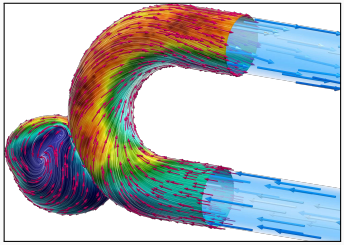
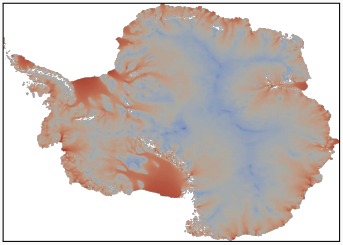
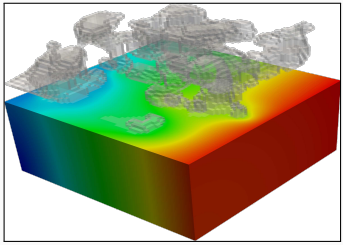
Domain decomposition for physics-informed neural networks

Alexander Heinlein[1]

ANCS Seminar, Laboratoire de Mathématiques, Besançon, France, February 15, 2024

[1]Delft University of Technology

Scientific Machine Learning in Computational Science and Engineering

Numerical methods

Based on physical models

+ Robust and generalizable
− Require availability of mathematical models

Machine learning models

Driven by data

+ Do not require mathematical models
− Sensitive to data, limited extrapolation capabilities

Scientific machine learning (SciML)

Combining the strengths and **compensating the weaknesses** of the individual approaches:

numerical methods	**improve**	machine learning techniques
machine learning techniques	**assist**	numerical methods

Outline

Physics-informed machine learning & motivation

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 $\Psi_t(\boldsymbol{x}, \theta)$ 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**

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**

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

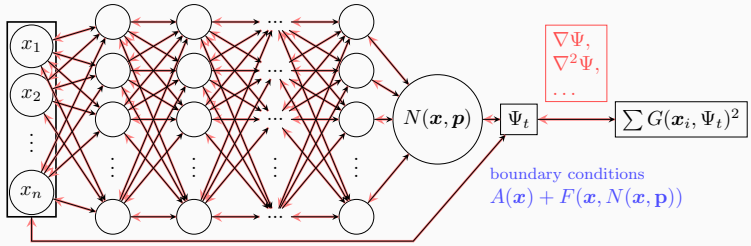
where $\Psi_t(\boldsymbol{x}, \theta)$ is a **trial function**, $\boldsymbol{x}_i$ **sampling points inside the domain** Ω and θ are **adjustable parameters**.

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**



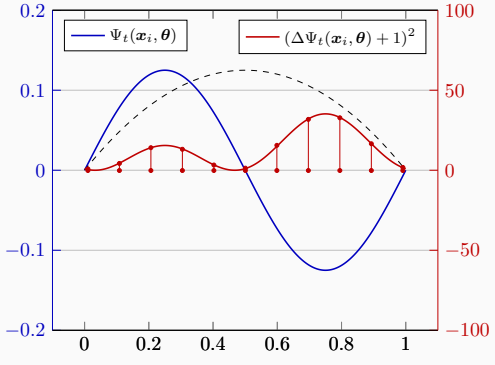
Lagaris et. al's Method – Motivation

Solve the boundary value problem

$$\begin{aligned} \Delta\Psi_t(\boldsymbol{x},\boldsymbol{\theta}) + 1 &= 0 \text{ on } [0,1], \\ \Psi_t(0,\boldsymbol{\theta}) = \Psi_t(1,\boldsymbol{\theta}) &= 0, \end{aligned}$$

via a **collocation approach**:

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

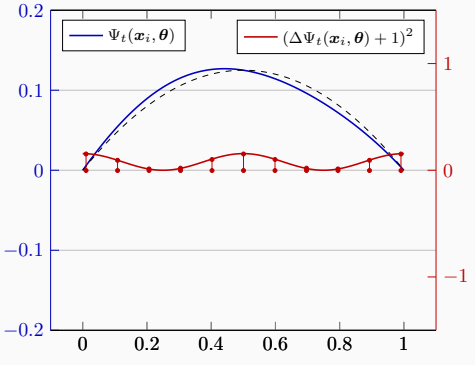


$(\Delta\Psi_t(\boldsymbol{x}_i,\boldsymbol{\theta}) + 1)^2 >> 0$

Boundary conditions

The boundary conditions can be **enforced explicitly**, for instance, via the ansatz:

$$\Psi_t(\boldsymbol{x},\boldsymbol{\theta}) = \sin(\pi\boldsymbol{x}) \cdot F(\boldsymbol{x}, N(\boldsymbol{x},\boldsymbol{\theta})$$



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

Physics-Informed Neural Networks (PINNs)

In the **physics-informed neural network (PINN)** approach introduced by Raissi et al. (2019), a **neural network** is employed to **discretize a partial differential equation**

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

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

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

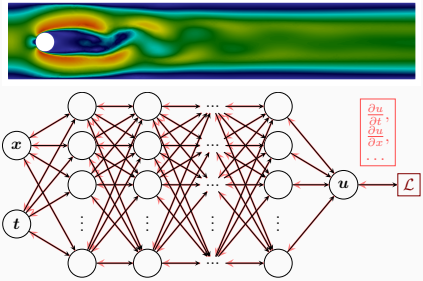
$$\mathcal{L}_{\text{PDE}} \quad = \quad \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} \left(\mathcal{N}[u](\boldsymbol{x}_i, \boldsymbol{t}_i) - f(\boldsymbol{x}_i, \boldsymbol{t}_i) \right)^2.$$

Advantages

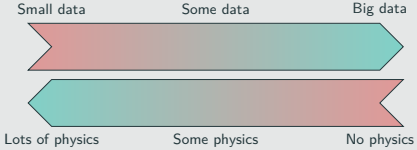
- "Meshfree"
- Small data
- Generalization properties
- High-dimensional problems
- Inverse and parameterized problems

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 (I^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

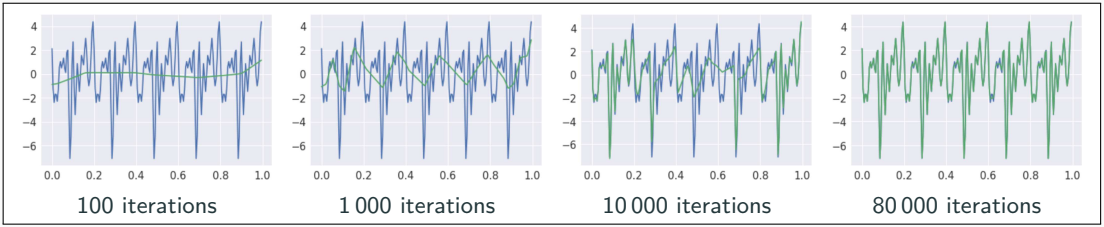
Rule of thumb:

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

Scaling Issues in Neural Network Training

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)**

Motivation – Some Observations on the Performance of PINNs

Solve

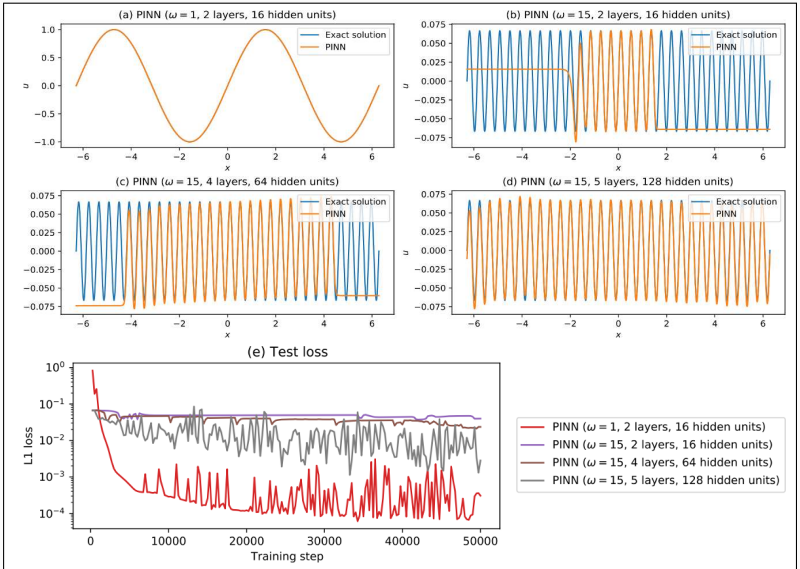
$$\begin{aligned} u' &= \cos(\omega x), \\ u(0) &= 0, \end{aligned}$$

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

Motivation – Some Observations on the Performance of PINNs

Solve

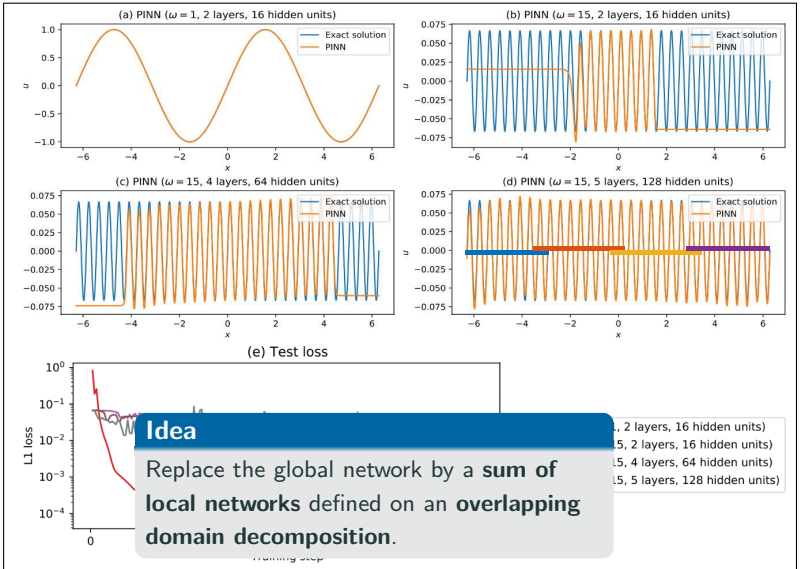
$$\begin{aligned} u' &= \cos(\omega x), \\ u(0) &= 0, \end{aligned}$$

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)

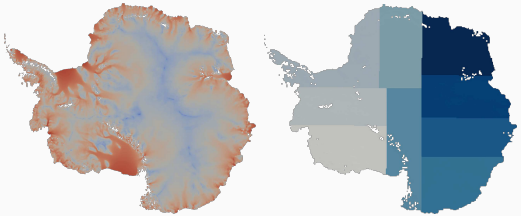


Idea

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

Domain Decomposition Methods



Images based on Heinlein, Perego, Rajamanickam (2022)

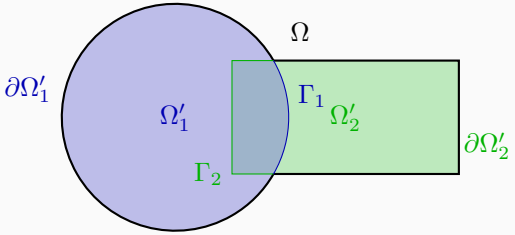
Idea

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

- **Better robustness** and **scalability** of numerical solvers
- **Improved computational efficiency**
- Introduce **parallelism**

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**.



DDM-Based Approaches for Neural Network-Based Discretizations – Literature

A **non-exhaustive overview**:

- **cPINNs**: Jagtap, Kharazmi, Karniadakis (2020)
- **XPINNs**: Jagtap, Karniadakis (2020)
- **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)
- **Schwarz Domain Decomposition Algorithm for PINNs**: Kim, Yang (2022, arXiv 2022)
- **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)

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:

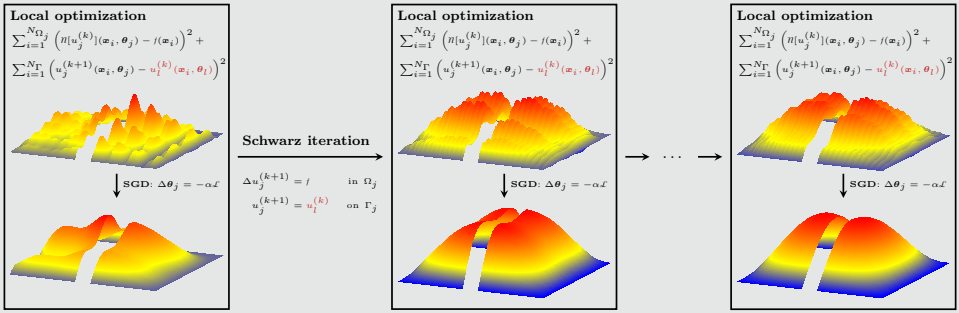
A. Klawonn, M. Lanser, J. Weber

Machine learning and domain decomposition methods – a survey

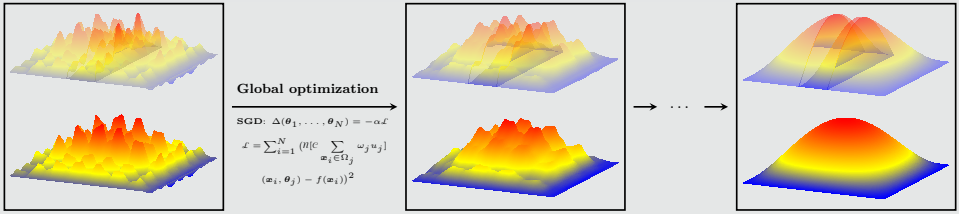
arXiv:2312.14050. 2023

Combining Schwarz Methods with Neural Network-Based Discretizations

Approach 1 – Classical Schwarz iteration



Approach 2 – Via the neural network architecture



Approach 1
Deep learning-based domain decomposition method

Deep Learning-Based Domain Decomposition Method (DeepDDM)

Li, Xiang, Xu. *Deep domain decomposition method: Elliptic problems.* **PMLR (2020)**

DeepDDM for Overlapping Schwarz

In the **DeepDDM method**, we train a **local networks** u_j using local loss functions on subdomain Ω_j

$$\mathcal{L}_j(\boldsymbol{\theta}_j) := \mathcal{L}_{\Omega_j}(\boldsymbol{\theta}_j) + \mathcal{L}_{\partial\Omega_j \setminus \Gamma_j}(\boldsymbol{\theta}_j) + \mathcal{L}_{\Gamma_j}(\boldsymbol{\theta}_j),$$

