Parallelization approaches for neural network-based collocation methods

Victorita Dolean (TU Eindhoven)

with: A. Heinlein, S. Mishra, B. Moseley, Valentin Mercier, Serge Gratton

COMinDS workshop, Delft, April 26, 2024

Introduction

Scientific Machine Learning as a Standalone Field

Priority Research Directions

Foundational research themes:

- Domain-awareness
- Interpretability
- Robustness

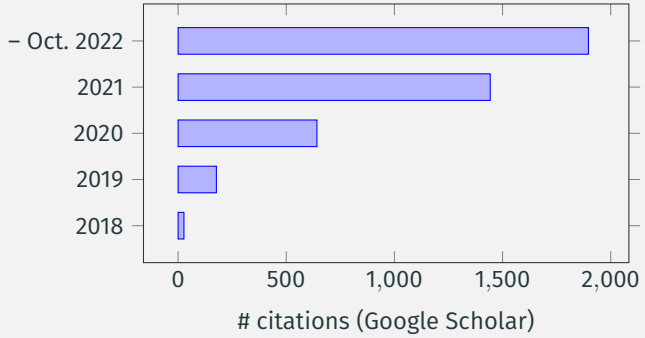
Capability research themes:

- Massive scientific data analysis
- Machine learning-enhanced modeling and simulation
- Intelligent automation and decision-support for complex systems

N. Baker, A. Frank, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, K. Willcox, and S. Lee.

Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for AI, Washington, 2019.

Development of the Field of Scientific Machine Learning



M. Raissi, P. Perdikaris, and G. E. Karniadakis.
Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.
Journal of Computational physics, 378, 686-707. 2019.

(and the respective arXiv preprints)

Physics-informed machine learning

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.

Idea: Solve a general differential equation subject to boundary conditions

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

by solving an **optimization problem**

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

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

Main Features of the Method

Construction of the trial functions

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

$$\Psi_t(\mathbf{x}, \mathbf{p}) = A(\mathbf{x}) + F(\mathbf{x}, N(\mathbf{x}, \mathbf{p})),$$

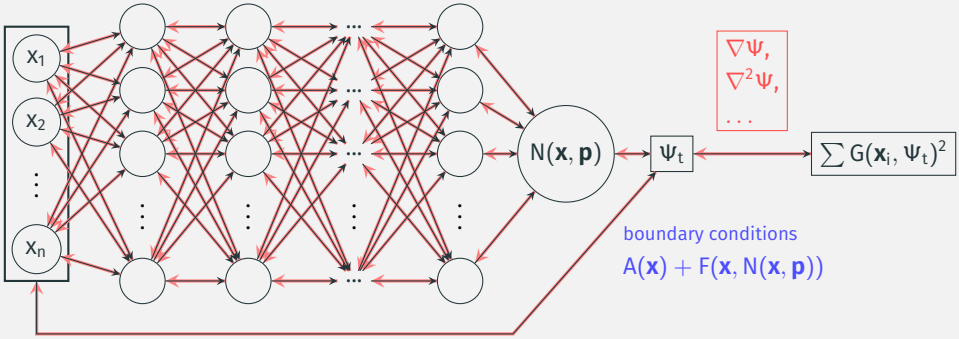
where

- $N(\mathbf{x}, \mathbf{p})$ is a **trainable feedforward neural network** with parameters $\mathbf{p}$ and input $x \in \mathbb{R}^n$ and
- the functions A and F are **fixed functions** chosen such that
  - A **satisfies the boundary conditions**, and
  - F **does not contribute to the boundary conditions**.

From the conclusion of the paper:

> "The **success of the method** can be attributed to **two factors**. The first is the employment of neural networks that are **excellent function approximators** and the second is the form of the **trial solution that satisfies by construction the BC's** and therefore the constrained optimization problem becomes a substantially simpler unconstrained one."

Sketch of the Approach by Lagaris et al. (1998)



Physics-Informed Neural Networks (PINNs)

In **Raissi, Perdikaris, Karniadakis (2019)**, the authors have revisited and modified the approach by **Lagaris et al. (1998)**, denoting their method as **physics-informed neural networks (PINNs)**. Consider the partial differential equation

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

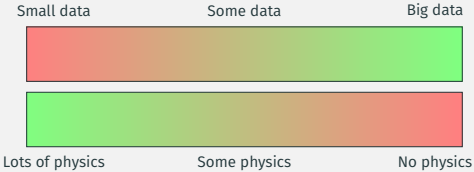
The main novelty of PINNs is that a **hybrid loss function** is used for training the feedforward neural network:

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

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

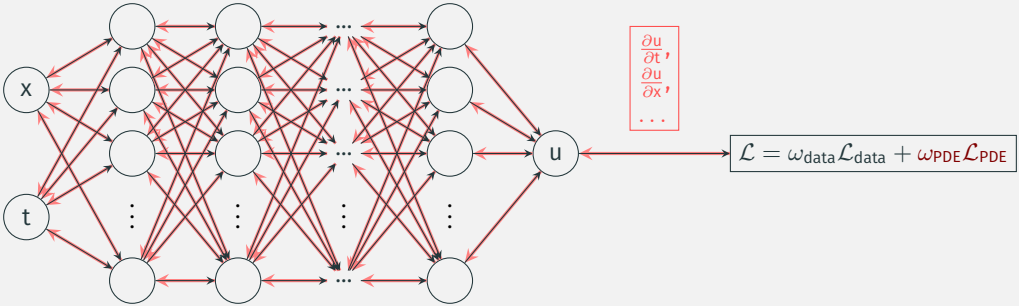
$$\mathcal{L}_{data} \quad = \quad \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} \left(u(x_i,t_i) - u_i\right)^2,$$

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



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

Sketch of the PINN approach by Raissi et al.



Advantages and Drawbacks of PINNs

Advantages

- **Mesh free**
- **Mostly unsupervised** and work with **incomplete models** (e.g., we learn only the missing physics) and **imperfect data**.
- Strong generalization properties with **small data** due to **embedded physics**.
- **High dimensional problems** (PDEs like Black-Scholes, Allen-Cahn).
- Solve **inverse and forward problems**, stationary and time-dependent, assimilate data in the **same way**.

Drawbacks

- **Large computational cost** associated with the training of the neural networks.
- Generally, the training process is **not robust** and **depends heavily on well-chosen weights**
- **Convergence** properties **not well-understood**.
- **Poor scaling** to large domains.
- Learning high frequencies (**spectral bias**) / multi-scale solutions is **difficult**.

Available theoretical results: PINNs

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

Estimate of the generalisation error

The generalisation error (or total error) verifies

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

where

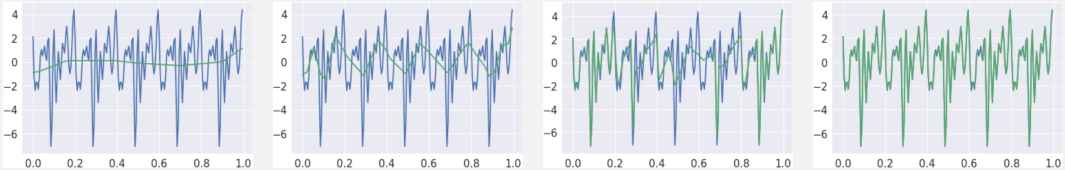
- $\mathcal{E}_G = \mathcal{E}_G(\theta^*; \mathcal{S}) := \|\mathbf{u} - \mathbf{u}^*\|_X$
- $\mathcal{E}_T$ is the training error
- C_{pde} and C_{quad}: constants depending on the PDE and the quadrature
- N: number of the training points

The devil is in the details:

"As long as the PINN is trained well it generalises well"

Scaling Issues in Neural Network Training

- **Spectral bias:** Neural networks prioritise learning lower frequency functions first irrespective of their amplitude.



