Domain decomposition and adaptive sampling for physics-informed neural networks

Alexander Heinlein[1]

Scientific Machine Learning: error control and analysis, Besançon, France, January 15-16, 2025

[1]Delft University of Technology

Outline

1. FBPINNs – Multilevel domain decomposition-based architectures for physics-informed neural networks

   Based on joint work with

   Victorita Dolean (Eindhoven University of Technology)
   Ben Moseley and **Siddhartha Mishra** (ETH Zürich)

2. Stacking multifidelity physics-informed neural networks

   Based on joint work with

   Damien Beecroft (University of Washington)
   Amanda A. Howard and **Panos Stinis** (Pacific Northwest National Laboratory)

3. Multilevel domain decomposition-based physics-informed deep operator networks

   Based on joint work with

   Amanda A. Howard and **Panos Stinis** (Pacific Northwest National Laboratory)

4. PACMANN – Point adaptive collocation method for artificial neural networks

   Based on joint work with

   Bianca Giovanardi and **Coen Visser** (Delft University of Technology)

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

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**

$$\mathcal{N}[u] = f, \quad \text{in } \Omega.$$

PINNs use a **hybrid loss function**:

$$\mathcal{L}(\boldsymbol{\theta}) = \omega_{\text{data}} \mathcal{L}_{\text{data}}(\boldsymbol{\theta}) + \omega_{\text{PDE}} \mathcal{L}_{\text{PDE}}(\boldsymbol{\theta}),$$

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

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

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

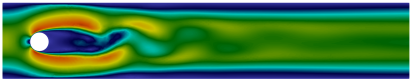
See also **Dissanayake and Phan-Thien (1994); Lagaris et al. (1998)**.

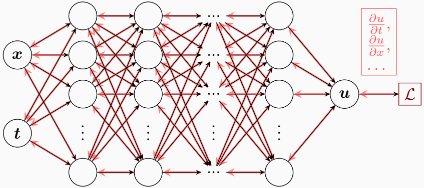
Advantages

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

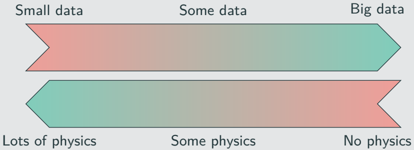
Drawbacks

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

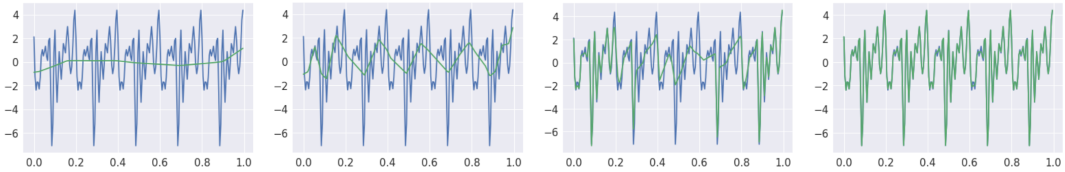
Estimate of the generalization error (Mishra and Molinaro (2022))

The generalization error (or total error) satisfies

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

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

Rule of thumb: **"As long as the PINN is trained well, it also generalizes well"**



100 iterations | 1 000 iterations | 10 000 iterations | 80 000 iterations

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

Solve

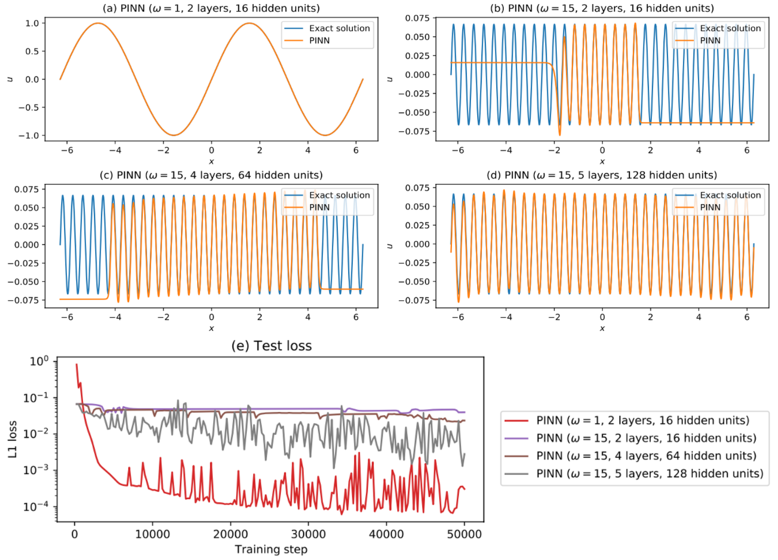
$$u' = \cos(\omega \boldsymbol{x}),$$
$$u(0) = 0,$$

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

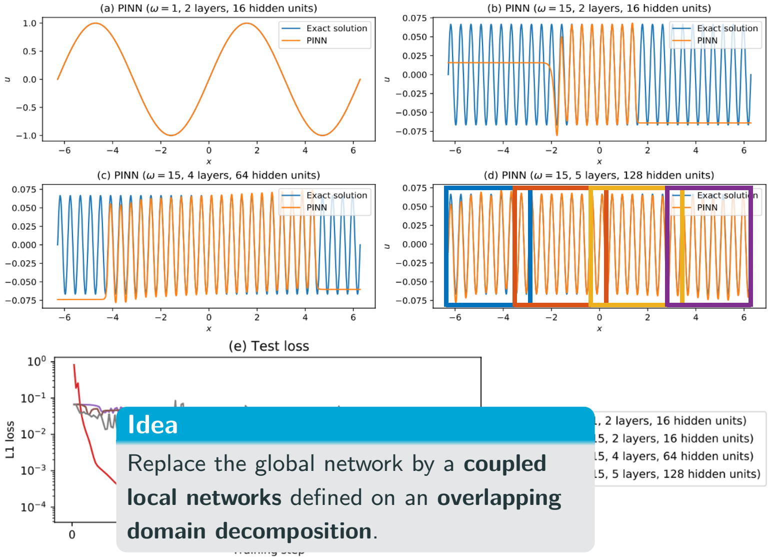
$$
\begin{aligned}
u' &= \cos(\omega \boldsymbol{x}), \\
u(0) &= 0,
\end{aligned}
$$

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

Scaling issues

- Large computational domains
- Small frequencies

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

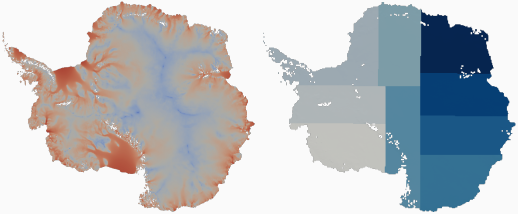


Idea

Replace the global network by a **coupled local networks** defined on an **overlapping domain decomposition**.