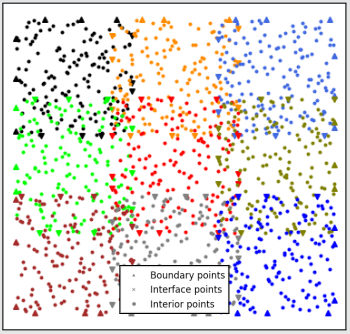
with **volume, boundary, and interface jump terms**

$$\mathcal{L}_{\Omega_j}(\boldsymbol{\theta}_j) := \frac{1}{N_{f_j}} \sum_{i=1}^{N_{f_j}} \left| \mathcal{N}\left(u_j\left(\boldsymbol{x}_{f_j}^i; \boldsymbol{\theta}_j\right)\right) - f\left(\boldsymbol{x}_{f_j}^i\right) \right|^2,$$

$$\mathcal{L}_{\partial\Omega_j \setminus \Gamma_j}(\boldsymbol{\theta}_j) := \frac{1}{N_{g_j}} \sum_{i=1}^{N_{g_j}} \left| \mathcal{B}\left(u_j\left(\boldsymbol{x}_{g_j}^i; \boldsymbol{\theta}_j\right)\right) - g\left(\boldsymbol{x}_{g_j}^i\right) \right|^2,$$

$$\mathcal{L}_{\Gamma_j}(\boldsymbol{\theta}_j) := \frac{1}{N_{\Gamma_j}} \sum_{i=1}^{N_{\Gamma_j}} \left| \mathcal{D}\left(u_j\left(\boldsymbol{x}_{\Gamma_j}^i; \boldsymbol{\theta}_j\right)\right) - \mathcal{D}\left(u_l\left(\boldsymbol{x}_{\Gamma_j}^i; \boldsymbol{\theta}_j\right)\right) \right|^2.$$

Overl. domain decomposition



Algorithm 1: DeepDDM for Ω_j

Data: Sampling points X_j, initial network parameters $\boldsymbol{\theta}_j^0$

while *Convergence (local network & interface values) not reached* **do**

  Train local network u_j;

  Communicate & update interface values $\mathcal{D}\left(u_l(\boldsymbol{x}_{\Gamma_j}^i; \boldsymbol{\theta}_j)\right)$ from other subdomains Ω_l;

end

Numerical Experiments

Strong scaling

Fix the **problem complexity** & **increase** the **model capacity**.

Optimal scaling: improving the convergence rate and/or accuracy at the same rate as the increase of model capacity.

Weak scaling

Increase the **problem complexity** & the **model capacity** at the same rate.

Optimal scaling: constant convergence rate and/or accuracy to stay approximately constant.

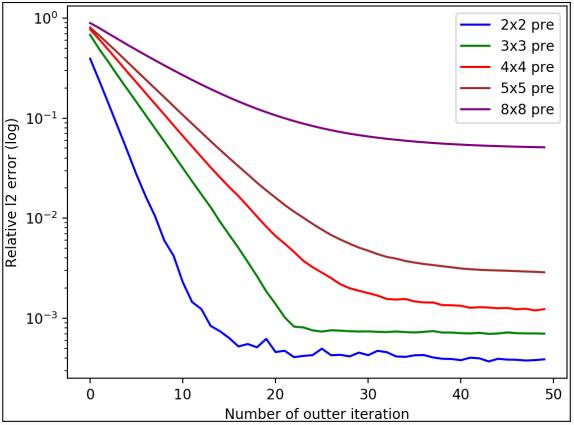
Let first consider a **strong scaling study** for a **two-dimensional Laplacian model problem**:

$$-\Delta u = 1 \text{ in } \Omega$$
$$u = 0 \text{ on } \partial\Omega.$$

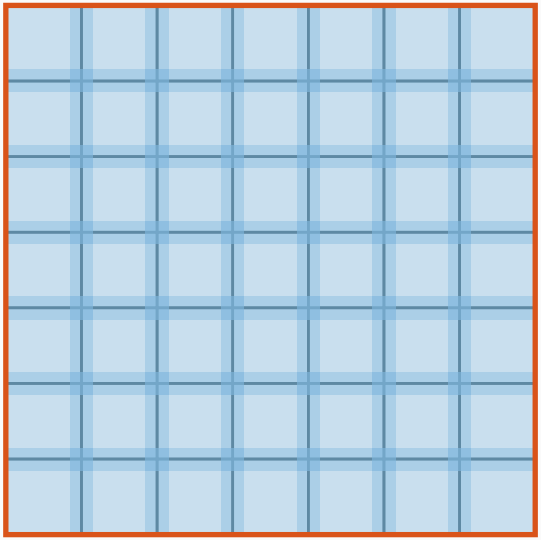
We increase the model capacity by **increasing the number of subdomains**.

Scaling issue

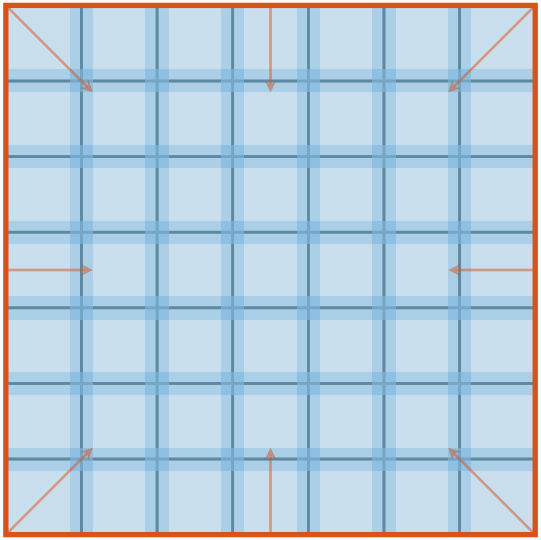
We observe that the performance of the DeepDDM method deteriorates.



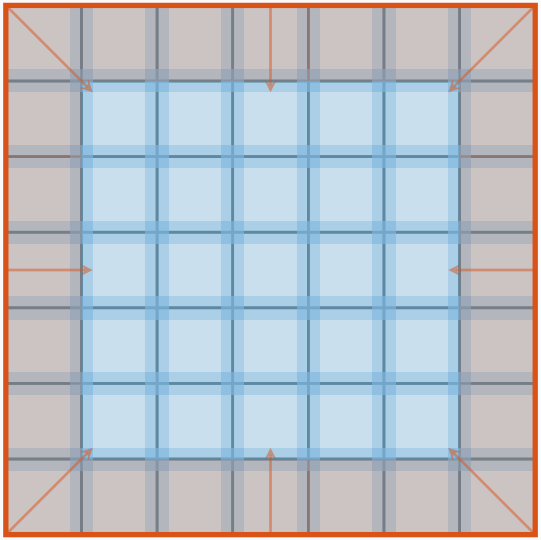
Transport of Information One-Level Overlapping Schwarz Methods