Rahaman, N., et al, On the spectral bias of neural networks**, ICML (2019)**
Hong, Siegel, Tan, Xu, On the Activation Function Dependence of the Spectral Bias of Neural Networks**, arXiv (2022)**

- 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**

When PINNs fail to train?

Perdikaris, P. et al, When and why PINNs fail to train: A neural tangent kernel perspective, 2022.

Neural tangent kernel (NTK) theory in a nutshell

- Write the gradient descent method at the continuous level ($\mathcal{L}$ is the loss function)

$$\frac{d\boldsymbol{\theta}}{dt} = -\nabla\mathcal{L}(\boldsymbol{\theta}),\ \mathcal{L}(\boldsymbol{\theta}), := \omega_b \sum \mathcal{R}(\mathbf{x}_b, \boldsymbol{\theta}(t))^2 + \omega_r \sum \mathcal{R}(\mathbf{x}_r, \boldsymbol{\theta}(t))^2$$

- Residual vectors in the collocation points obey an ODE

$$\frac{d}{dt}\begin{bmatrix} \mathcal{R}(\mathbf{x}_b, \boldsymbol{\theta}(t)) \\ \mathcal{R}(\mathbf{x}_r, \boldsymbol{\theta}(t)) \end{bmatrix} = -K(t)\begin{bmatrix} \mathcal{R}(\mathbf{x}_b, \boldsymbol{\theta}(t)) \\ \mathcal{R}(\mathbf{x}_r, \boldsymbol{\theta}(t)) \end{bmatrix}$$

- The Neural Tangent Kernel $K(t) \to K^*$ for infinitely wide and shallow networks
- Spectral properties if K^* explain the speed of training.

"To provide further insight, we analyze the training dynamics of fully-connected PINNs through the lens of their NTK and show that not only they suffer from spectral bias, but they also exhibit a discrepancy in the convergence rate among the different loss components contributing to the total training error"

A Motivation for Domain Decomposition Approaches

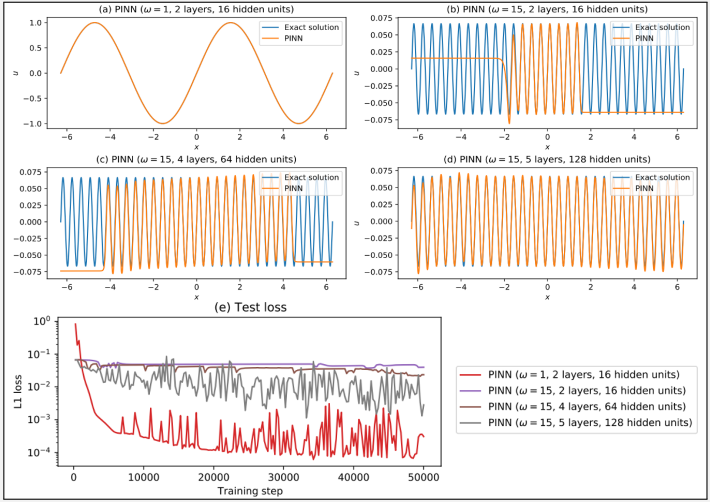
Solve

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

for different values of ω.

Scaling issues

- Size of the computational domain
- Size of frequencies

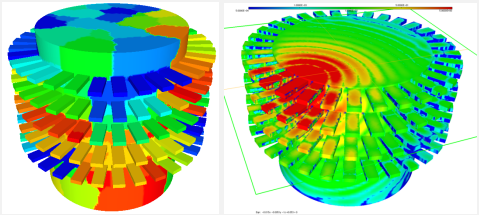


(a) 321 free parameters (d) 66 433 free parameters

Cf. **Moseley, Markham, and Nissen-Meyer (arXiv 2021)**

Domain decomposition-based training strategies

Domain Decomposition Methods



Images based on **Tournier et al. (2019)**

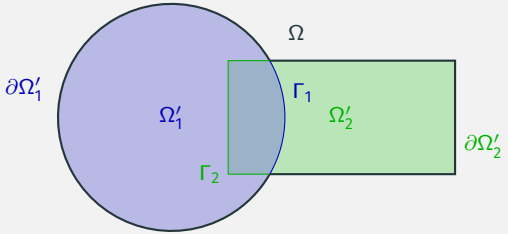
Idea

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

- Better **robustness** (like direct solvers) and **scalability** (like iterative solvers).
- Improved **computational efficiency**
- Introduce **parallelism**

Historical remarks: The **alternating Schwarz method** is the earliest **domain decomposition method (DDM)**, which was introduced by **H. A. Schwarz** in **1870**:

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



Dolean, Jolivet, Nataf: An Introduction to Domain Decomposition Methods: algorithms, theory and parallel implementation, SIAM, 2015.

Machine Learning and Domain Decomposition Methods

A **non-exhaustive overview**:

- **Machine Learning-enhanced adaptive FETI–DP** (finite element tearing and interconnecting – dual primal): **Heinlein, Klawonn, Lanser, Weber (2019, 2020, 2021, 2022); Klawonn, Lanser, Weber (preprint 2022)**
- **Domain decomposition for CNNs**: **Gu et al. (2022); Lee, Park, Lee (2022); Klawonn, Lanser, Weber (2023)**
- **D3M** (deep domain decomposition method): **Li, Tang, Wu, and Liao (2019)**
- **DeepDDM** (deep-learning-based DDM): **Li, Xiang, Xu (2020)**, **Mercier, Gratton (arXiv 2021)**
- **cPINNs** (conservative physics-informed neural networks): **Jagtap, Kharazmi, and Karniadakis (2020)**
- **XPINNs** (extended physics-informed neural networks): **Jagtap, Karniadakis (2020)**
- **Schwarz Domain Decomposition Algorithm for PINNs**: **Kim, Yang (2022, arXiv 2022)**
- **FBPINNs** (finite basis physics-informed neural networks): **Moseley, Markham, and Nissen-Meyer (2021), Dolean, Heinlein, Mishra, Moseley (2024, subm. 2023,); Heinlein, Howard, Beecroft, Stinis (subm. 2024**

An overview of the SOTA 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 SOTA end of 2023:

A. Klawonn, M. Lanser, J. Weber

Machine learning and domain decomposition methods – a survey

arXiv:2312.14050. 2023

XPINNs (extended physics-informed neural networks)

Jagtap, A. D., Karniadakis, G. E. Extended physics-informed neural networks (XPINNs), Communications in Computational Physics (2020)

$$\mathcal{J}(\tilde{\Theta}_q) = W_{u_q}\,\mathrm{MSE}_{u_q}(\tilde{\Theta}_q;\{\mathbf{x}_{u_q}^{(i)}\}_{i=1}^{N_{u_q}}) + W_{\mathcal{F}_q}\,\mathrm{MSE}_{\mathcal{F}_q}(\tilde{\Theta}_q;\{\mathbf{x}_{F_q}^{(i)}\}_{i=1}^{N_{F_q}}) + W_{I_q}\underbrace{\mathrm{MSE}_{u_{avg}}(\tilde{\Theta}_q;\{\mathbf{x}_{I_q}^{(i)}\}_{i=1}^{N_{I_q}})}_{\text{Interface condition}}$$

$$+ W_{I_{\mathcal{F}_q}}\underbrace{\mathrm{MSE}_{\mathcal{R}}(\tilde{\Theta}_q;\{\mathbf{x}_{I_q}^{(i)}\}_{i=1}^{N_{I_q}})}_{\text{Interface condition}} + \underbrace{\text{Additional Interface Condition's}}_{\text{Optional}},$$

