# From Classical Domain Decomposition Solvers to Learning

**Victorita Dolean**

with: A. Heinlein, S. Mishra, B. Moseley

LJLL French-Spanish summer school, 7-11 July 2025



A. Heinlein, B. Moseley, S. Mishra

# 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** $\Omega$ and $\mathbf{p}$ are **adjustable parameters**.

# Main Features of the Method

## Construction of the trial functions

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

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

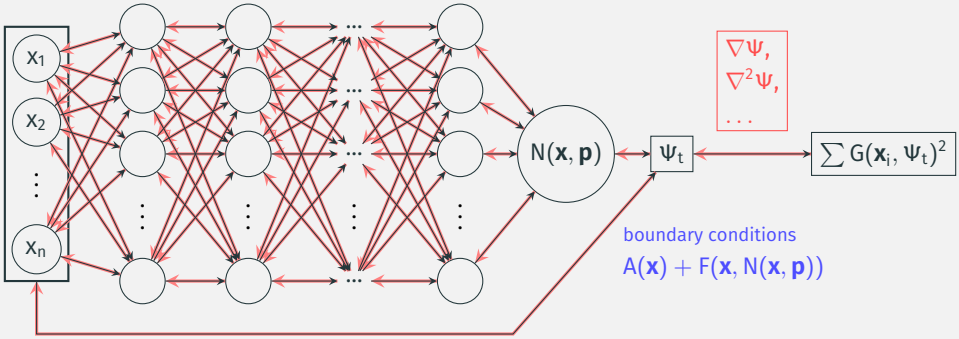
where

- $N(\mathbf{x}, \mathbf{p})$ 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

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

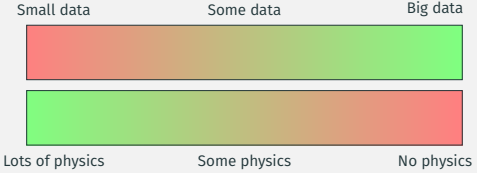
The main novelty of PINNs is that a **hybrid loss function** is used for training the feedforward neural network:

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

where $\omega_{data}$ and $\omega_{PDE}$ are **weights** and

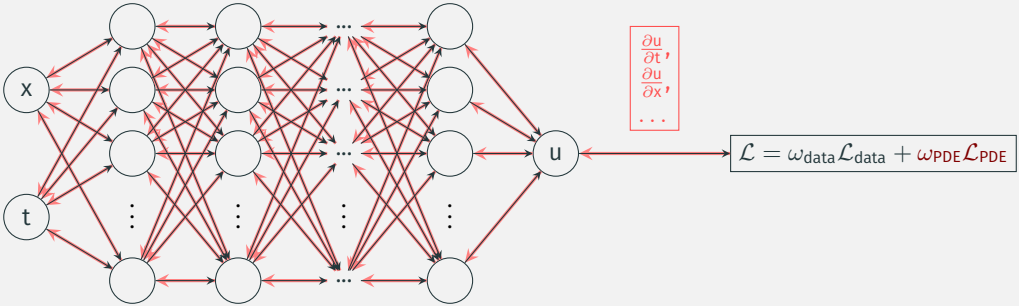
$$\mathcal{L}_{data} = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} \left(u(x_i, t_i) - u_i\right)^2,$$

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




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

# Sketch of the PINN approach by Raissi et al.



# 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

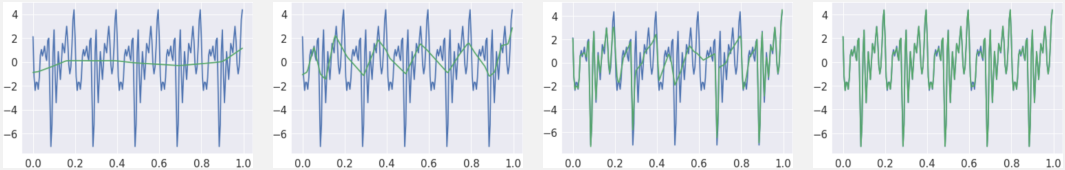
- $\mathcal{E}_G = \mathcal{E}_G(\theta^*; \mathcal{S}) := \|\mathbf{u} - \mathbf{u}^*\|_X$
- $\mathcal{E}_T$ is the training error
- $C_{pde}$ and $C_{quad}$: constants depending on the PDE and the quadrature
- N: number of the training points