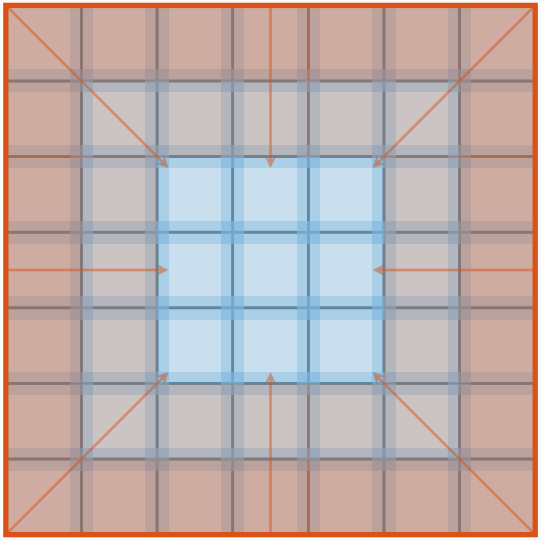
Transport of Information One-Level Overlapping Schwarz Methods



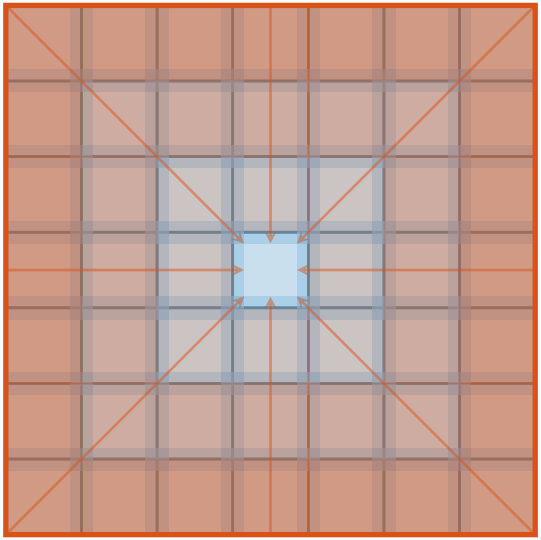
Transport of Information One-Level Overlapping Schwarz Methods



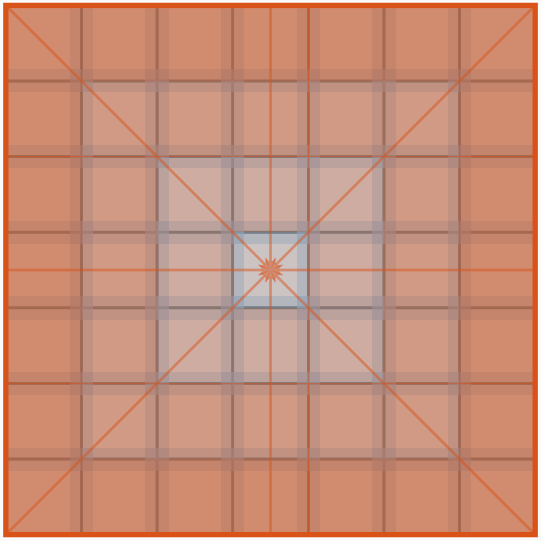
Transport of Information One-Level Overlapping Schwarz Methods



Transport of Information One-Level Overlapping Schwarz Methods



Transport of Information One-Level Overlapping Schwarz Methods



Information (in particular, boundary data) is **only exchanged via the overlapping regions**, leading to **slow convergence** $\rightarrow$ establish a **faster / global transport of information**.

Coarse space for the DeepDDM method

- Sparse sampling $X_f^0 = \left\{x_f^{i,coarse}\right\}_i$ over the whole domain Ω
- Train **coarse network**, that is, a global PINN u_0 with additional loss term

$$\lambda_f \frac{1}{N_0} \sum_{x_f^{i,0} \in X_f^0} \left| u_0(x_f^{i,0}) - \sum_{j=1}^{N} E_j(\chi_j u_j(x_f^{i,0})) \right|^2$$

for **incorporating information from the first level**. Here,

  - E_j extension by zero outside Ω_j
  - χ_j partition of unity function with support in Ω_j

- **Incorporate coarse information** into the loss for the local subdomain Ω_j:

$$\frac{1}{N_{\Gamma_j}} \sum_{i=1}^{N_{\Gamma_j}} \left| \mathcal{D}\left(u_j\left(\boldsymbol{x}_{\Gamma_j}^i; \boldsymbol{\theta}_j\right)\right) - W_j^i \right|^2$$

with $W_f^i = \mathcal{D}(\lambda_c \cdot u_l(x_\Gamma^i) + (1 - \lambda_c) \cdot u_0(x_\Gamma^i))$.

Algorithm 2: Two-level DeepDDM

Data: Sampling points X_j and coarse sampling points X_0, initial network parameters θ_j^0, parameters λ_f and λ_c

while *Convergence (local network & interface values) not reached* **do**

  Train local network u_j;

  Communicate & compute $\sum_{j=1}^{S} E_j(\chi_j u_j(x_f^{i,coarse}))$ for each coarse points in X_f^{coarse};

  Train coarse network u_0;

  Communicate & update interface values $\mathcal{D}\left(u_l(\boldsymbol{x}_{\Gamma_j}^i; \boldsymbol{\theta}_j)\right)$ from other subdomains Ω_l;

  Update λ_f and λ_c;

end

2D Poisson Equation – Problem Setup

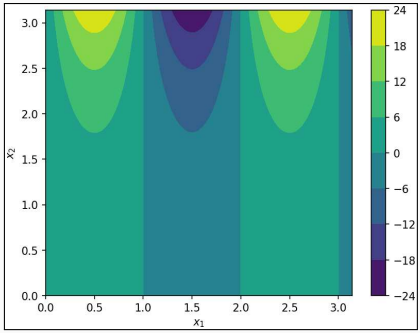
Model problem:

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

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.

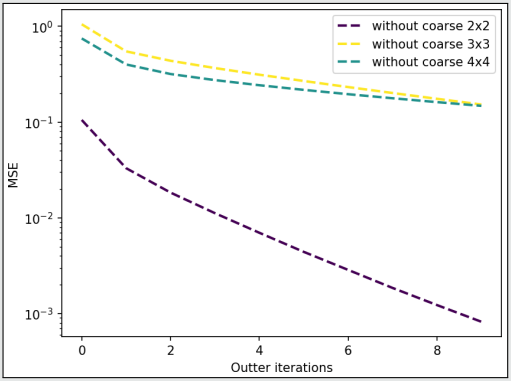


Training setup

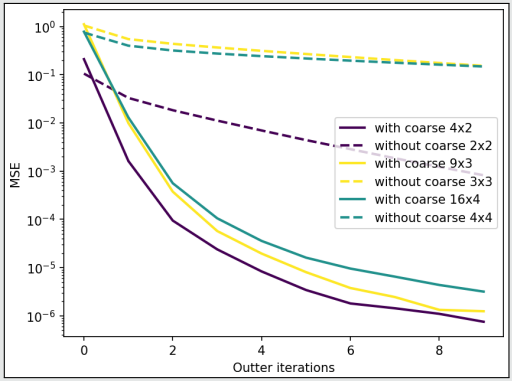
- **Strong scaling**: Latin hypercube sampling for training points with $N_\Omega = 30\,000$ and $N_{\partial\Omega} = N_\Gamma = 16\,000$.
- **Weak scaling**: Latin hypercube sampling for training points with $N_\Omega = 4\,000$ and $N_{\partial\Omega} = N_\Gamma = 1\,500$ per subdomain.
- Each network is composed of two hidden layers with 30 neurons
- **Optimization** of local/coarse networks: 2500 epochs using the Adam optimizer with initial learning rate $2{\cdot}10^{-4}$ and exp. decay of 0.999 every 100 epochs.
- Codes implemented in TensorFlow2 (v2.2.0) run on a single NVIDIA GeForce GTX 1080 Ti.
- The overlap is set to 30% of the subdomain larger side