$$\mathrm{MSE}_{u_q}(\tilde{\Theta}_q;\{\mathbf{x}_{u_q}^{(i)}\}_{i=1}^{N_{u_q}}) = \frac{1}{N_{u_q}}\sum_{i=1}^{N_{u_q}}\left|u_q^{(i)} - u_{\tilde{\Theta}_q}(\mathbf{x}_{u_q}^{(i)})\right|^2,$$

$$\mathrm{MSE}_{\mathcal{F}_q}(\tilde{\Theta}_q;\{\mathbf{x}_{F_q}^{(i)}\}_{i=1}^{N_{F_q}}) = \frac{1}{N_{F_q}}\sum_{i=1}^{N_{F_q}}\left|\mathcal{F}_{\tilde{\Theta}_q}(\mathbf{x}_{F_q}^{(i)})\right|^2,$$

$$\mathrm{MSE}_{u_{avg}}(\tilde{\Theta}_q;\{\mathbf{x}_{I_q}^{(i)}\}_{i=1}^{N_{I_q}}) = \sum_{\forall q^+}\left(\frac{1}{N_{I_q}}\sum_{i=1}^{N_{I_q}}\left|u_{\tilde{\Theta}_q}(\mathbf{x}_{I_q}^{(i)}) - \left\{\left\{u_{\tilde{\Theta}_q}(\mathbf{x}_{I_q}^{(i)})\right\}\right\}\right|^2\right),$$

$$\mathrm{MSE}_{\mathcal{R}}(\tilde{\Theta}_q;\{\mathbf{x}_{I_q}^{(i)}\}_{i=1}^{N_{I_q}}) = \sum_{\forall q^+}\left(\frac{1}{N_{I_q}}\sum_{i=1}^{N_{I_q}}\left|\mathcal{F}_{\tilde{\Theta}_q}(\mathbf{x}_{I_q}^{(i)}) - \mathcal{F}_{\tilde{\Theta}_{q^+}}(\mathbf{x}_{I_q}^{(i)})\right|^2\right).$$

Advantages

- Separate NNs for each subdomain
- Allows PINN to be trained in parallel

Drawbacks

- Multiple terms in the loss function can slow training if the weights are not well chosen
- Do not mimick completely the DD behaviour

Finite Basis Physics-Informed Neural Networks (FBPINNs)

In the **finite basis physics informed neural network (FBPINNs) method** introduced in **Moseley, Markham, and Nissen-Meyer (arXiv 2021)**, we solve the boundary value problem

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

using neural networks, we employ the **PINN** approach and **enforce the boundary conditions using a constraining operator**, similar to **Lagaris et al. (1998)**.

Weak enforcement of boundary conditions

Loss function

$$\mathcal{L}(\boldsymbol{\theta}) = w_I \mathcal{L}_{PDE} + w_B \mathcal{L}_{BC},$$

where

$$\begin{aligned} \mathcal{L}_{PDE}(\boldsymbol{\theta}) &= \frac{1}{N_I} \sum_{i=1}^{N_I} (\mathcal{N}[u](\mathbf{x}_i, \boldsymbol{\theta}) - f(\mathbf{x}_i))^2, \\ \mathcal{L}_{BC}(\boldsymbol{\theta}) &= \frac{1}{N_B} \sum_{i=1}^{N_B} (\mathcal{B}_k[u](\mathbf{x}_i, \boldsymbol{\theta}) - g_k(\mathbf{x}_i))^2. \end{aligned}$$

Hard enforcement of boundary conditions

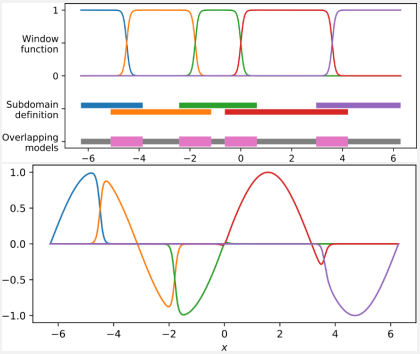
Loss function

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

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

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

FBPINNs – Overlapping Domain Decomposition



- Domain decomposition: $\Omega = \cup_{j=1}^{J} \Omega_j$
- Collocation points (global): $\{x_i\}_{i=1}^{N}$
- Overlapping/interior parts Ω_j° and Ω_j^{int}
- Local solutions u_j, window functions w_j
- Global solution $\mathcal{C}u = \mathcal{C} \sum_{j, x_i \in \Omega_j} \omega_j u_j$

Global loss function

$$\mathcal{L}(\theta_1, \dots, \theta_J) = \underbrace{\frac{1}{N} \sum_{j=1}^{J} \sum_{x_i \in X_j^\circ} \left(\mathcal{N}[\mathcal{C} \sum_{l, x_i \in X_l} \omega_l u_l](x_i, \theta_l) - f(x_i) \right)^2}_{=: \mathcal{L}^\circ(\theta_1, \dots, \theta_J)} + \underbrace{\frac{1}{N} \sum_{j=1}^{J} \sum_{x_i \in X_j^{\text{int}}} \left(\mathcal{N}[\mathcal{C} \sum_{l, x_i \in X_l} \omega_l u_l](x_i, \theta_l) - f(x_i) \right)^2}_{=: \mathcal{L}^{\text{int}}(\theta_1, \dots, \theta_J)}.$$

Since $X_i^{\text{int}} \cap X_j^{\text{int}} = \emptyset$ for $i \neq j$,

$$\mathcal{L}^{\text{int}}(\theta_1, \dots, \theta_J) = \frac{1}{N} \sum_{j=1}^{J} \sum_{x_i \in X_j^{\text{int}}} (\mathcal{N}[\mathcal{C} \omega_j u_j](x_i, \theta_j) - f(x_i))^2$$

- The subdomains can be split active (trained in parallel) and inactive (fixed)
- This corresponds to classical parallel (all active) or multiplicative (one active at a time) Schwarz methods

DeepDDM (deep-learning-based domain decomposition method)

W. Li, X. Xiang, Y. Xu Deep domain decomposition method: Elliptic problems. Mathematical and Scientific Machine Learning (2020)

Train locally by Batch SGD and transmitting only interface values to the neighbouring subdomains.

Local loss functions contain both **volume, boundary and interface jump terms**.

$$\mathcal{M}_s(\theta;\mathbf{X}_s) := \mathcal{M}_{\Omega_s}(\theta;\mathbf{X}_{f_s}) + \mathcal{M}_{\partial\Omega_s\setminus\Gamma_s}(\theta;\mathbf{X}_{g_s}) + \mathcal{M}_{\Gamma_s}(\theta;\mathbf{X}_{\Gamma_s}),$$

$$\mathcal{M}_{\Omega_s}(\theta;\mathbf{X}_{fs}) := \frac{1}{N_{f_s}}\sum_{i=1}^{N_{f_s}} \left|\mathcal{L}\left(h_s(\mathbf{x}_{f_s}^i;\theta)\right) - f(\mathbf{x}_{f_s}^i)\right|^2,$$

$$\mathcal{M}_{\partial\Omega_s\setminus\Gamma_s}(\theta;\mathbf{X}_{g_s}) := \frac{1}{N_{g_s}}\sum_{i=1}^{N_{g_s}} \left|\mathcal{B}\left(h_s(\mathbf{x}_{g_s}^i;\theta)\right) - g(\mathbf{x}_{g_s}^i)\right|^2,$$

$$\mathcal{M}_{\Gamma_s}(\theta;\mathbf{X}_{\Gamma_s}) := \frac{1}{N_{\Gamma_s}}\sum_{i=1}^{N_{\Gamma_s}} \left|\mathcal{D}\left(h_s(\mathbf{x}_{\Gamma_s}^i;\theta)\right) - \mathcal{D}\left(h_r(\mathbf{x}_{\Gamma_s}^i;\theta)\right)\right|^2,$$

Advantages

- The outer-iteration depends on the size overlap and the number of subdomains
- DeepDDM can easily handle PDE with **curved interfaces and heterogeneities**.