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

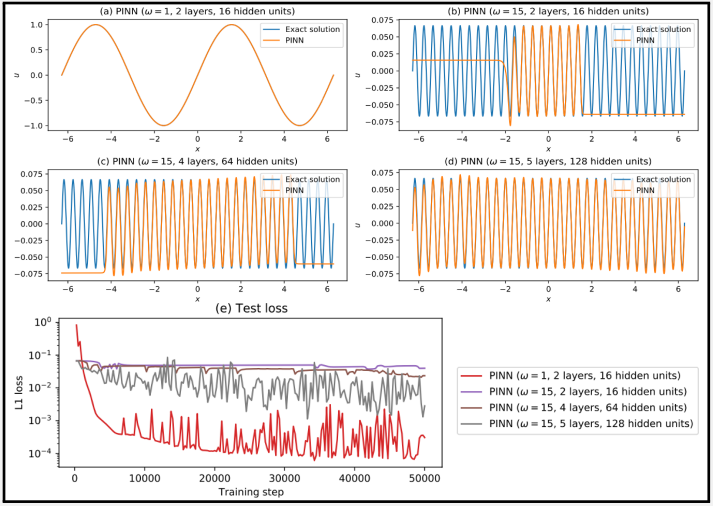
Solve

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

for different values of $\omega$.

**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)**

# 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

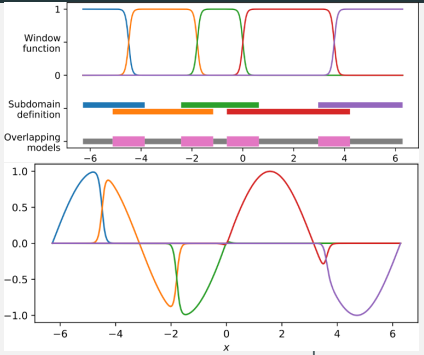
Loss function

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

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 $\Omega_j^\circ$ and $\Omega_j^{\text{int}}$
- 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

# FBPINN With Flexible Scheduling

FBPINN training step

**if** $j \in \mathcal{A}$ ($\Omega_j$ 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.

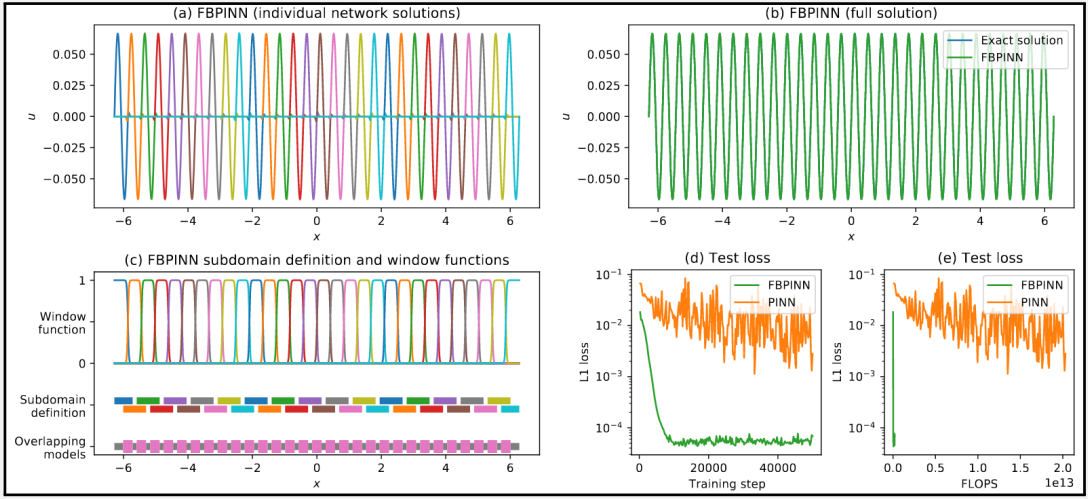
Apply the constraining operator $u(\mathbf{x}, \boldsymbol{\theta}_j^k) \leftarrow \mathcal{C}(u(\mathbf{x}, \boldsymbol{\theta}_j^{k+p}))$.

**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:**

$\rightsquigarrow$ Learn low frequencies using a small **global network**, train high frequencies using local networks.