2D Poisson Equation – Weak Scaling

One-level DeepDDM



Two-level DeepDDM



→ **Adding a coarse level fixes the issue.**

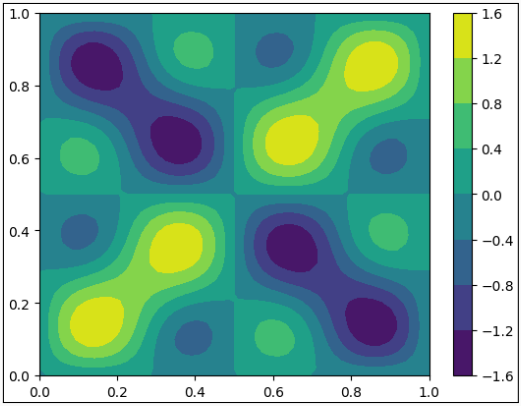
2D Poisson Equation With Variable Frequency

Model problem:

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

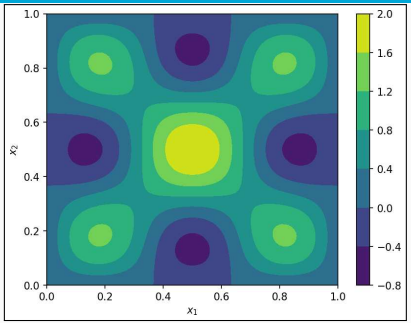
We choose r and g to ensure that exact solution is:

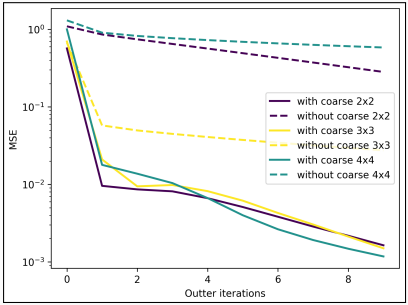
$$\begin{aligned} u(z) &= sin(w_1 \pi x_1) sin(w_1 \pi x_2) \\ &\quad + sin(w_2 \pi x_1) sin(w_2 \pi x_2) \end{aligned}$$



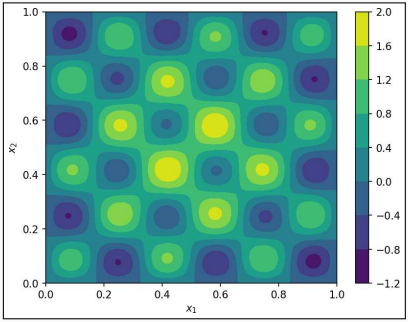
2D Poisson Equation With Variable Frequency – Weak Scaling

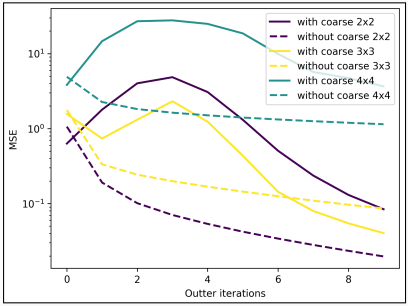
Low frequency test:
$w_1 = 1$ **and** $w_2 = 3$





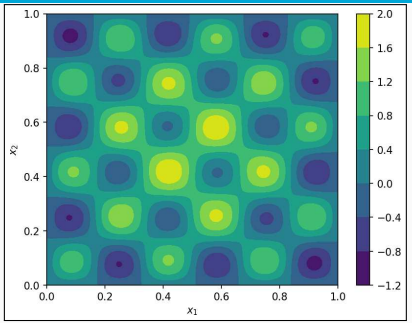
Higher frequency test:
$w_1 = 1$ **and** $w_2 = 6$

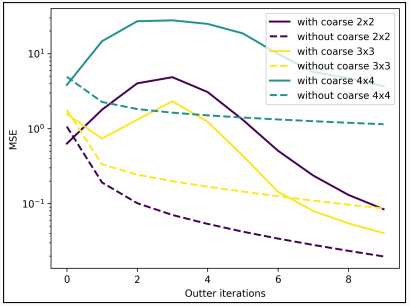




2D Poisson Equation With Variable Frequency – Weak Scaling

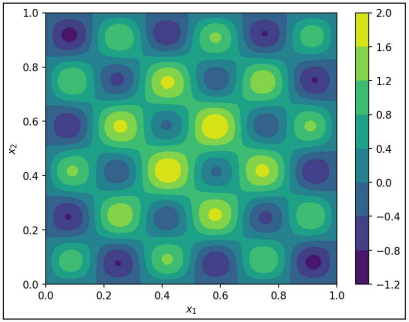
Higher frequency test:
$w_1 = 1$ **and** $w_2 = 6$

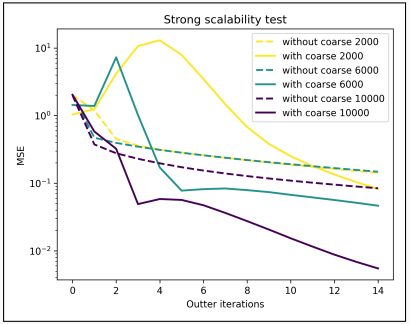




Hyper parameter tuning

- ↑ # epochs for each sub problem
- ↑ # outer Schwarz iterations





DeepDDM for Helmholtz Problems

Li, Wang, Cui, Xiang, Xu. *Deep domain decomposition method: Helmholtz equation*. **Advances in Applied Mathematics and Mechanics (2023)**

Train **local networks** u_s by transmitting only Robin interface values to the neighboring subdomains.

Use of NNs with **plane wave (PW) activation** to account for oscillatory nature of the solution. The loss function reads

$$\mathcal{L}_s\left(\theta_s; \boldsymbol{X}_s\right) := \mathcal{L}_{\Omega_s}\left(\theta_s; \boldsymbol{X}_s\right) + \mathcal{L}_{\partial\Omega_s \setminus \Gamma_s}\left(\theta_s; \boldsymbol{X}_s\right) + \mathcal{L}_{\Gamma_s}\left(\theta_s; \boldsymbol{X}_s\right),$$

where

$$\mathcal{L}_{\Omega_s}\left(\theta_s; \boldsymbol{X}_{f_s}\right) := \frac{1}{N_{f_s}} \sum_{i=1}^{N_{f_s}} \left| \mathcal{H}\left(u_s\left(\boldsymbol{x}_{f_s}^i; \theta_s\right)\right) - f\left(\boldsymbol{x}_{f_s}^i\right) \right|^2,$$

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

$$\mathcal{L}_{\Gamma_s}\left(\theta_s; \boldsymbol{X}_{\Gamma_s}\right) := \frac{1}{N_{\Gamma_s}} \sum_{i=1}^{N_{\Gamma_s}} \left| \frac{\partial u_s\left(\boldsymbol{x}_{g_{\Gamma_s}}^i; \theta_s\right)}{\partial \boldsymbol{n}_s} + \gamma_s u_s\left(\boldsymbol{x}_{\Gamma_s}^i; \theta_s\right) - g_s\left(\boldsymbol{x}_{\Gamma_s}^i\right) \right|^2.$$

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 an iterative Schwarz method – what about Krylov acceleration?
- All relies on PW activation function – **number of parameters dependent on** k.