(a) 321 free parameters

(d) 66 433 free parameters



Images based on **Heinlein, Perego, Rajamanickam (2022)**

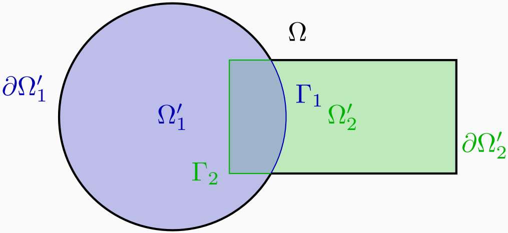
Idea

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**.



Domain Decomposition Methods and Machine Learning – Literature

A **non-exhaustive literature overview**:

- **Machine Learning for adaptive BDDC, FETI–DP, and AGDSW**: Heinlein, Klawonn, Lanser, Weber (2019, 2020, 2021, 2021, 2021, 2022); Klawonn, Lanser, Weber (2024)
- **cPINNs, XPINNs**: Jagtap, Kharazmi, Karniadakis (2020); Jagtap, Karniadakis (2020)
- **Classical Schwarz iteration for PINNs or DeepRitz (D3M, DeepDDM, etc)**:: Li, Tang, Wu, and Liao (2019); Li, Xiang, Xu (2020); Mercier, Gratton, Boudier (arXiv 2021); Dolean, Heinlein, Mercier, Gratton (subm. 2024 / arXiv:2408.12198); Li, Wang, Cui, Xiang, Xu (2023); Sun, Xu, Yi (arXiv 2023, 2024); Kim, Yang (2023, 2024, 2024)
- **FBPINNs, FBKANs**: Moseley, Markham, and Nissen-Meyer (2023); Dolean, Heinlein, Mishra, Moseley (2024, 2024); Heinlein, Howard, Beecroft, Stinis (acc. 2024 / arXiv:2401.07888); Howard, Jacob, Murphy, Heinlein, Stinis (arXiv:2406.19662)
- **DDMs for CNNs**: Gu, Zhang, Liu, Cai (2022); Lee, Park, Lee (2022); Klawonn, Lanser, Weber (2024); Verburg, Heinlein, Cyr (subm. 2024)

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 mid 2024:

A. Klawonn, M. Lanser, J. Weber

Machine learning and domain decomposition methods – a survey

Computational Science and Engineering. 2024

Finite Basis Physics-Informed Neural Networks (FBPINNs)

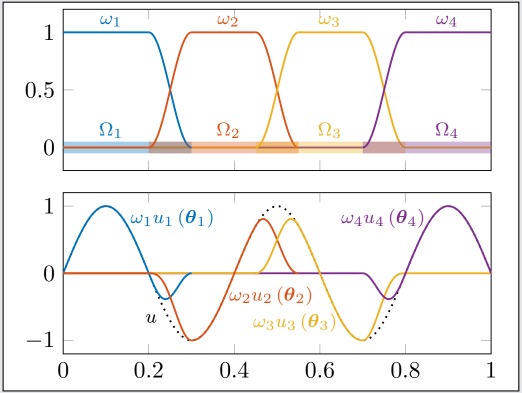
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

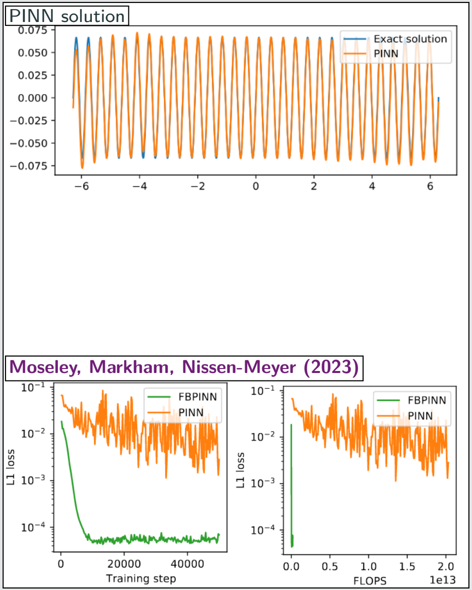
$$u(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J) = \sum\nolimits_{j=1}^{J} \omega_j u_j(\boldsymbol{\theta}_j)$$

and the **loss function**

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



1D single-frequency problem



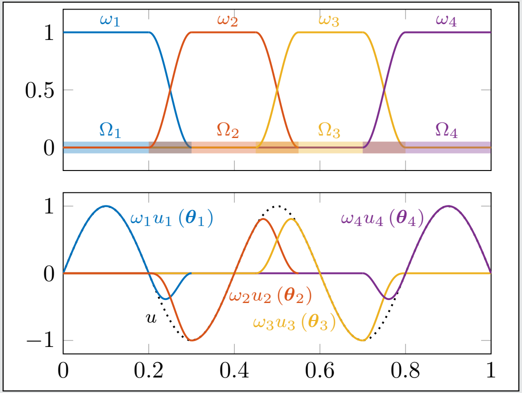
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

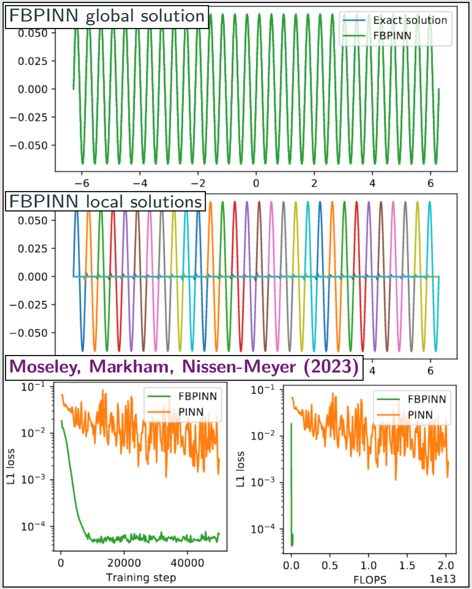
$$u(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J) = \sum\nolimits_{j=1}^{J} \omega_j u_j\left(\boldsymbol{\theta}_j\right)$$

and the **loss function**

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



1D single-frequency problem



Finite Basis Physics-Informed Neural Networks (FBPINNs)