Drawbacks

- **No quantitative estimates** of the convergence rate of deep learning solving PDE.
- What is the standard to design the best network architecture?

W. Li, Z. Wang, T. Cui, X. Xiang, Y. Xu Deep domain decomposition method: Helmholtz equation. Advances in Applied Mathematics and Mechanics (2023)

Train locally by transmitting only Robin interface values to the neighbouring subdomains.
Use **PWNN (plane wave NN)** instead of PINNs to account for oscillatory nature of the solution.

$$\mathcal{M}(\Theta) = \mathcal{M}_{\Omega_1}(\Theta) + \mathcal{M}_{\partial\Omega_1\setminus\Gamma}(\Theta) + \mathcal{M}_{\Gamma}(\Theta),$$

$$\mathcal{M}_{\Omega_1}(\Theta) = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|-\Delta\mathcal{N}(\mathbf{x}_f^i;\Theta) - k^2\mathcal{N}(\mathbf{x}_f^i;\Theta) - f_1(\mathbf{x}_f^i)\right|^2,$$

$$\mathcal{M}_{\partial\Omega_1\setminus\Gamma}(\Theta) := \frac{1}{N_g}\sum_{i=1}^{N_g}\left|\frac{\partial\mathcal{N}(\mathbf{x}_g^i;\Theta)}{\partial\mathbf{n}_1} + \mathrm{i}k\mathcal{N}(\mathbf{x}_g^i;\Theta) - g_a(\mathbf{x}_g^i)\right|^2,$$

$$\mathcal{M}_{\Gamma}(\Theta) := \frac{1}{N_\Gamma}\sum_{i=1}^{N_\Gamma}\left|\frac{\partial\mathcal{N}(\mathbf{x}_\Gamma^i;\Theta)}{\partial\mathbf{n}_1} + \gamma_1\mathcal{N}(\mathbf{x}_\Gamma^i;\Theta) - g_1(\mathbf{x}_\Gamma^i)\right|^2,$$

Advantages

- Number of outer iterations comparable to the use of FDM with DDM
- **Competitive solution time/iteration** when the wave number increases.

Drawbacks

- Comparison done with the purely iterative version of DDM - what about Krylov accelerated version?
- All relies on the activation function - **number of parameters dependent on** k.

Multilevel FBPINN

FBPINN With Flexible Scheduling

FBPINN training step

if $j \in \mathcal{A}$ (Ω_j is an active domain) **then**

Perform p iterations of gradient descent on $\boldsymbol{\theta}_j$ ($\boldsymbol{\theta}_l^k, l \neq j$ are kept fixed):

$$\boldsymbol{\theta}_j^{k+l} = \boldsymbol{\theta}_j^{k+l-1} - \lambda \nabla_{\boldsymbol{\theta}_j} \mathcal{L}(\boldsymbol{\theta}_1^k, ..., \boldsymbol{\theta}_j^{k+l-1}, ..., \boldsymbol{\theta}_j^k), l = 1, .., p.$$

Update the solutions in the overlap (communication with the neighbours):

$$\forall \mathbf{x} \in \Omega_j^\circ,\ u(\mathbf{x}, \boldsymbol{\theta}_j^{k+p}) \leftarrow \sum_{l, \mathbf{x} \in \Omega_l} \omega_l u_l(\mathbf{x}, \boldsymbol{\theta}_l^{k+p}).$$

Update the gradients in the overlap.

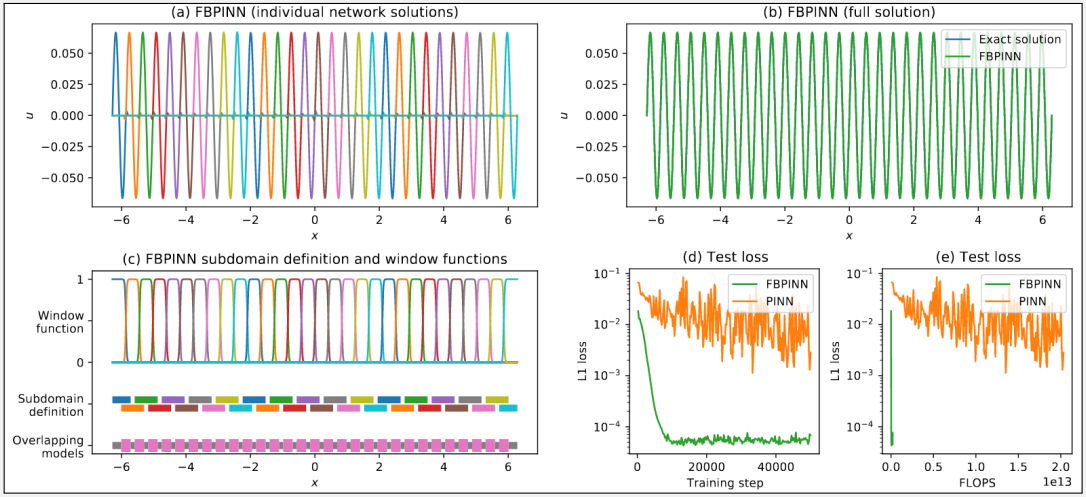
Apply the constraining operator $u(\mathbf{x}, \boldsymbol{\theta}_j^k) \leftarrow \mathcal{C}(u(\mathbf{x}, \boldsymbol{\theta}_j^{k+p}))$.

end if

Summary

- Communication every p iterations (better for overall efficiency)
- Multiplication with a window function : a way to restrict to the local domain.
- The set of active domains can be changed after the local training is completed.

Numerical Results – Comparison of PINNs and FBPINNs



Cf. **Moseley, Markham, and Nissen-Meyer (arXiv 2021)**

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**:

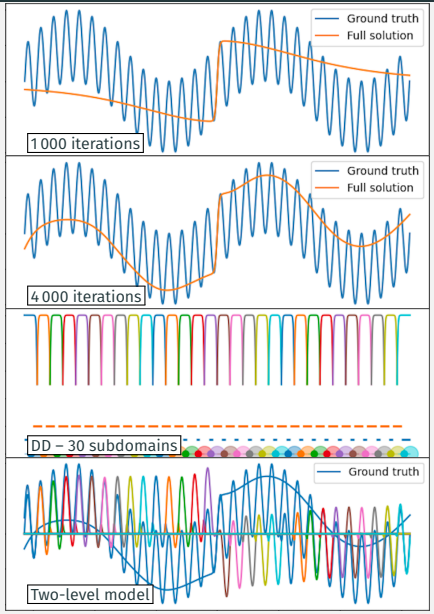
$$u(\theta_0, \theta_1, \dots, \theta_J) = \mathcal{C}\left(\mathbf{u}_0(\theta_1) + \sum_{\Omega_j} \omega_l \mathbf{u}_l(\theta_l)\right)$$

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

$$\begin{cases} \dfrac{du}{dx} &= \omega_1 \cos(\omega_1 x) + \omega_2 \cos(\omega_2 x) \\ u(0) &= 0. \end{cases}$$

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

Cf. **V. Dolean, A. Heinlein, S. Mishra, B. Moseley,** Finite basis physics-informed neural networks as a Schwarz domain decomposition method, **arXiv:2211.05560, 2022**