Approach 2
Multilevel domain decomposition-based architectures for physics-informed neural networks

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**

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

and the **loss function**

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

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

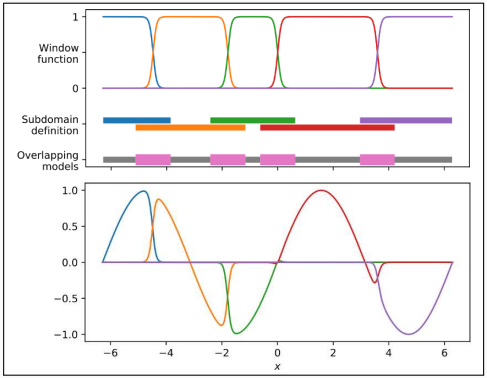
Hard enforcement of boundary conditions

Loss function

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

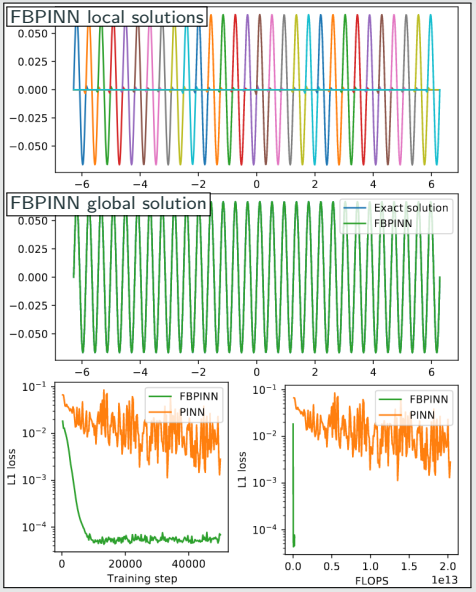
with constraining operator $\mathcal{C}$, which **explicitly enforces the boundary conditions**.

→ Often **improves training performance**



Numerical Results for FBPINNs

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



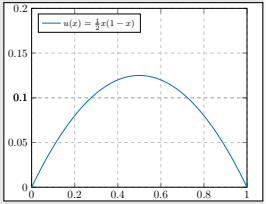
Scalability of FBPINNs

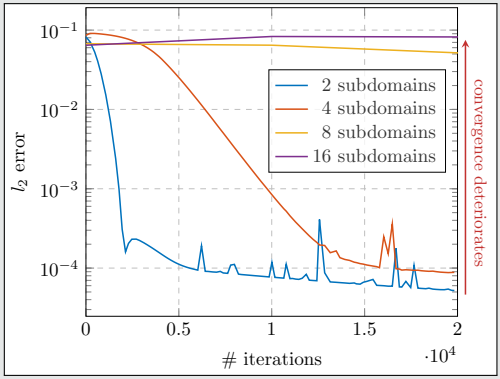
Consider the **simple boundary value problem**

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

which has the **solution**

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





Multi-Level FBPINN Algorithm

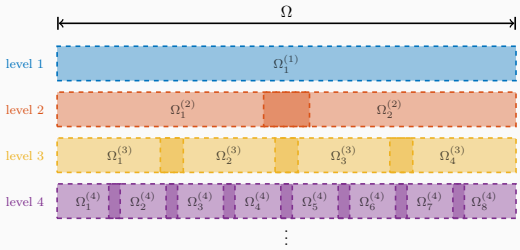
We introduce a **hierarchy of L overlapping domain decompositions**

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

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

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

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

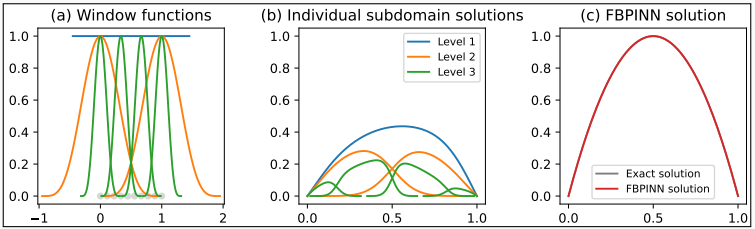


L-level network architecture

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

Loss function

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



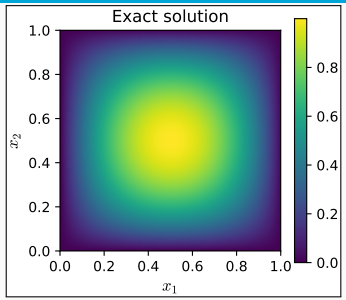
Multilevel FBPINNs – 2D Laplace

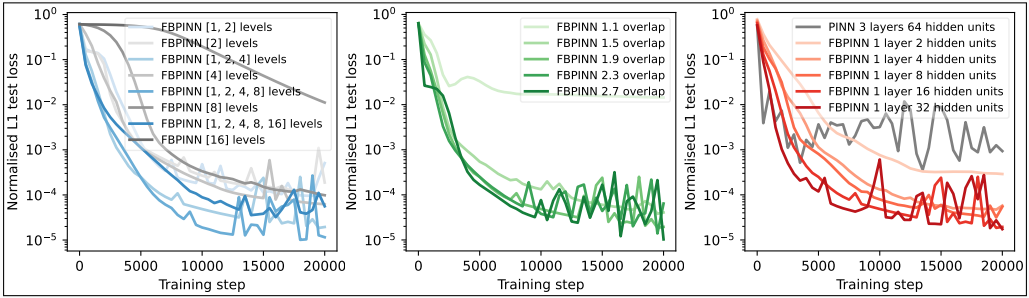
Let us consider the **simple two-dimensional boundary value problem**