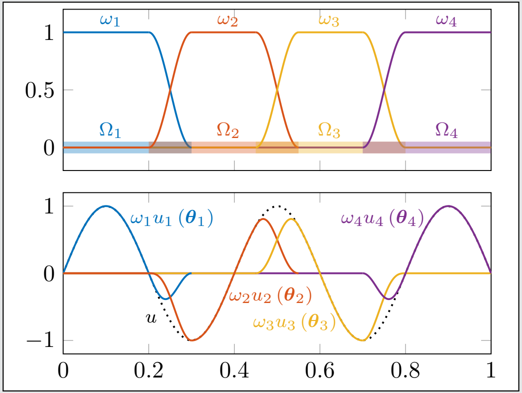
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

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

and the **loss function**

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



Scalability of FBPINNs

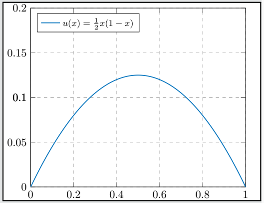
Consider the **simple boundary value problem**

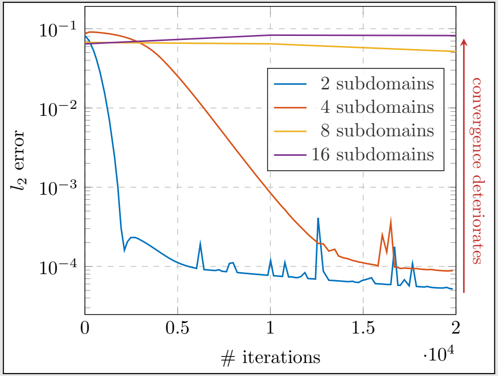
$$-u'' = 1 \text{ in } [0, 1],$$

$$u(0) = u(1) = 0,$$

which has the **solution**

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

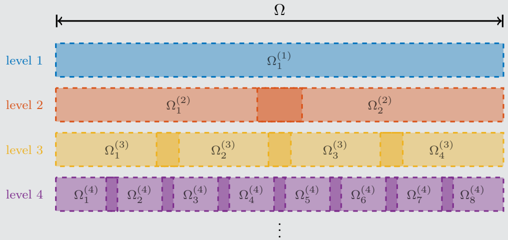




Numerical Results for FBPINNs

Multi-level FBPINNs (ML-FBPINNs)

ML-FBPINNs (**Dolean, Heinlein, Mishra, Moseley (2024)**) are based on a **hierarchy of domain decompositions:**



This yields the **network architecture**

$$u\left(\boldsymbol{\theta}_1^{(1)}, \ldots, \boldsymbol{\theta}_{J^{(L)}}^{(L)}\right) = \sum_{l=1}^{L} \sum_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}\left(\boldsymbol{\theta}_j^{(l)}\right)$$

and the **loss function**

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

Multi-Frequency Problem

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

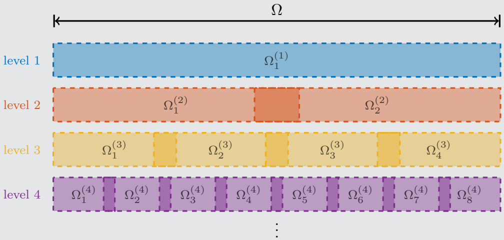
$$\begin{aligned} -\Delta u &= 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

with $\omega_i = 2^i$.

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

Multi-level FBPINNs (ML-FBPINNs)

ML-FBPINNs (**Dolean, Heinlein, Mishra, Moseley (2024)**) are based on a **hierarchy of domain decompositions:**



This yields the **network architecture**

$$u\left(\boldsymbol{\theta}_1^{(1)}, \ldots, \boldsymbol{\theta}_{J^{(L)}}^{(L)}\right) = \sum\nolimits_{l=1}^{L} \sum\nolimits_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}\left(\boldsymbol{\theta}_j^{(l)}\right)$$

and the **loss function**

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

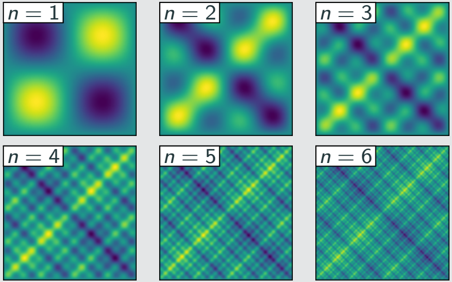
Multi-Frequency Problem

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

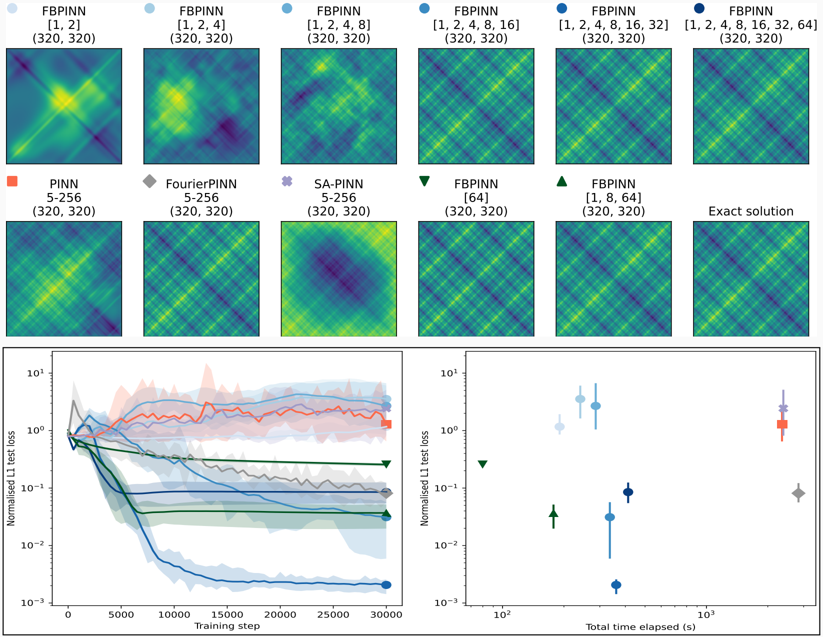
$$\begin{aligned} -\Delta u &= 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

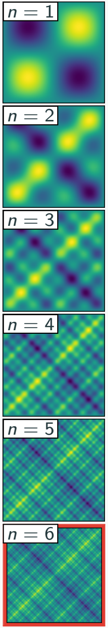
with $\omega_i = 2^i$.

