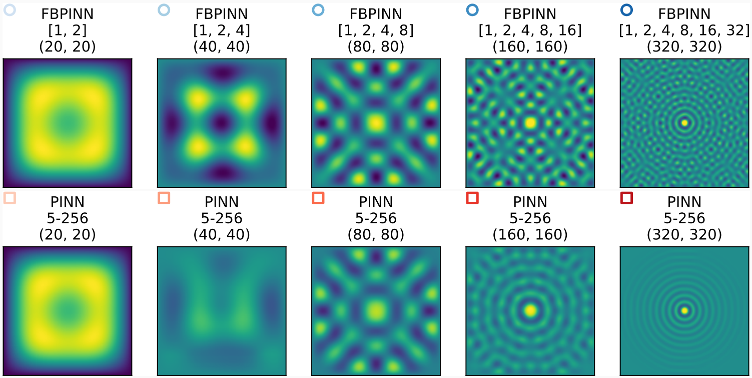
Finally, let us consider the **two-dimensional Helmholtz boundary value problem**

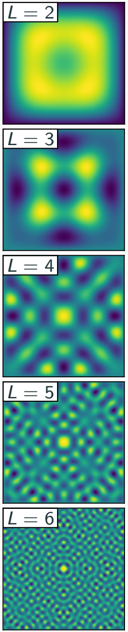
$$\begin{aligned} \Delta u - k^2 u &= f \quad \text{in } \Omega = [0,1]^2, \\ u &= 0 \quad \text{on } \partial\Omega, \\ f(\mathbf{x}) &= e^{-\frac{1}{2}(\|\mathbf{x}-0.5\|/\sigma)^2}. \end{aligned}$$

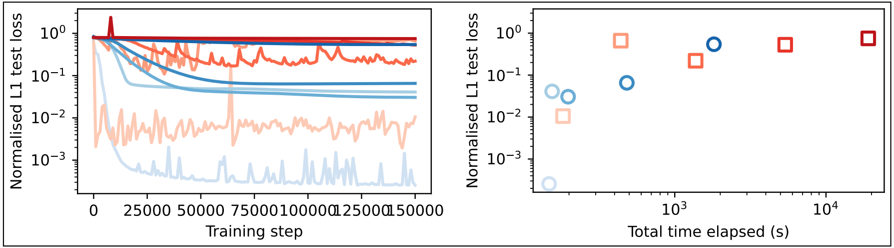
With $k = 2^L\pi/1.6$ and $\sigma = 0.8/2^L$, we obtain the **solutions**:



Multi-Level FBPINNs for the Helmholtz Problem – Weak Scaling



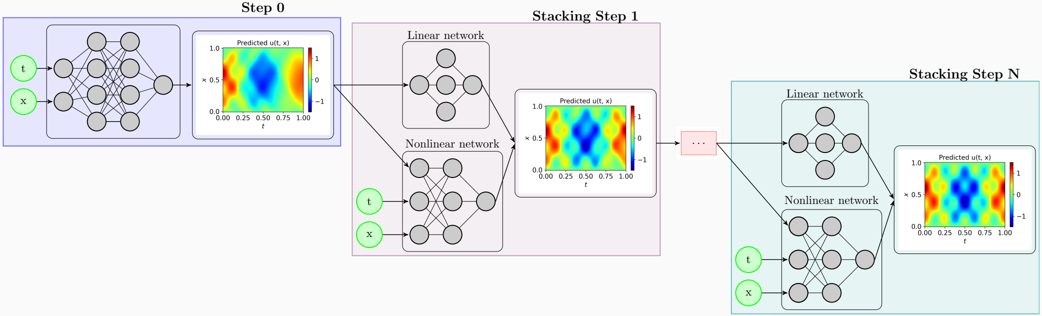




Stacking Multifidelity FBPINNs

In the **stacking multifidelity PINNs** approach introduced in **Howard, Murphy, Ahmed, Stinis (arXiv 2023)**, **multiple networks are stacked on top of each other** in a recursive way. In particular, the next model $\hat{u}^{MF}$ is trained as a corrector for the previous model $\hat{u}^{SF}$:

$$\hat{u}^{MF}(\mathbf{x}, \theta^{MF}) = (1 - |\alpha|)\,\hat{u}^{MF}_{linear}(\mathbf{x}, \hat{u}^{SF}, \theta^{MF}) + |\alpha|\,\hat{u}^{MF}_{nonlinear}(\mathbf{x}, \hat{u}^{SF}, \theta^{MF})$$



Stacking multifidelity FBPINNs

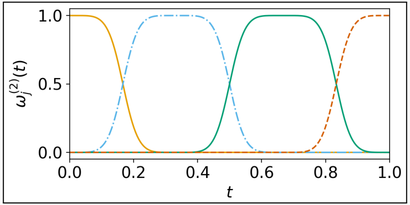
We **combine stacking multifidelity PINNs with FBPINNs** by using an FBPINN model (with an increasing number of subdomains) in each stacking step. → **One-way sequential coupling** of the levels