Two-level FBPINN **network architecture**:

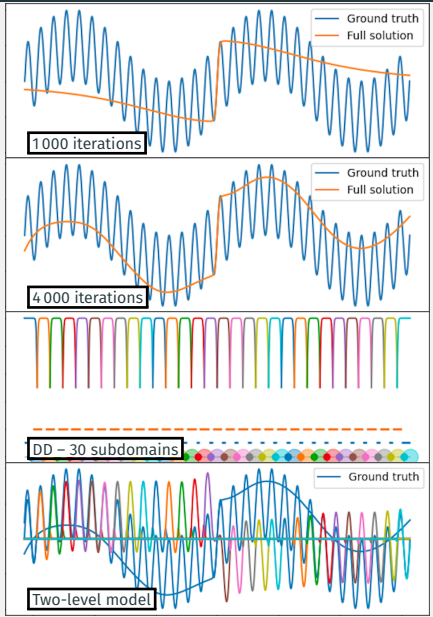
$$u(\theta_0, \theta_1, \dots, \theta_J) = \mathcal{C}\left(\mathbf{u}_0(\theta_l) + \sum_{\Omega_j} \omega_l \mathbf{u}_l(\theta_l)\right)$$

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**



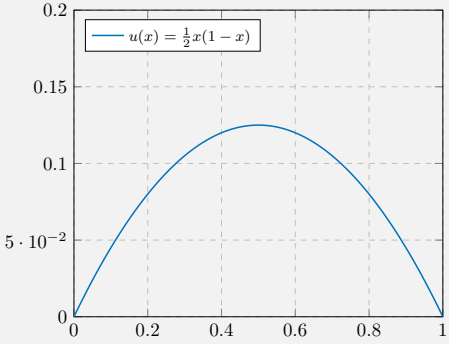
# Laplace Problem – One- Vs Two-Level FBPINN

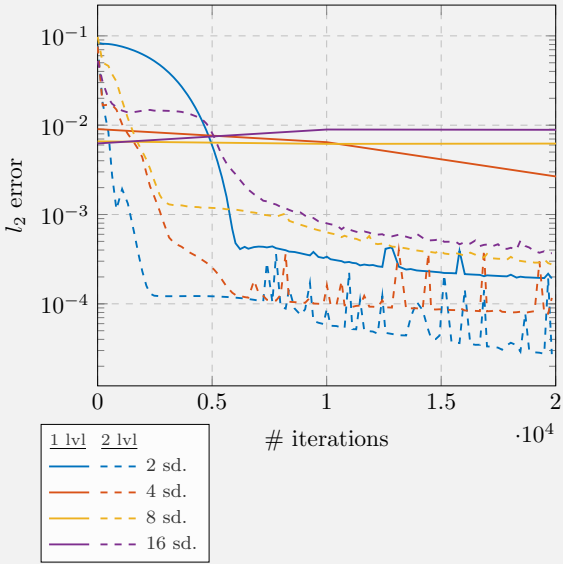
Let us now consider the **simple boundary value problem**

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

which has the **solution**

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



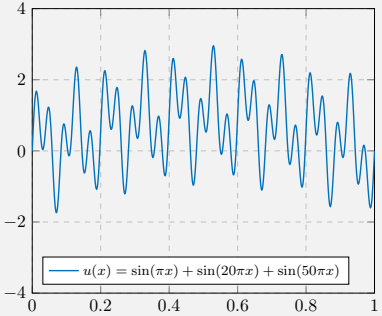
# Multi-Frequency Laplace Problem – One- Vs Two-Level FBPINN

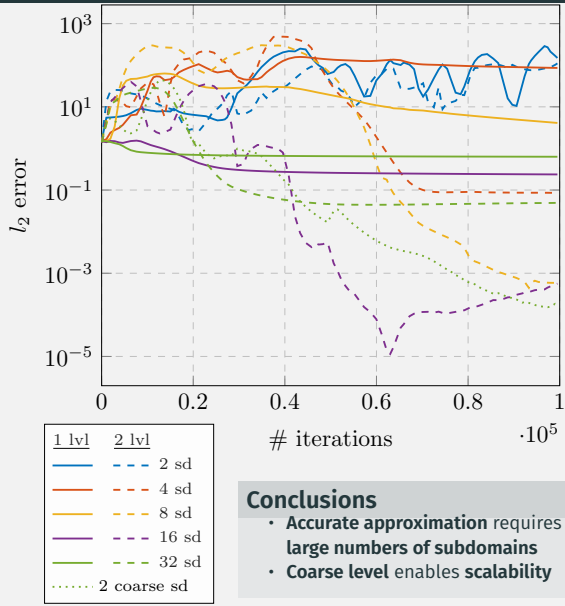
**Multi-frequency boundary value problem**