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

which has the **solution**

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





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

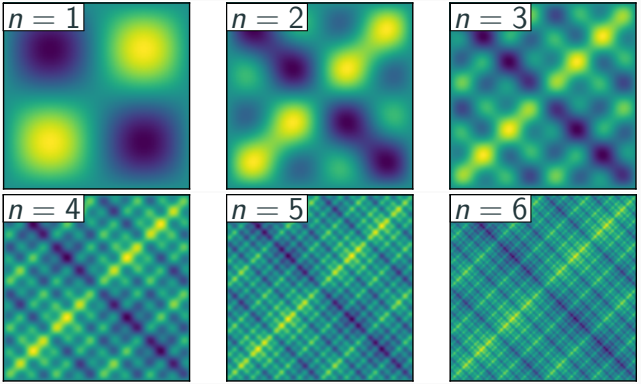
Multi-Frequency Problem

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

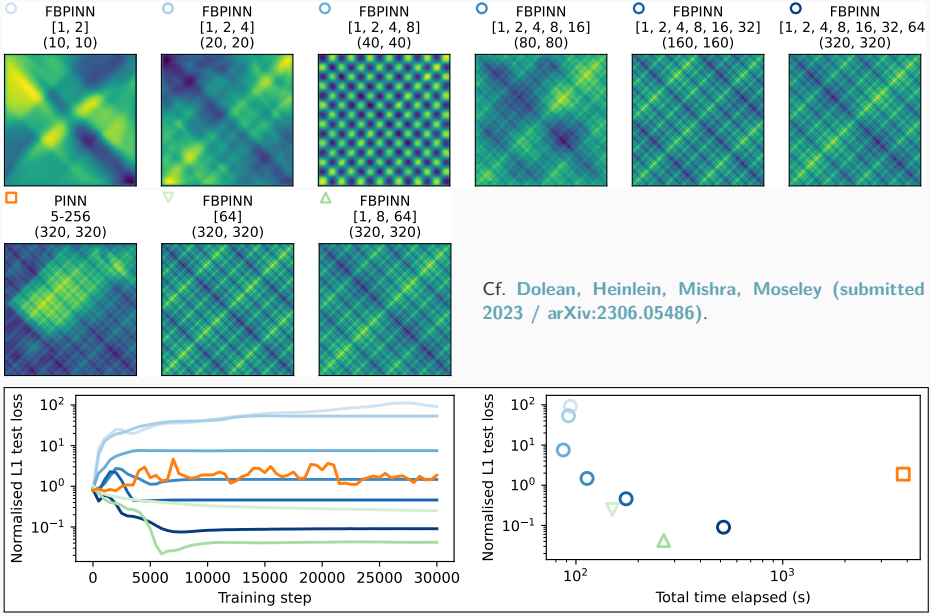
$$
\begin{aligned}
-\Delta u &= 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi \boldsymbol{x}) \sin(\omega_i \pi \boldsymbol{y}) && \text{in } \Omega = [0,1]^2, \\
u &= 0 && \text{on } \partial\Omega,
\end{aligned}
$$

with $\omega_i = 2^i$.

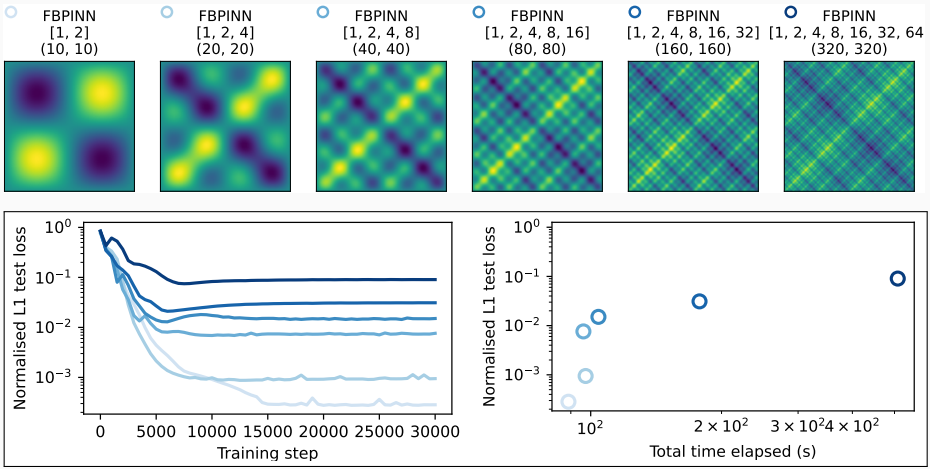
For increasing values of n, we obtain the **analytical solutions**:

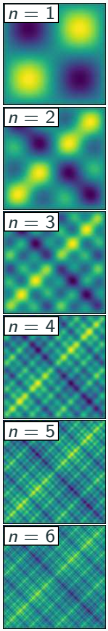


Multi-Level FBPINNs for a Multi-Frequency Problem – Strong Scaling



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





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

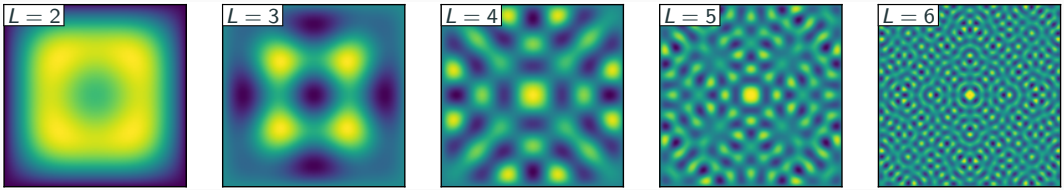
Cf. **Dolean, Heinlein, Mishra, Moseley (submitted 2023 / arXiv:2306.05486)**.

Helmholtz Problem

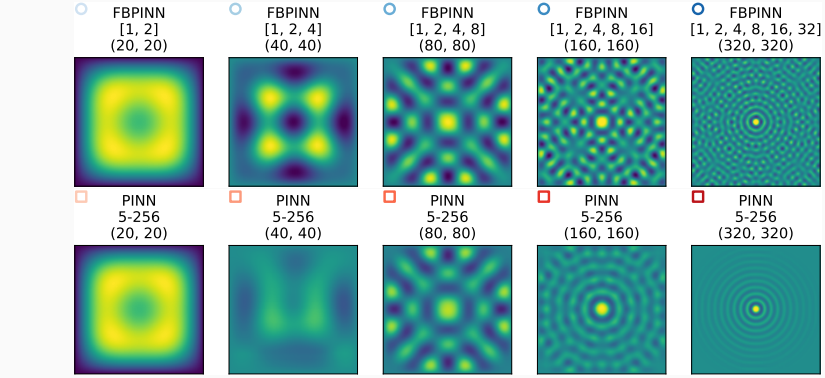
Finally, let us consider the **two-dimensional Helmholtz boundary value problem**

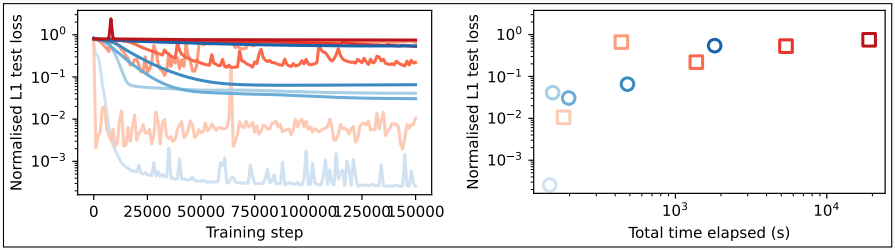
$$\begin{aligned} \Delta u - k^2 u &= f \quad \text{in } \Omega = [0,1]^2, \\ u &= 0 \quad \text{on } \partial\Omega, \\ f(\mathbf{x}) &= e^{-\frac{1}{2}(\|\mathbf{x}-0.5\|/\sigma)^2}. \end{aligned}$$