Cf. **Heinlein, Howard, Beecroft, Stinis (subm. 2024 / arXiv:2401.07888)**

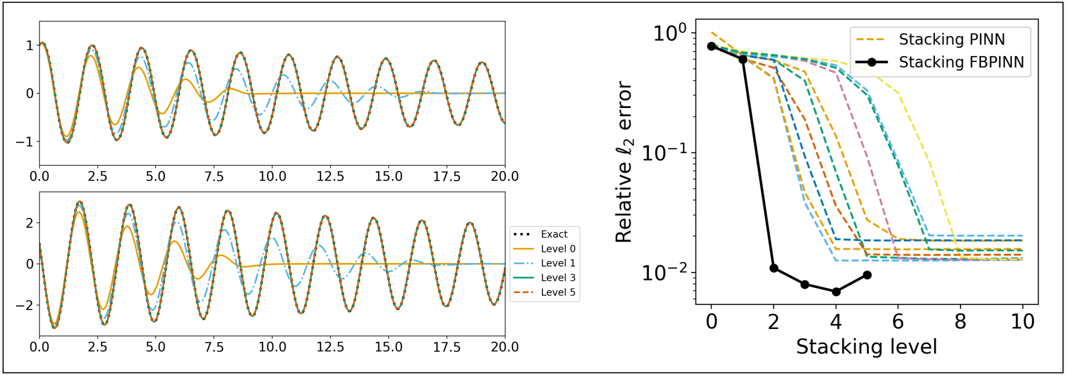
First, we consider a **pedulum problem** and **compare the stacking multifidelity PINN and FBPINN** approaches:

$$\frac{ds_1}{dt} = s_2,$$
$$\frac{ds_2}{dt} = -\frac{b}{m}s_2 - \frac{g}{L}\sin(s_1)$$

with $m = L = 1$, $b = 0.05$, $g = 9.81$, and $T = 20$.



Exemplary partition of unity in time

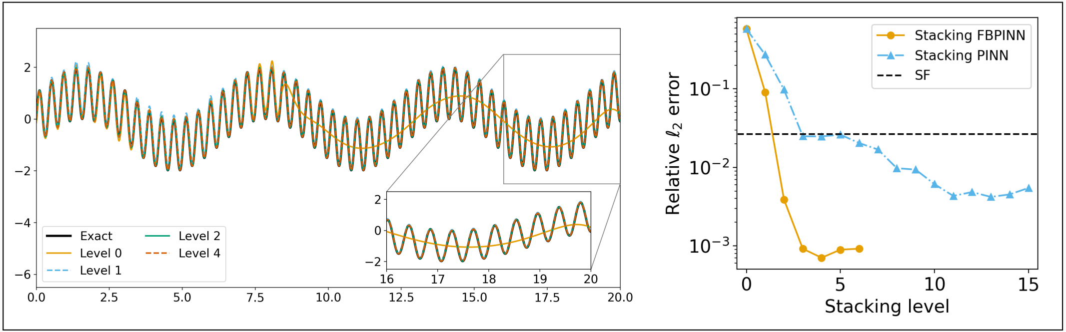


Numerical Results – Two-Frequency Problem

Second, we consider a **two-frequency problem**:

$$\frac{ds}{dx} = \omega_1 \cos(\omega_1 x) + \omega_2 \cos(\omega_2 x),$$
$$s(0) = 0,$$

on domain $\Omega = [0, 20]$ with $\omega_1 = 1$ and $\omega_2 = 15$.

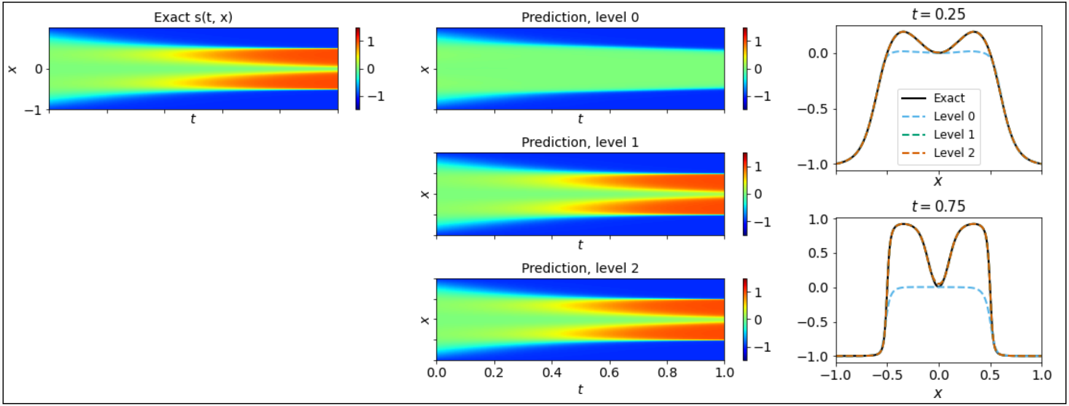


→ Due to the **multiscale structure of the problem**, the **improvements** due to the **multifidelity FBPINN approach** are **even stronger**.

Numerical Results – Allen–Cahn Equation

Finally, we consider the **Allen–Cahn equation**:

$$
\begin{aligned}
s_t - 0.0001 s_{xx} + 5s^3 - 5s &= 0, && t \in (0,1], x \in [-1,1],\\
s(x,0) &= x^2 \cos(\pi x), && x \in [-1,1],\\
s(x,t) &= s(-x,t), && t \in [0,1], x=-1, x=1,\\
s_x(x,t) &= s_x(-x,t), && t \in [0,1], x=-1, x=1.
\end{aligned}
$$



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**.

Multifidelity stacking FBPINNs

- The **combination of multifidelity stacking PINNs with the multilevel FBPINN approach** yields **significant improvements in the accuracy and efficiency** for **time-dependent problems**.

Thank you for your attention!

→ Talk by **Victorita Dolean** on Tuesday, 10.45 am, in Hall 2: *Domain decomposition-based training strategies for PINNs*

Details

Date: April 24-26 2024

Location: Delft University of Technology

This workshop brings together scientists from mathematics, computer science, and application areas working on computational and mathematical methods in data science.

Confirmed invited speakers

- Christoph Brune (University of Twente)
- Victorita Dolean (TU Eindhoven)
- Thomas Richter
- ...

For more details, see
https://searhein.github.io/gamm-cominds-2024/