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

**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

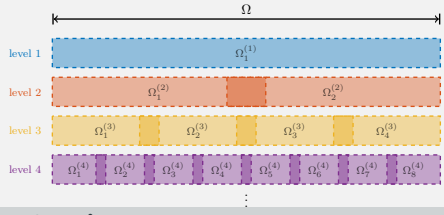
## L-level network architecture

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

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

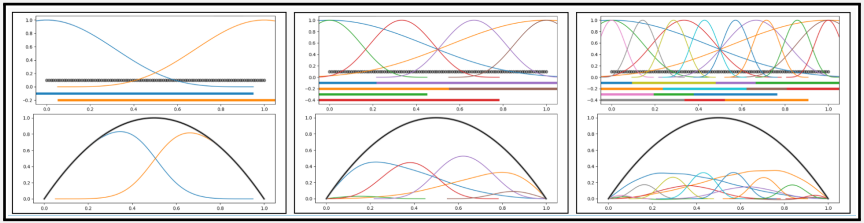
Then the multilevel architecture is

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




## Loss function

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



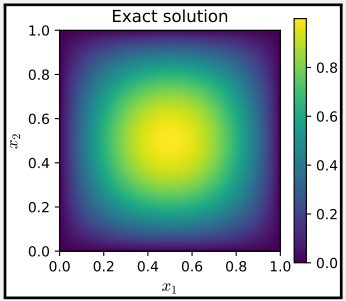
# Multilevel FBPINNs – 2D Laplace

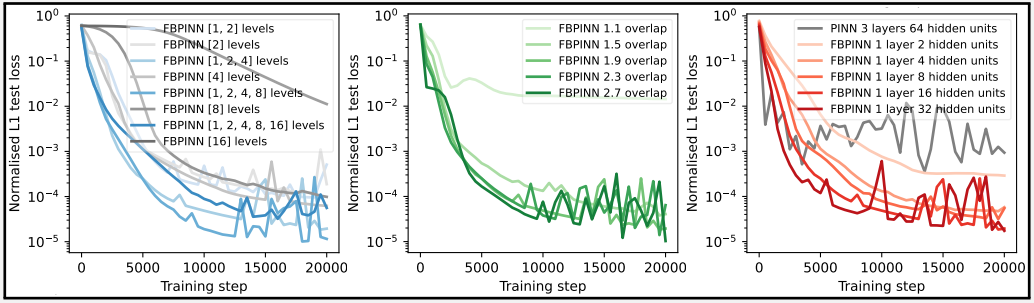
Let us consider the **simple two-dimensional boundary value problem**