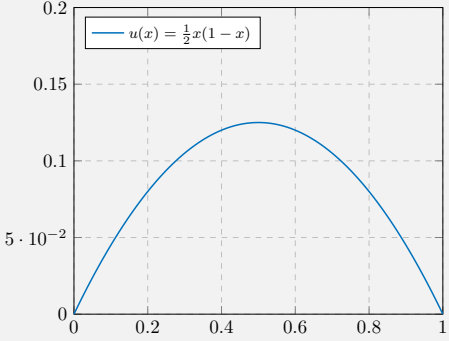
Laplace Problem – One- Vs Two-Level FBPINN

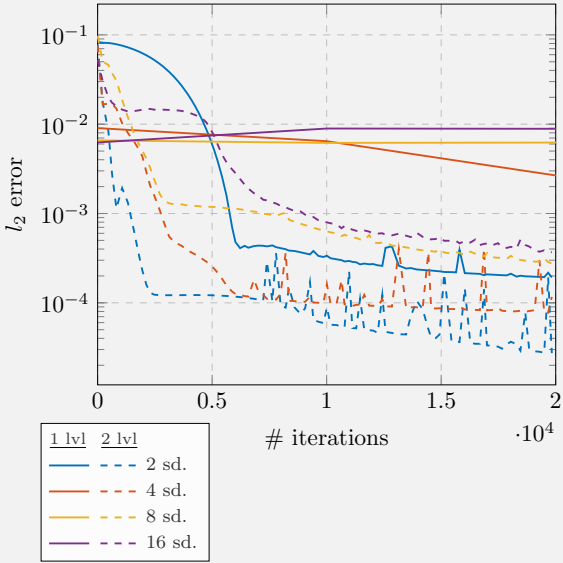
Let us now consider the **simple boundary value problem**

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

which has the **solution**

$$u(x) = \frac{1}{2}x(1-x).$$





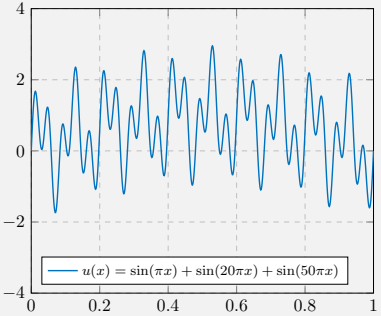
Multi-Frequency Laplace Problem – One- Vs Two-Level FBPINN

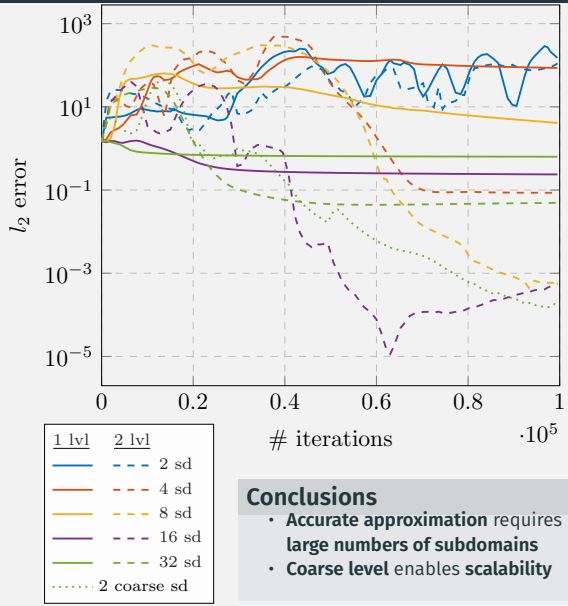
Multi-frequency boundary value problem

$$-u'' = \pi^2 \sin(\pi x) + (20\pi)^2 \sin(20\pi x) + (50\pi)^2 \sin(50\pi x) \text{ in } [0,1],$$
$$u(0) = u(1) = 0,$$

Solution

$u(x) = \sin(\pi x) + \sin(20\pi x) + \sin(50\pi x).$





Conclusions

- **Accurate approximation** requires **large numbers of subdomains**
- **Coarse level** enables **scalability**

Multilevel FBPINN Algorithm

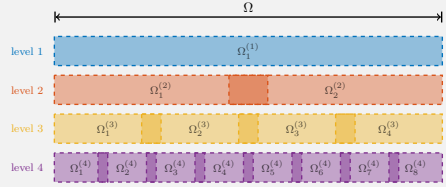
L-level network architecture

Let L **overlapping domain decompositions** $\Omega = \cup_{j=1}^{J^{(l)}} \Omega_j^{(l)}$ and **window functions** $\omega_j^{(l)}$ with

$$\sum_{j=1}^{J^{(l)}} \omega_j^{(l)} \equiv 1, \ \mathrm{supp}(\omega_j^{(l)}) \subset \Omega_j^{(l)}.$$

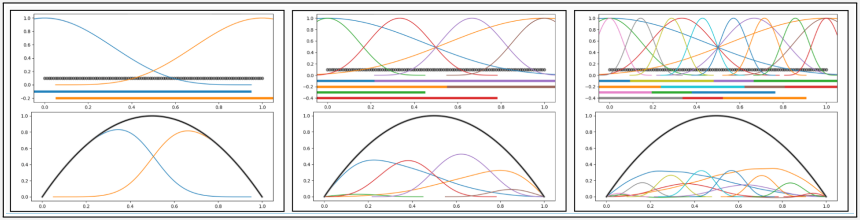
Then the multilevel architecture is

$$u(\mathbf{x}, \boldsymbol{\theta}) = \mathcal{C} \sum_{l=1}^{L} \sum_{j=1}^{J^{(l)}} \omega_j^{(l)} u_j^{(l)}(\mathbf{x}, \boldsymbol{\theta}_j^{(l)})$$



Loss function

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



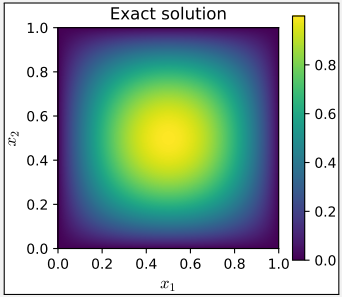
Multilevel FBPINNs – 2D Laplace

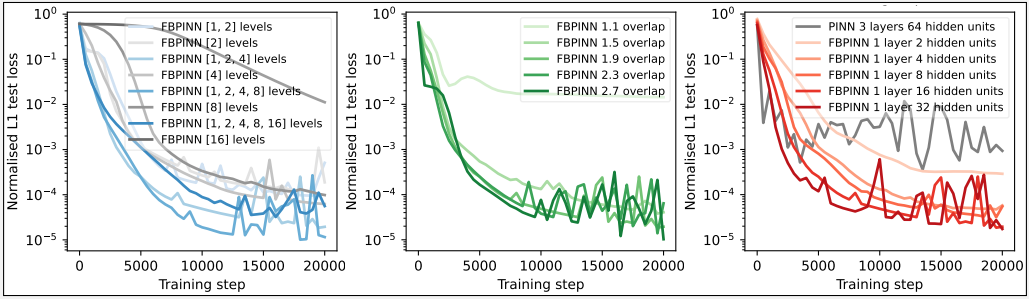
Let us consider the **simple two-dimensional boundary value problem**