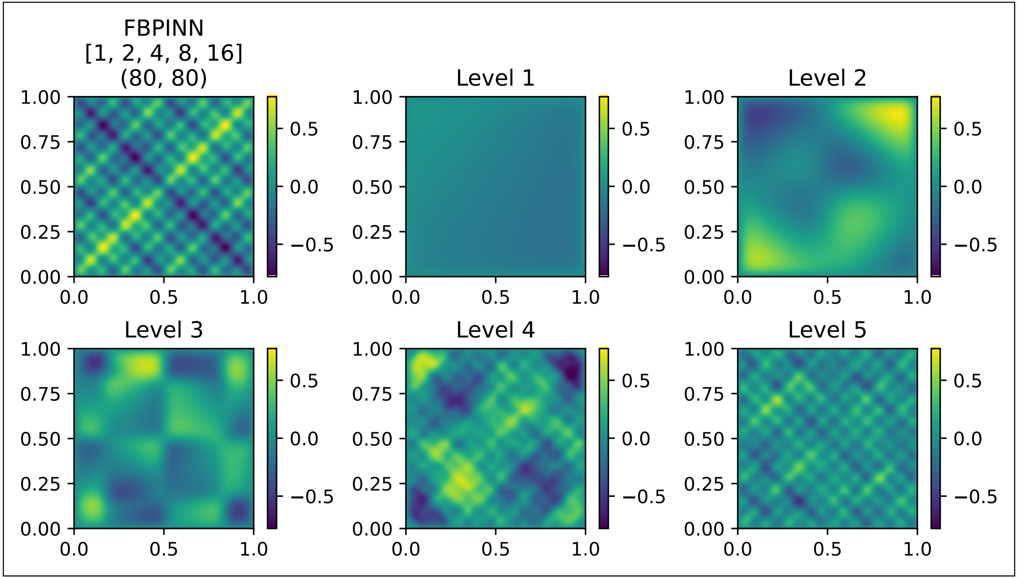
For increasing values of n, we obtain the **analytical solutions**:



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

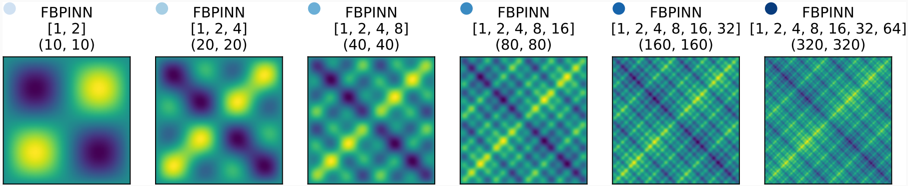


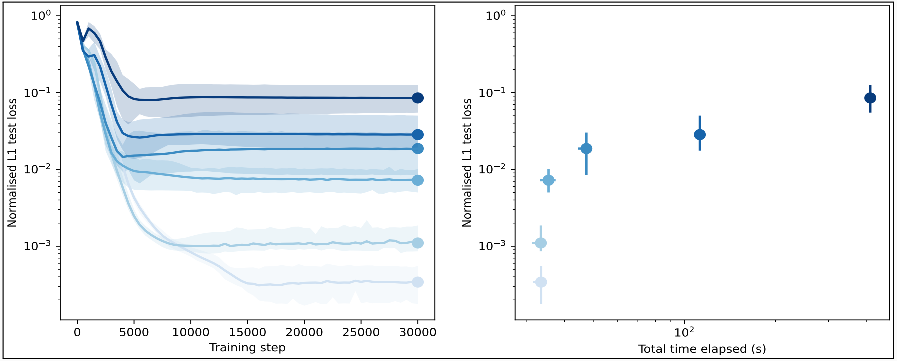


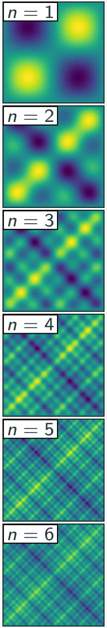


Cf. **Dolean, Heinlein, Mishra, Moseley (2024)**.

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







→ Details and results for the Helmholtz equation can be found in **Dolean, Heinlein, Mishra, Moseley (2024)**.

Stacking multifidelity physics-informed neural networks

PINNs for Time-Dependent Problems

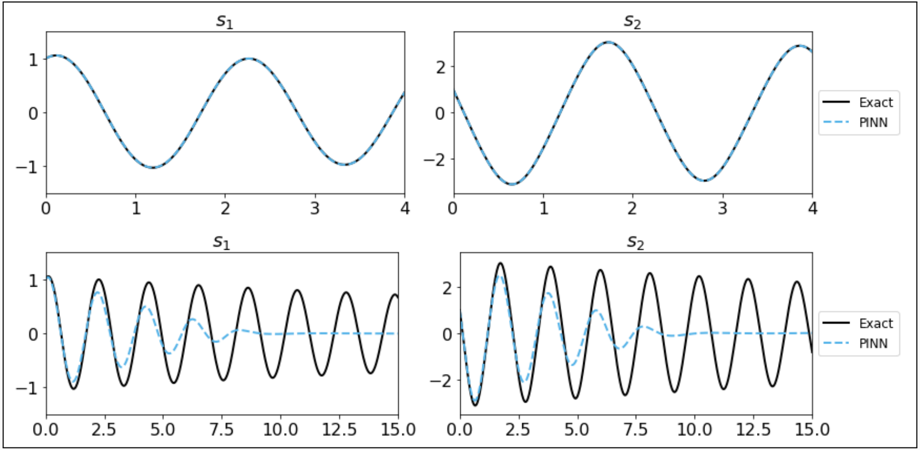
We investigate the performance of PINNs for **time-dependent problems**. Therefore, consider the simple **pedulum problem**:

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

Problem parameters

$m = L = 1,\ b = 0.05,$
$g = 9.81$

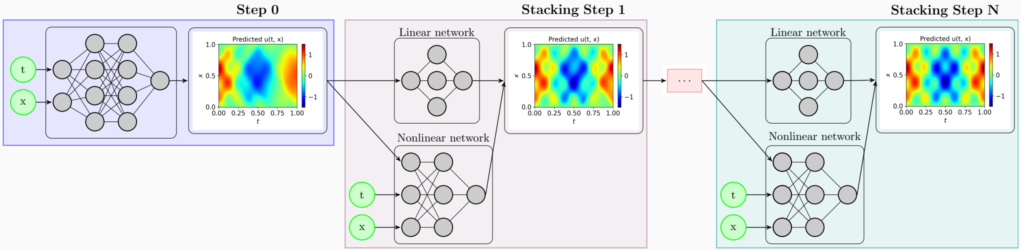
- **Top:** $T = 4$
- **Bottom:** $T = 20$



Stacking Multifidelity PINNs

In the **stacking multifidelity PINNs approach** introduced in **Howard, Murphy, Ahmed, Stinis (arXiv 2023)**, **multiple PINNs are trained in a recursive way.** In each step, a model u^{MF} is trained based on the previous model u^{SF}:

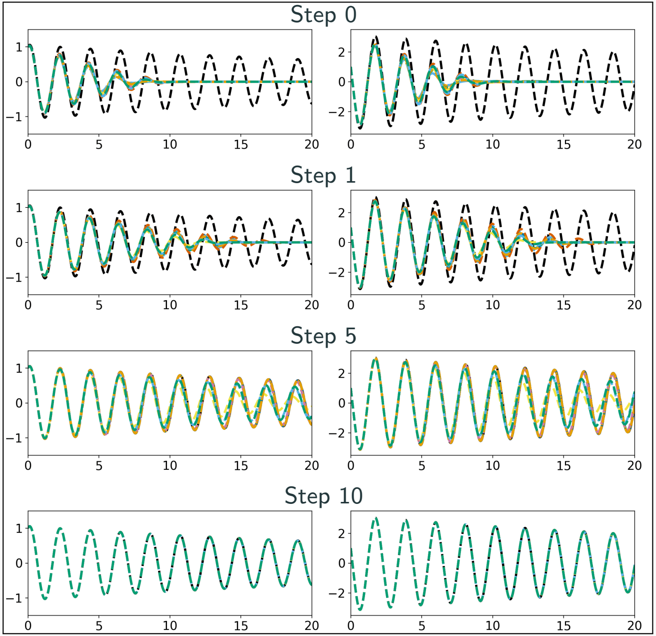
$$u^{MF}(\boldsymbol{x}, \boldsymbol{\theta}^{MF}) = (1 - |\alpha|)\, u^{MF}_{\text{linear}}(\boldsymbol{x}, \boldsymbol{\theta}^{MF}, u^{SF}) + |\alpha|\, u^{MF}_{\text{nonlinear}}(\boldsymbol{x}, \boldsymbol{\theta}^{MF}, u^{SF})$$



Related works (non-exhaustive list)

- **Cokriging & multifidelity Gaussian process regression:** E.g., **Wackernagel (1995); Perdikaris et al. (2017); Babaee et al. (2020)**
- **Multifidelity PINNs & DeepONet: Meng and Karniadakis (2020); Howard, Fu, and Stinis (2024); Howard, Perego, Karniadakis, Stinis (2023); Howard, Murphy, Ahmed, Stinis (arXiv 2023)**
- **Galerkin, multi-level, and multi-stage neural networks: Ainsworth and Dong (2021); Ainsworth and Dong (2022); Aldirany et al. (2024); Wang and Lai (2024)**

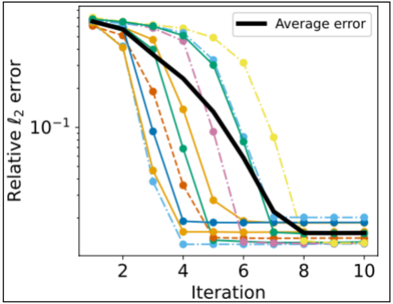
Stacking Multifidelity PINNs for the Pendulum Problem



Pendulum problem:

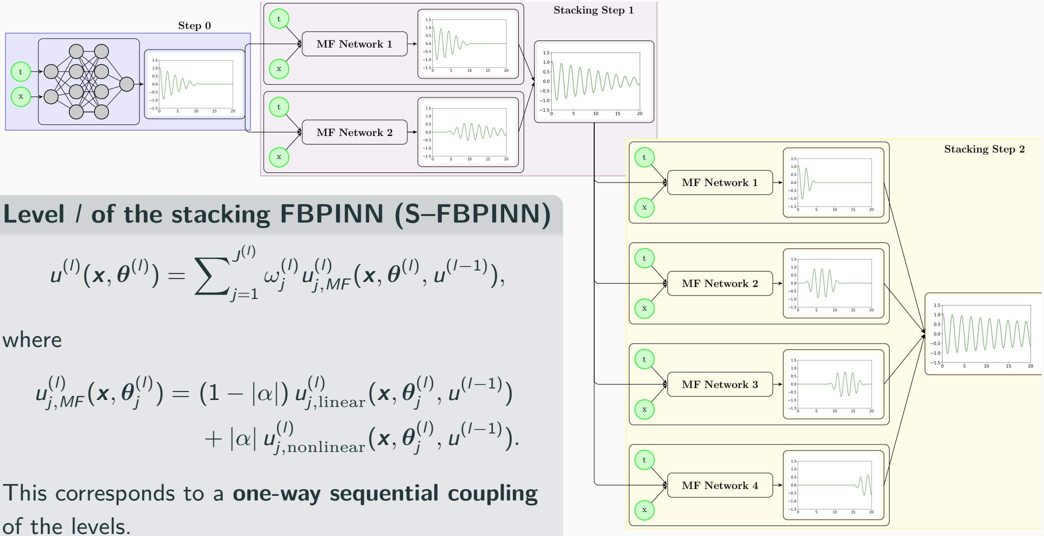
$$\frac{ds_1}{dt} = s_2,$$
$$\frac{ds_2}{dt} = -\frac{b}{m}s_2 - \frac{g}{L}\sin(s_1).$$

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



Stacking Multifidelity FBPINNs

In **Heinlein, Howard, Beecroft, and Stinis (acc. 2024 / arXiv:2401.07888)**, we **combine stacking multifidelity PINNs with FBPINNs** by using an FBPINN model in each stacking step.