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

which has the **solution**

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



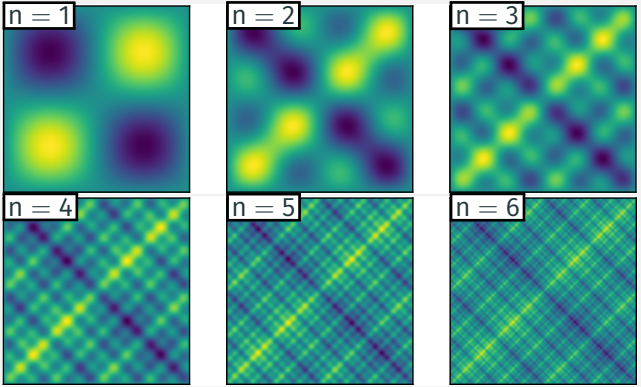
# Multi-Frequency Problem

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

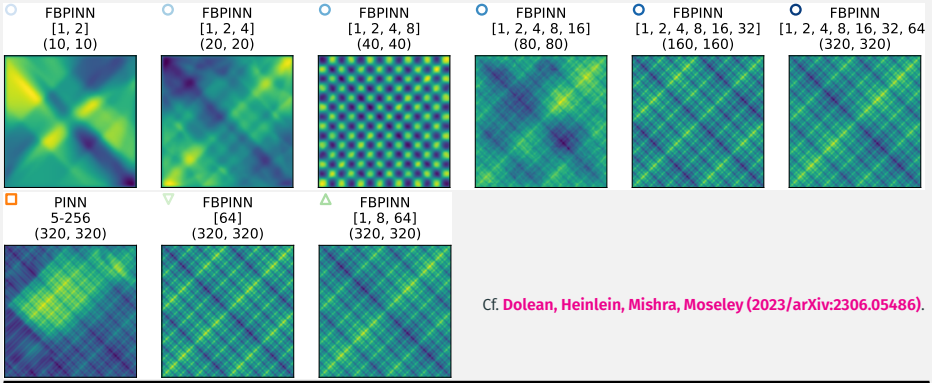
$$
\begin{aligned}
-\Delta u &= 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi \mathbf{x}) \sin(\omega_i \pi \mathbf{y}) && \text{in } \Omega = [0,1]^2, \\
u &= 0 && \text{on } \partial\Omega,
\end{aligned}
$$

with $\omega_i = 2^i$.

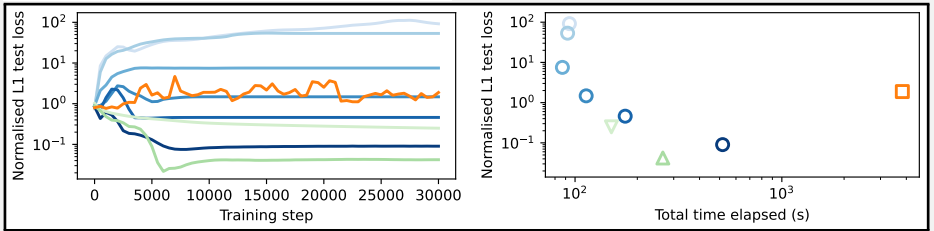
For increasing values of n, we obtain the **analytical solutions**:

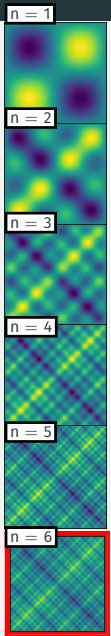


# Multi-Level FBPINNs for a Multi-Frequency Problem – Strong Scaling

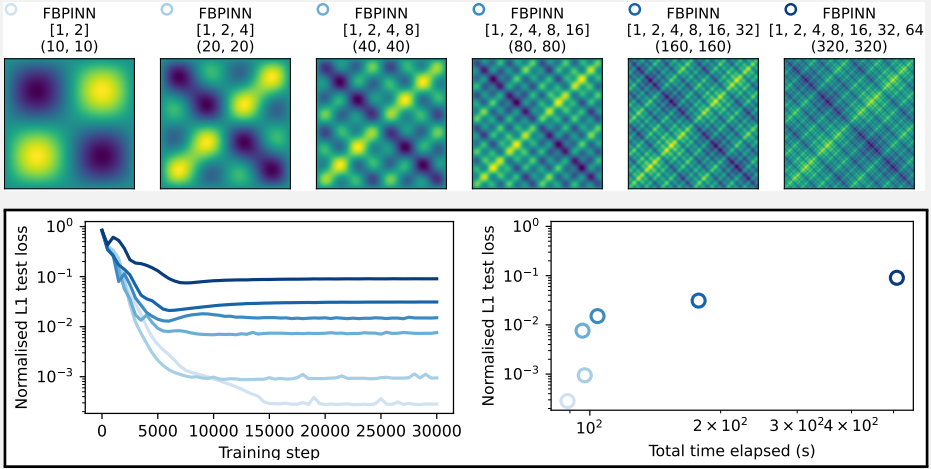


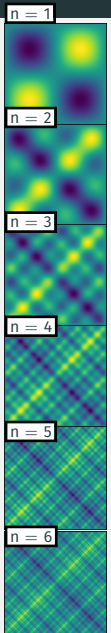

Cf. **Dolean, Heinlein, Mishra, Moseley (2023/arXiv:2306.05486)**.



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



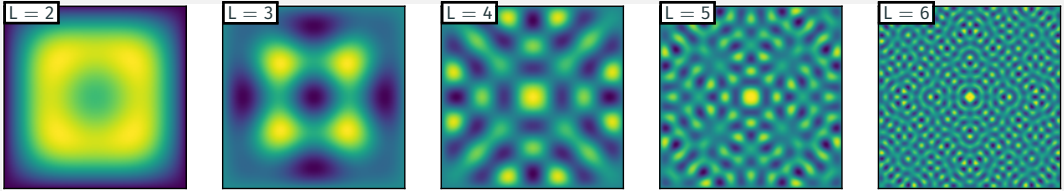

Cf. **Dolean, Heinlein, Mishra, Moseley (2023/arXiv:2306.05486)**.