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

which has the **solution**

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





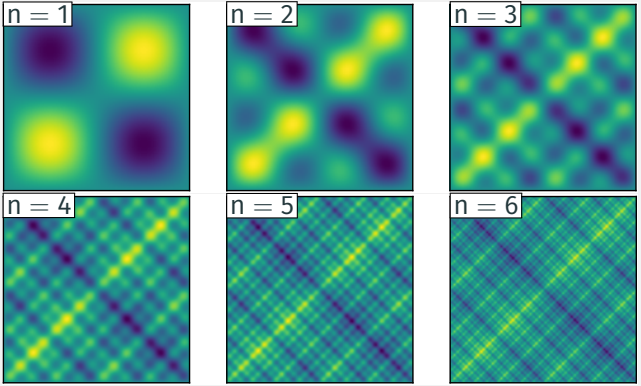
Multi-Frequency Problem

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

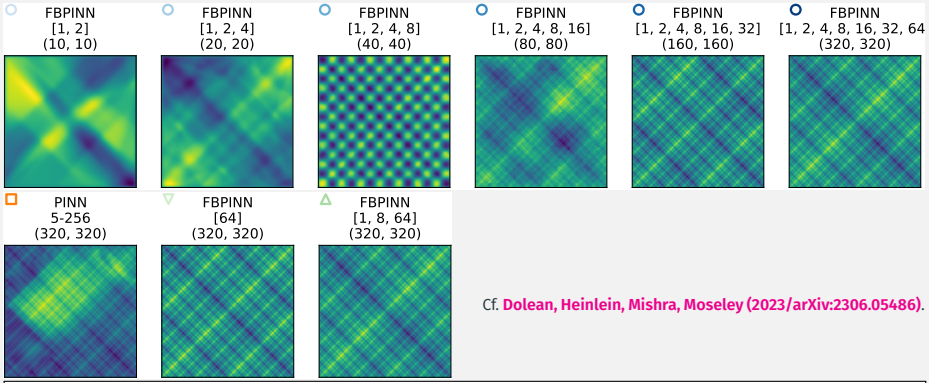
$$\begin{aligned} -\Delta u &= 2\sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi \mathbf{x}) \sin(\omega_i \pi \mathbf{y}) && \text{in } \Omega = [0,1]^2, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

with $\omega_i = 2^i$.

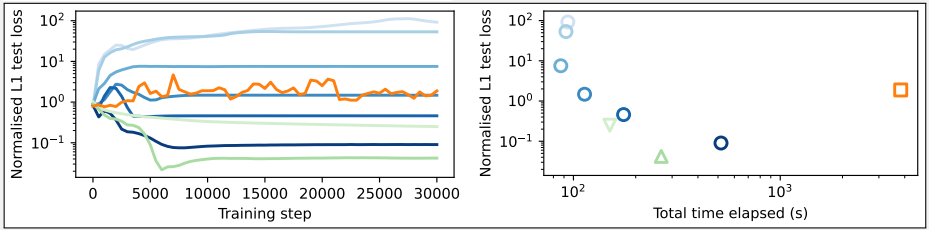
For increasing values of n, we obtain the **analytical solutions**:

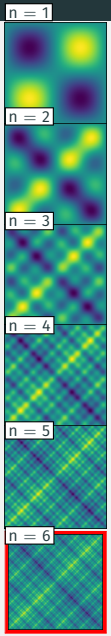


Multi-Level FBPINNs for a Multi-Frequency Problem – Strong Scaling

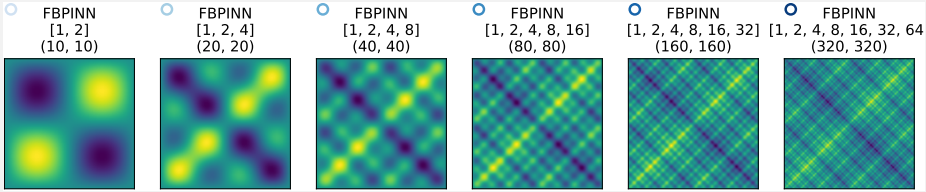


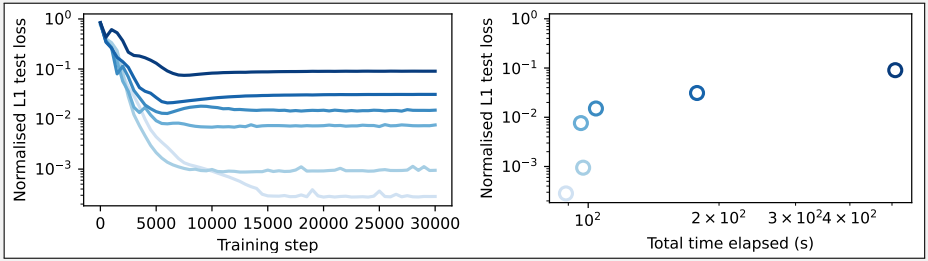
Cf. **Dolean, Heinlein, Mishra, Moseley (2023/arXiv:2306.05486)**.

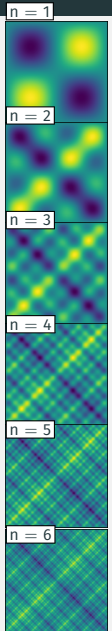




Multi-Level FBPINNs for a Multi-Frequency Problem – Weak Scaling







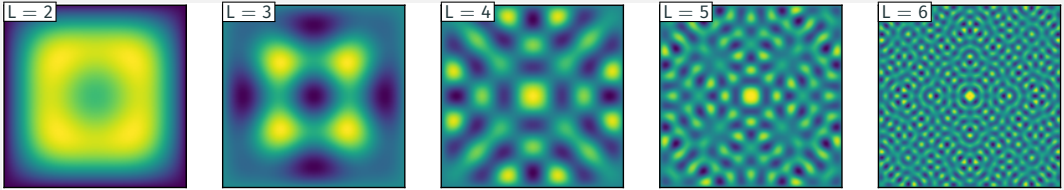
Cf. **Dolean, Heinlein, Mishra, Moseley (2023/arXiv:2306.05486)**.

Helmholtz Problem

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**:



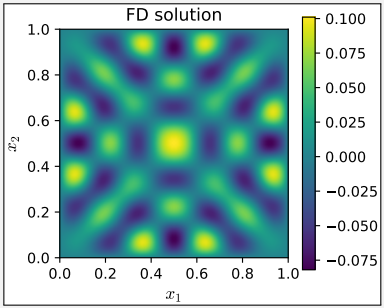
Multilevel FBPINNs – 2D Helmholtz Problem

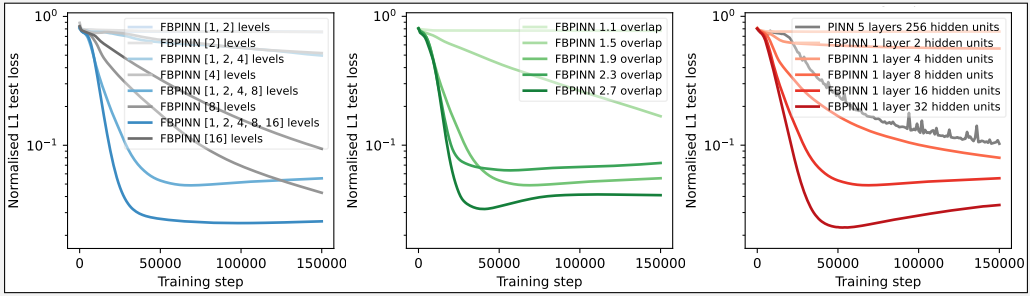
Let us consider the **two-dimensional Helmholtz boundary value problem**

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

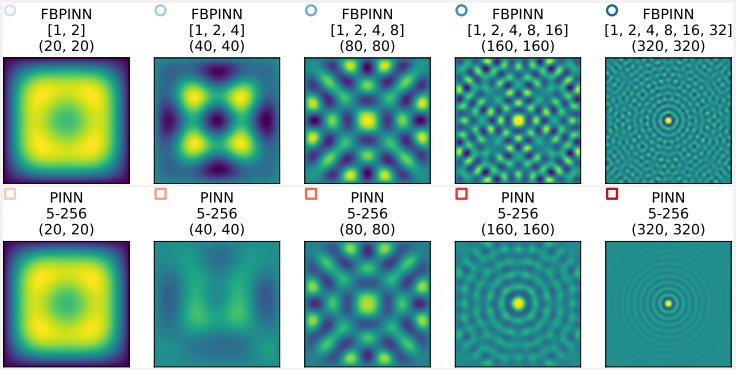
with $k = 2^4\pi/1.6$ and $\sigma = 0.8/2^4$.