Level l of the stacking FBPINN (S–FBPINN)

$$u^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}^{(l)}) = \sum\nolimits_{j=1}^{J^{(l)}} \omega_j^{(l)} u_{j,MF}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}^{(l)}, u^{(l-1)}),$$

where

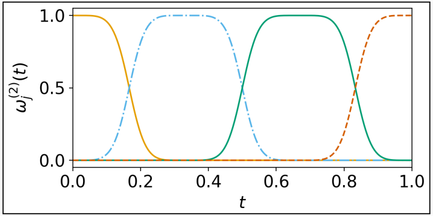
$$u_{j,MF}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}_j^{(l)}) = (1 - |\alpha|)\, u_{j,\mathrm{linear}}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}_j^{(l)}, u^{(l-1)}) + |\alpha|\, u_{j,\mathrm{nonlinear}}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}_j^{(l)}, u^{(l-1)}).$$

This corresponds to a **one-way sequential coupling** of the levels.

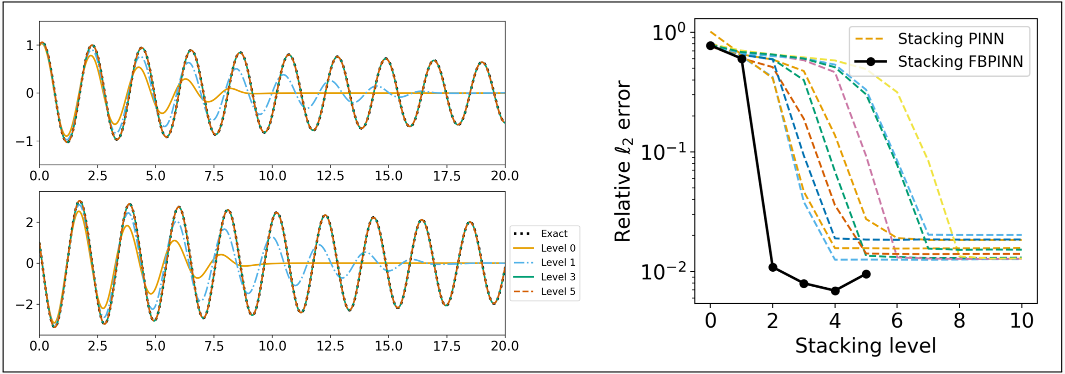
First, we consider a **pedulum problem** and **compare the stacking multifidelity PINN and FBPINN** approaches:

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

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



Exemplary partition of unity in time



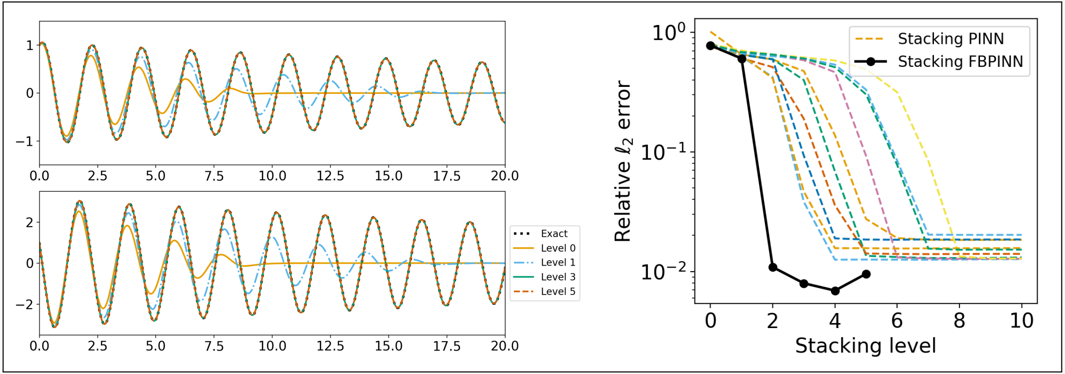
First, we consider a **pedulum problem** and **compare the stacking multifidelity PINN and FBPINN** approaches:

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

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

Model details:

method	arch.	# levels	# params	error
S–PINN	5x50, 1x20	4	63 018	0.0125
S–FBPINN	3x32, 1x 4	2	34 570	0.0074

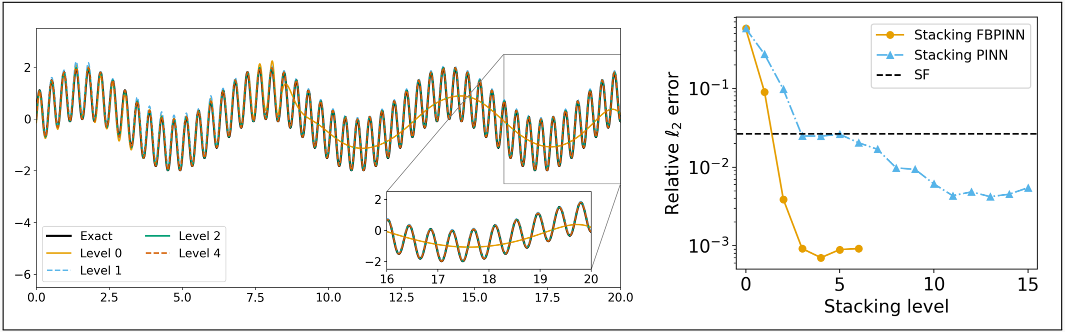


Second, we consider a **two-frequency problem**:

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

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

method	arch.	# levels	# params	error
PINN	4x64	0	12 673	0.6543
PINN	5x64	0	16 833	0.0265
S–PINN	4x16, 1x5	3	4900	0.0249
S–PINN	4x16, 1x5	10	11 179	0.0061
S–FBPINN	4x16, 1x5	2	7822	0.00415
S–FBPINN	4x16, 1x5	5	59 902	0.00083

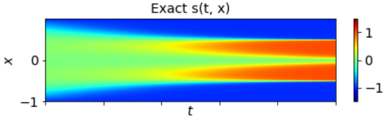


→ Due to the **multiscale structure of the problem**, the **improvements** due to the **multifidelity FBPINN approach** are **even stronger**.