# 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**:



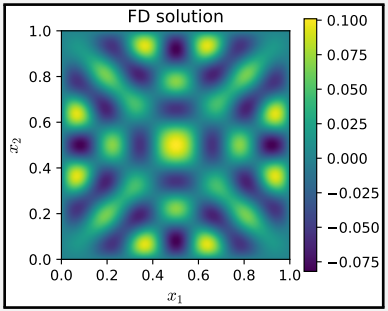
# Multilevel FBPINNs – 2D Helmholtz Problem

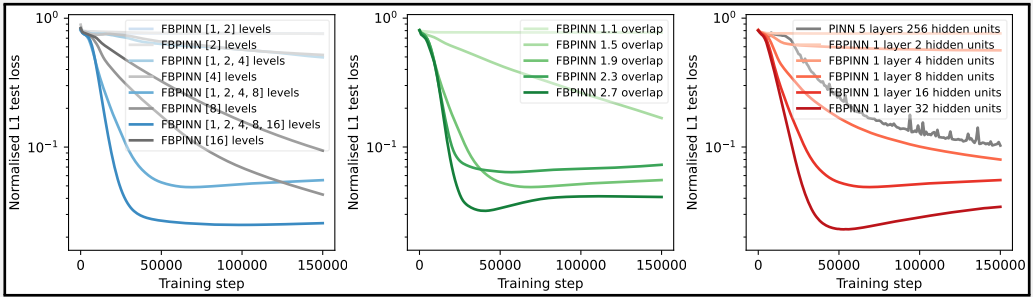
Let us consider the **two-dimensional Helmholtz boundary value problem**

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

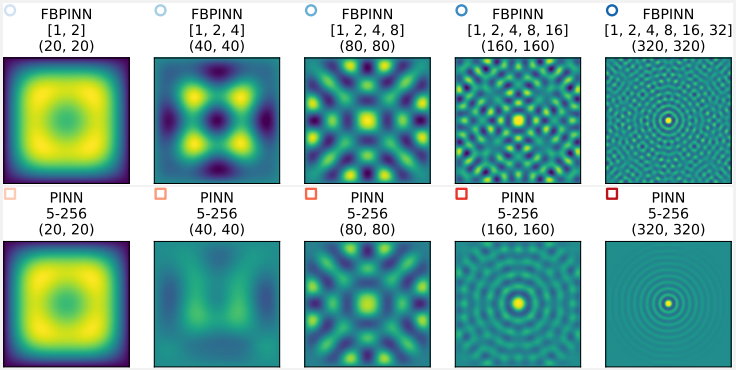
with $k = 2^4\pi/1.6$ and $\sigma = 0.8/2^4$.

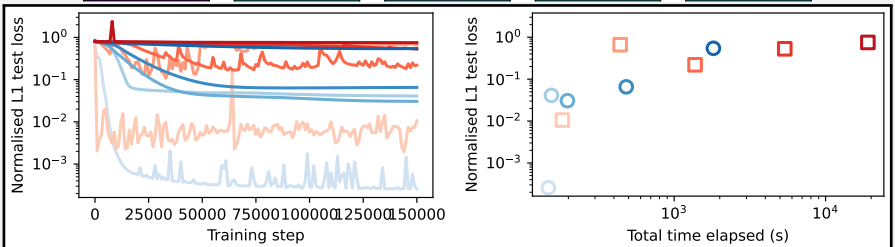
We compute a **reference solution** using **finite differences** with a 5-point stencil on a $320 \times 320$ grid.

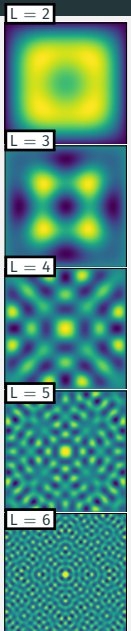


# Multi-Level FBPINNs for the Helmholtz Problem – Weak Scaling



**Conclusions**

- 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 or a given architecture.
- **BUT** training times are still more important than traditional methods when solving PDEs $\rightsquigarrow$ multi-GPU training, parallel FBPINN training algorithm, more inputs to the subdomains networks ... more theory?

**Thanks for your attention!**