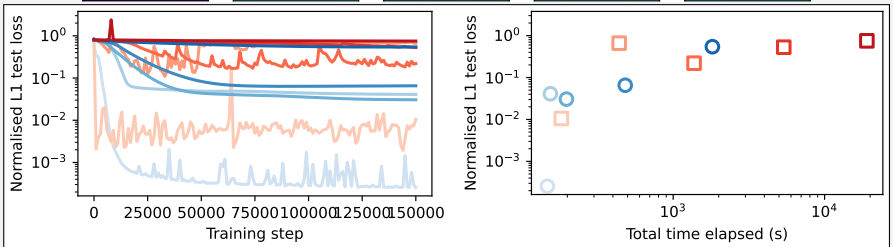
We compute a **reference solution** using **finite differences** with a 5-point stencil on a 320×320 grid.

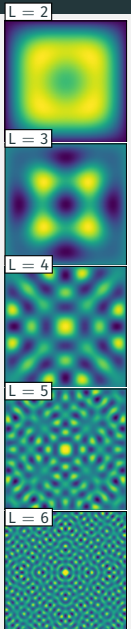




Multi-Level FBPINNs for the Helmholtz Problem – Weak Scaling



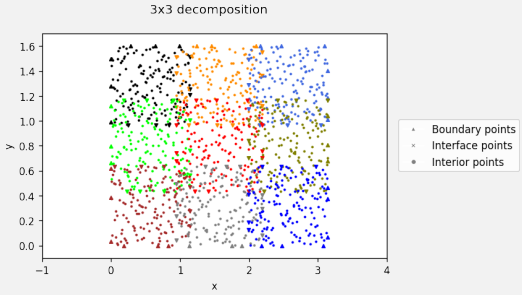




Two-level Deep DDM

Best of two worlds

From classic mesh to sampling



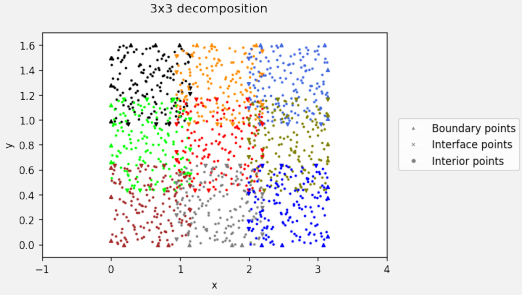
Domain decomposition (overlapping)

- State of the art method for numerical simulation : decompose, iterate and exchange
- Incorporating an interface-related loss term in PINNs:
  - Points on interfaces, $X_\Gamma = \{x_\Gamma^i\}_{i=1}^{N_\Gamma}$, interact with neighbouring networks.
  - The total loss includes a term for the discrepancy at interfaces.

$$\mathcal{M}_\Gamma(\theta) = \frac{1}{N_\Gamma}\sum_{i=1}^{N_\Gamma} |\mathcal{D}(h_s(x_\Gamma^i)) - W_s^i|^2,\ W_s^i = \mathcal{D}(h_r(x_\Gamma^i)).$$

Best of two worlds

From classic mesh to sampling



Deep ddm

Algorithm 1 DeepDDM for the s-th subproblem

1: Sampling of the X_s points
2: Initialization of the network parameters θ_s^0
3: Initialization of information at interfaces W_s^0
4: **while** Non convergence and iteration limits not reached **do**
5: Local network training
6: Update of values at interfaces
7: Network convergence test
8: Interface convergence test
9: **end while**

A scalability problem

- **Strong Scalability :** Strong scalability is defined as how the solution time varies with the number of cores for a fixed total problem size.
- **Weak Scalability :** Weak scalability is defined as how the solution time varies with the number of cores for a fixed problem size per core.

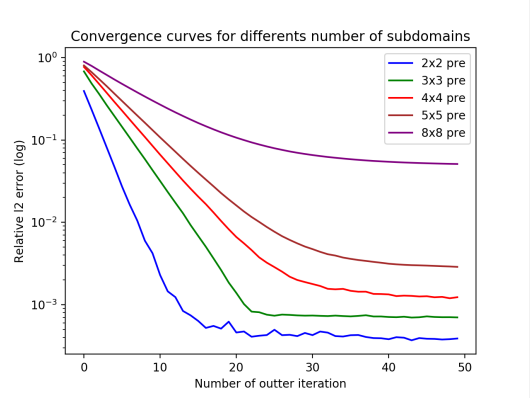


Figure 1: Strong scalability test on Poisson equation

A coarse solution

Using a coarse space in deep-ddm:

- **Coarse Solver** : a sparse sampling over the whole domain, same network size as subdomains network.
  - Compute

$$P = \sum_{s=1}^{S} E_i(\chi_i h_i(x_f^{i,coarse})$$

  - Train as a classical PINN by adding a term imposing P value on X_f^{coarse}

  Information goes from fine to coarse

- **Exchange** : constrain the local (global) solver to a weighted value computed with a coarse (fine) network at a point.

$$W_s^i = \mathcal{D}(\lambda_c \cdot h_r(x_\Gamma^i) + (1 - \lambda_c) \cdot h_{coarse}(x_\Gamma^i))$$

  Information goes from coarse to fine

Algorithm 2 Two-level DeepDDM

1: Sampling of the X_s and X^{coarse} points
2: Initialization of the network parameters θ_s^0 and θ_c
3: Initialization of information at interfaces W_s^0
4: **while** Non convergence and iteration limits not reached **do**
5: Local network training
6: Compute $\sum_{s=1}^{S} E_i(\chi_i h_i(x_f^{i,coarse})$ for each coarse points in X_f^{coarse}
7: Coarse network training
8: Update of values at interfaces
9: Network convergence test
10: Interface convergence test
11: **end while**

A test problem

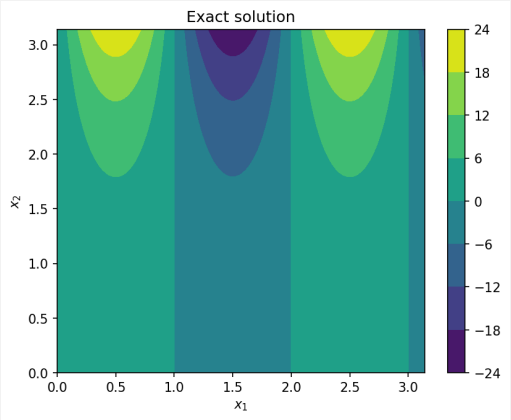
2D Poisson equation

$$\begin{cases} \Delta u = r(x) \text{ in } \Omega = [0, \pi] \times [0, 1] \\ u = g(x) \text{ on } \Gamma \end{cases}$$

We choose r and g to ensure that exact solution is

$$u(z) = \sin(\alpha \pi x_1) e^{x_2}$$

where α is an integer.