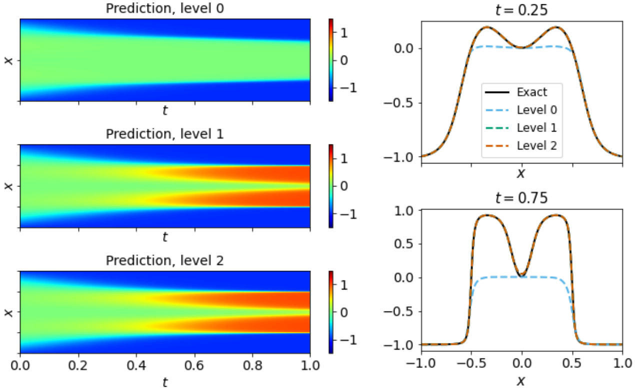
Numerical Results – Allen–Cahn Equation

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

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



method	arch.	# params	error
PINN	0 levels 6x50	12 951	0.499
S–FBPINN	2 levels 5x32, 1x20	39 627	0.00594

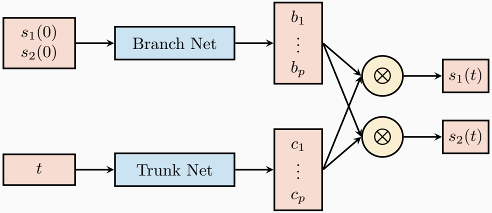


PINN **gets stuck** at **fixed point of the of dynamical system**; cf. **Rohrhofer et al. (arXiv 2023)**.

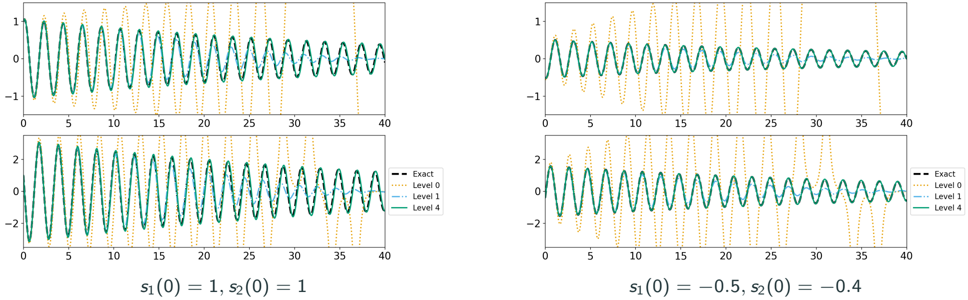
Multilevel domain decomposition-based physics-informed deep operator networks

DeepONets (Lu et al. (2021))

- While PINNs learn individual solutions, neural operators learn operators between function spaces, such as **solution operators**
- Deep operator networks (DeepONets) are **compatible with the PINN approach** but physics-informed DeepONets (PI-DONs) are challenging to train



Approach based on the **single-layer case** analyzed in Chen and Chen (1995)

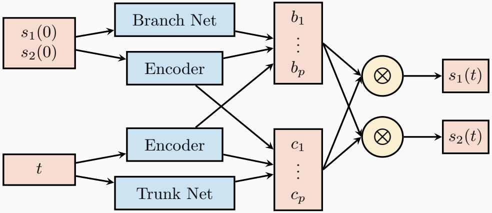


Cf. Heinlein, Howard, Beecroft, and Stinis (acc. 2024).

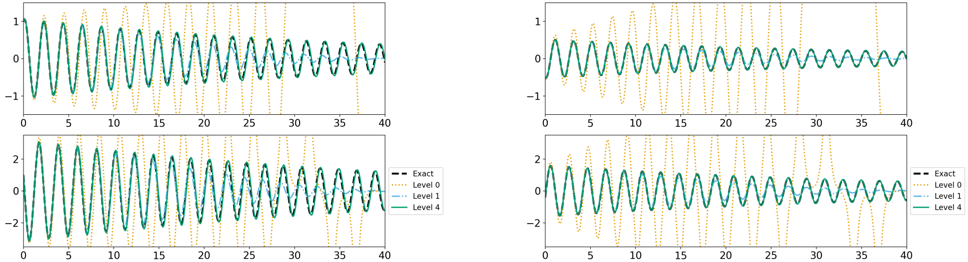
Deep Operator Networks (DeepONets / DONs)

DeepONets (Lu et al. (2021))

- While PINNs learn individual solutions, neural operators learn operators between function spaces, such as **solution operators**
- **Deep operator networks (DeepONets)** are **compatible with the PINN approach** but **physics-informed DeepONets (PI-DONs)** are challenging to train



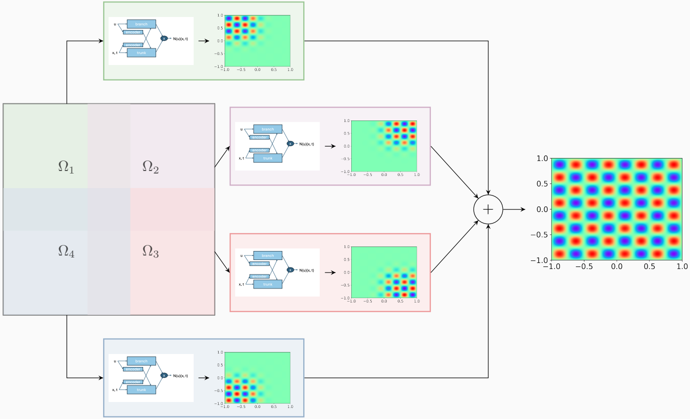
Modified DeepONet architecture; cf. **Wang, Wang, and Perdikaris (2022)**



$s_1(0) = 1, s_2(0) = 1$ $\qquad$ $s_1(0) = -0.5, s_2(0) = -0.4$

Cf. **Heinlein, Howard, Beecroft, and Stinis (acc. 2024)**.

Finite Basis DeepONets (FBDONs)



Howard, Heinlein, Stinis (in prep.)

Variants:

Shared-trunk FBDONs (ST-FBDONs)

The trunk net learns spatio-temporal basis functions. In ST-FBDONs, we use the same trunk network for all subdomains.

Stacking FBDONs

Combination of the stacking multifidelity approach with FBDONs.