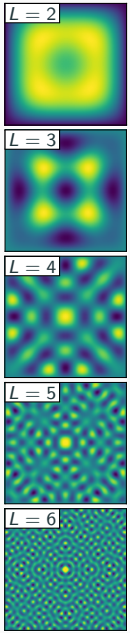
With $k = 2^L\pi/1.6$ and $\sigma = 0.8/2^L$, we obtain the **solutions**:



Multi-Level FBPINNs for the Helmholtz Problem – Weak Scaling





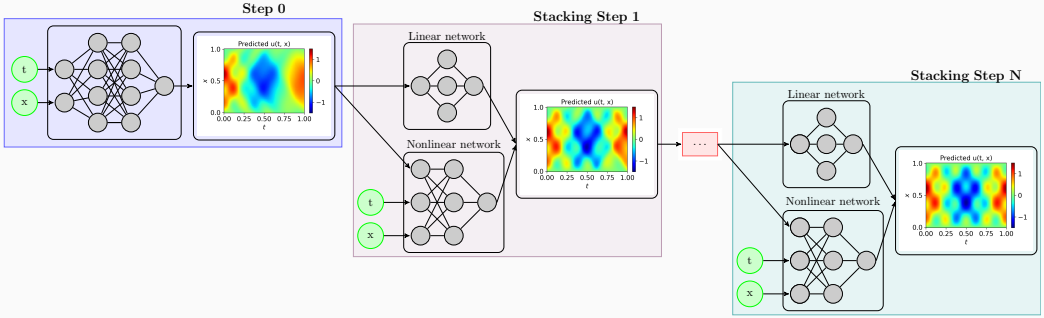


Multifidelity domain decomposition-based physics-informed neural networks for time-dependent problems

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

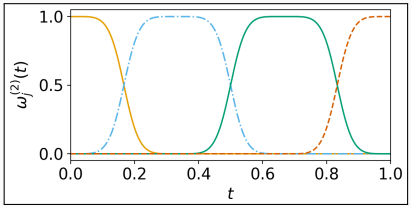
We **combine stacking multifidelity PINNs with FBPINNs** by using an FBPINN model (with an increasing number of subdomains) in each stacking step. → **One-way sequential coupling** of the levels Cf. **Heinlein, Howard, Beecroft, Stinis (subm. 2024 / arXiv:2401.07888)**

Numerical Results – Pendulum Problem

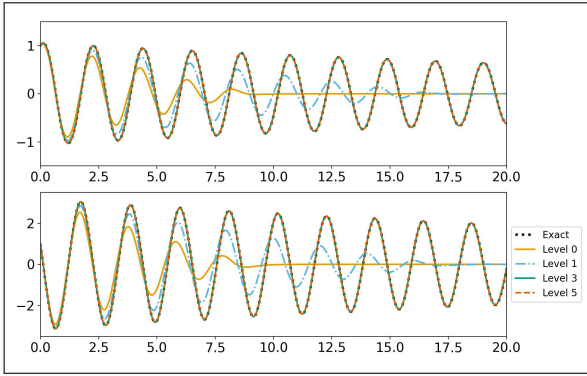
First, we consider a **pedulum problem** and **compare the stacking multifidelity PINN and FBPINN** approaches:

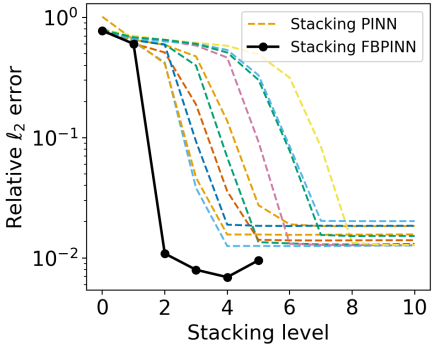
$$\frac{ds_1}{dt} = s_2,$$
$$\frac{ds_2}{dt} = -\frac{b}{m}s_2 - \frac{g}{L}\sin(s_1)$$

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



Exemplary partition of unity in time



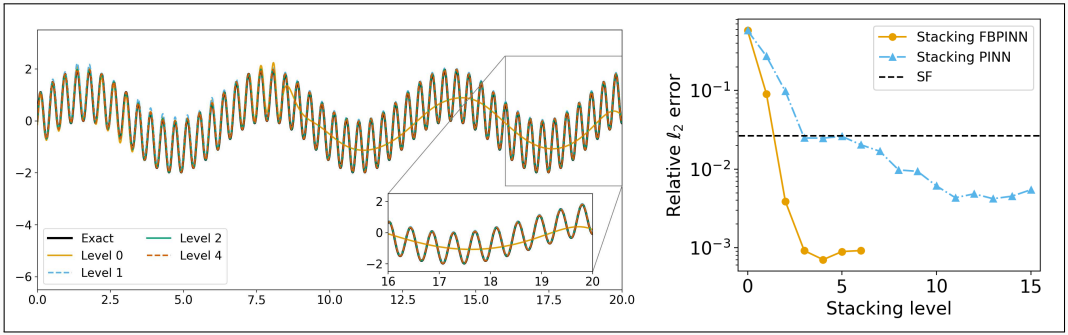


Numerical Results – Two-Frequency Problem

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 $\omega_2 = 15$.

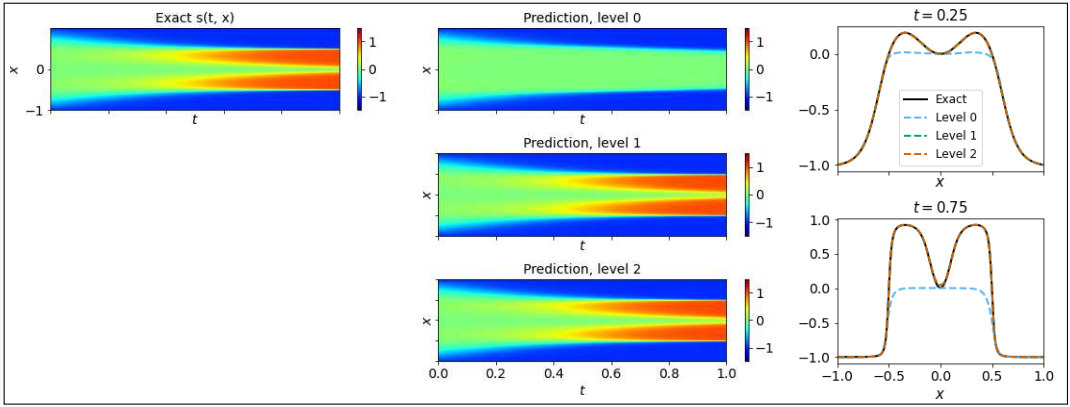


→ Due to the **multiscale structure of the problem**, the **improvements** due to the **multifidelity FBPINN approach** are **even stronger**.

Numerical Results – Allen–Cahn Equation

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

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



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**

DeepDDM for PINNs

- The **DeepDDM method** is a **classical Schwarz iteration** with local PINN solver.
- **Scalability** is enabled by **adding a coarse level**.

(Multilevel) 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**.

Multifidelity stacking FBPINNs

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

Thank you for your attention!

Workshop on Computational and Mathematical Methods in Data Science 2024

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/