Training setup

- Training points are sampled using the Latin hypercube sampling with $N_\Omega = 30000$ and $N_{\partial\Omega} = N_\Gamma = 16000$ for strong scalability
- Training points are sampled using the Latin hypercube sampling with $N_\Omega = 4000$ and $N_{\partial\Omega} = N_\Gamma = 1500$ per subdomain for weak scalability
- All networks use Adam optimizer with an initial learning rate of 2×10^{-4} and exponential decay of 0.999 every 100 epoch.
- Codes implemented in TensorFlow2 (v2.2.0) run on a single NVIDIA GeForce GTX 1080 Ti.
- Each network is composed of two hidden layers with 30 neurons
- The overlap is set to 30% of the subdomain larger side
- Each problem is trained 2500 epochs at each outter iteration

Results

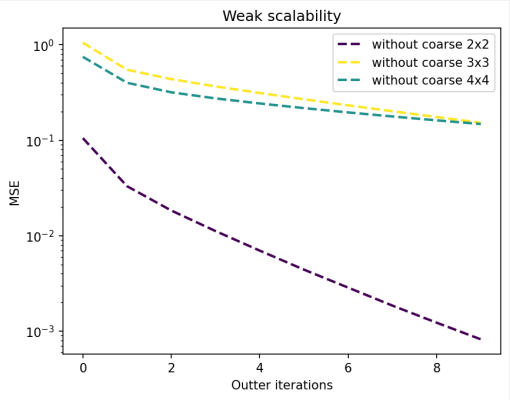


Figure 2: Deep-ddm convergence

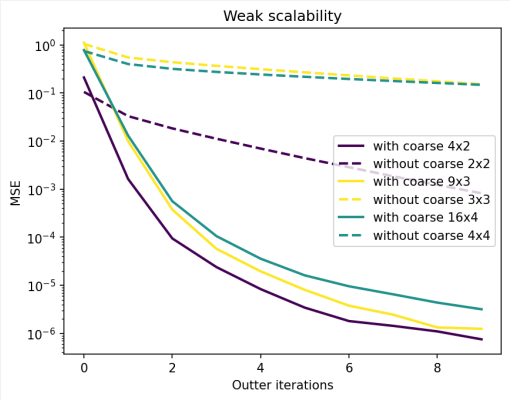


Figure 3: Two-level deep-ddm

2D Poisson equation with frequency content

$$\begin{cases} \Delta u = r(x) \text{ in } \Omega = [0, \pi] \times [0, 1] \\ u = g(x) \text{ on } \Gamma \end{cases}$$

We choose r and g to ensure that exact solution is :

$$u(z) = \sin(w_1 \pi x_1)\sin(w_1 \pi x_2) + \sin(w_2 \pi x_1)\sin(w_2 \pi x_2)$$

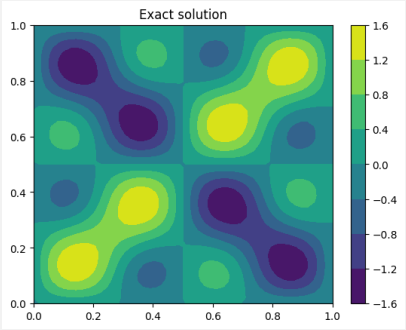
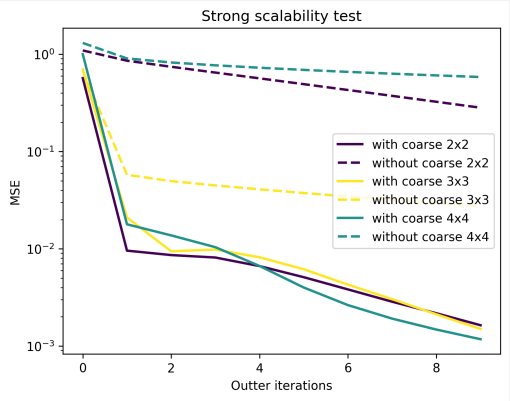
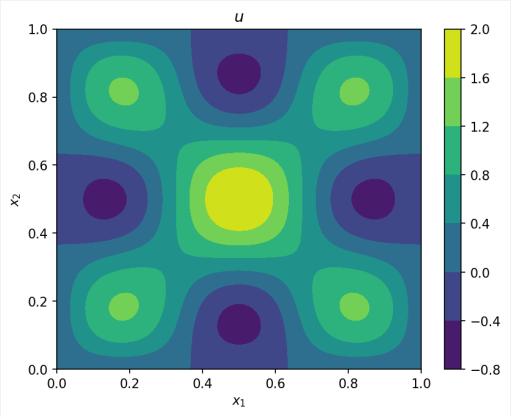


Figure 4: Example with $w_1 = 2$ and $w_2 = 4$

Low frequency content

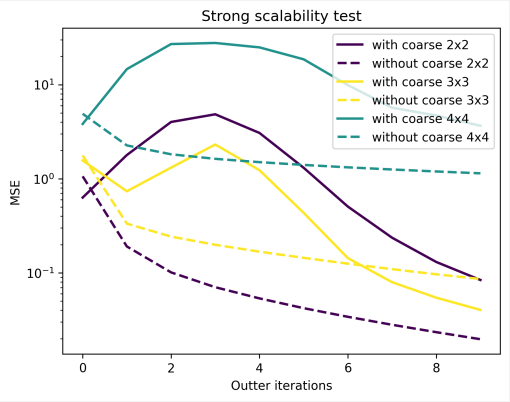
Low frequency test : $w_1 = 1$ **and** $w_2 = 3$

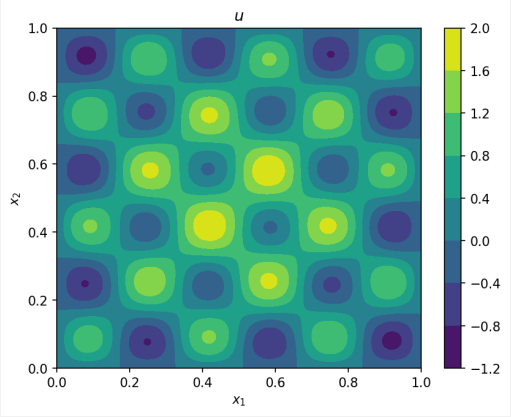




Higher frequency content

Higher frequency test : $w_1 = 1$ **and** $w_2 = 6$





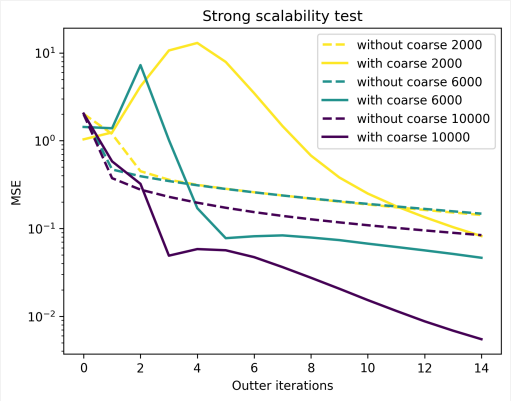
Two-level method doesn't seem to work ⇝ go **multilevel**?

Higher frequency content

Higher frequency test : $w_1 = 1$ **and** $w_2 = 6$

- Vary hyper parameters to solve the problem
- Increase the number of outter iterations: you end up converging faster
- The number of epochs per sub-problem makes it possible to override the F-principle

```
coarse MSE at outter_it 15 : 0.014212156793679516
fine MSE at outter_it 15 : 0.0054742377112197566
```



Conclusions and discussion

Emerging research directions including: two-level Deep DDM and FBPINNs

- Interpretation of FBPINNs from a DD perspective brings the fields of ML and classical DD closer together GitHub: `https://github.com/benmoseley/FBPINNs`
- Multilevel versions have a potential for high wave numbers in an inverse problem setting.
- The multilevel training idea not limited to PINNs. It is also related to the Random Feature Method (RFM), POU Networks, Gated Networks,...
- **Domain decomposition** is a natural idea to ”improve” training (see also Local ELM)
- **BUT** training times are still more important than traditional methods when solving PDEs and **precision is still an issue** $\rightsquigarrow$ preconditioning, multi-GPU training ... more theoretical insight?

PhD position on NLA for ML at TU Eindhoven

Thanks for your attention!