Heinlein, Howard, Beecroft, Stinis (acc. 2024/arXiv:2401.07888)

Pendulum problem

$$\frac{ds_1}{dt} = s_2, \qquad t \in [0, T],$$
$$\frac{ds_2}{dt} = -\frac{b}{m}s_2 - \frac{g}{L}\sin(s_1), \qquad t \in [0, T],$$

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

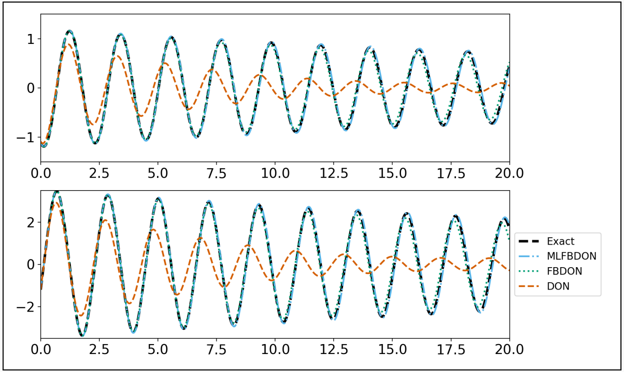
Parametrization

Initial conditions:

$$s_1(0) \in [-2, 2] \qquad s_2(0) \in [-1.2, 1.2]$$

$s_1(0)$ and $s_2(0)$ are the also inputs of the branch network.

Training on 50 k different configurations



Mean rel. l_2 error on 100 config.	
DeepONet	0.94
FBDON (32 subd.)	0.84
MLFBDON ([1, 4, 8, 16, 32] subd.)	0.27

Cf. **Howard, Heinlein, Stinis (in prep.)**

DD-DONs Wave Equation

Wave equation

$$\frac{d^2 s}{dt^2} = 2\frac{d^2 s}{dx^2}, \qquad (x,t) \in [0,1]^2$$

$$s_t(x,0) = 0, x \in [0,1], \quad s(0,t) = s(1,t) = 0,$$

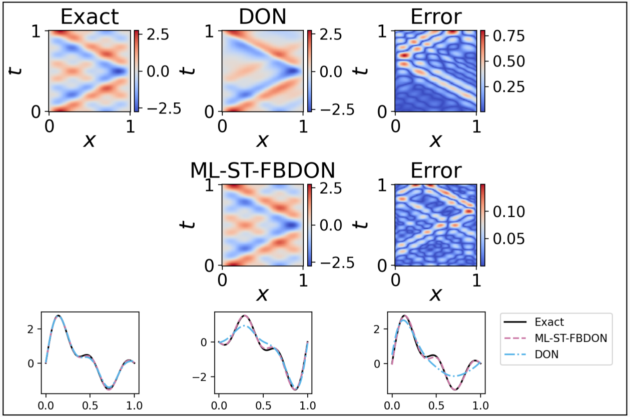
Solution: $s(x,t) = \sum_{n=1}^{5} b_n \sin(n\pi x)\cos(n\pi\sqrt{2}t)$

Parametrization

Initial conditions for s parametrized by $b = (b_1, \ldots, b_5)$ (normally distributed):

$$s(x,0) = \sum_{n=1}^{5} b_n \sin(n\pi x) \quad x \in [0,1]$$

Training on 1000 random configurations.



Mean rel. l_2 error on 100 config.	
DeepONet	0.30 ± 0.11
ML-ST-FBDON ([1, 4, 8, 16] subd.)	0.05 ± 0.03
ML-FBDON ([1, 4, 8, 16] subd.)	0.08 ± 0.04

→ Sharing the trunk network does not only save in the number of parameters but even yields better performance

Cf. **Howard, Heinlein, Stinis (in prep.)**

PACMANN – Point adaptive collocation method for artificial neural networks

Motivation

The **number and distribution of the collocation points in the PDE loss $\mathcal{L}_{\text{PDE}}$ have a significant influence on the accuracy of the PINN solution**. Since the **computational work grows with the number of collocation points**, the **effective placement of the collocation points** is important.

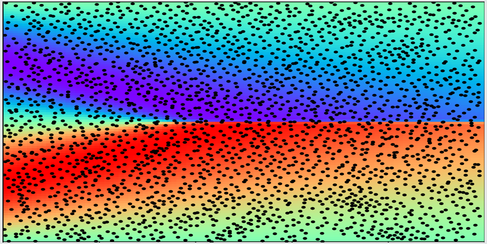
Burger's equation in 1D

Consider the Burger's with one spatial dimension:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \qquad x \in [-1, 1], \quad t \in [0, 1]$$

$$u(x, 0) = -\sin(\pi x) \quad u(-1, t) = u(1, t) = 0.$$

sampling method	L_2 relative error		mean
	mean	1 SD	runtime [s]
uniform grid	25.9%	14.2%	425
Hammersley grid	0.61%	0.53%	443
random resampling	0.40%	0.35%	**423**
Residual-based	0.11%	**0.05%**	450



Implementation based on DeepXDE with PyTorch (v1.12.1) backend; cf. **Visser, Heinlein, and Giovanardi (arXiv:2411.19632)**

Similar results in earlier study: **Wu et al. (2023)**

Overview of Various Sampling Schemes (Not Exhaustive)

Non-adaptive sampling

- **Equispaced uniform grid**
- **Uniformly random sampling**, using a pseudo-random number generator (e.g., PCG-64 **O'Neill (2014)**)
- **Latin hypercube sampling** (**McKay, Beckman, and Conover (2000); Stein (1987)**)

Quasi-random low-discrepancy sequences:

- **Halton sequence** (**Halton (1960)**)
- **Hammersley sequence** (**Hammersley (1964)**)
- **Sobol sequence** (**Sobol' (1967)**)

Adaptive sampling

- **Residual-based Adaptive Refinement (RAR)** (**Lu et al (2021)**): placement of **additional points** in regions with the largest PDE residuals
- **Probability Density Function (PDF)** (**Nabian, Gladstone, and Meidani (2021)**): **randomly resample all points** based on a **PDF proportional to the residual**
- **Residual-based Adaptive Distribution (RAD)** (**Wu et al. (2023)**): **all collocation points are resampled** using a **PDF based on the residual (nonlinear)**.
- **Residual-based Adaptive Refinement with Distribution (RAR-D)** (**Wu et al. (2023)**): **sampling of additional collocation points** using the **PDF used in RAD**

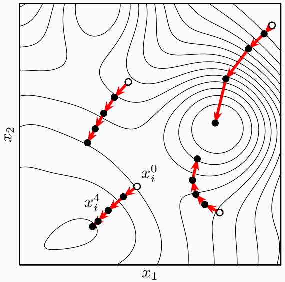
(RAD and RAR-D are based on / extensions of PDF approach)

PACMANN – Point Adaptive Collocation Method for Artificial Neural Networks

In **Visser, Heinlein, and Giovanardi (arXiv:2411.19632)**, the collocation points are updated by solving the **min-max problem**

$$\min_{\boldsymbol{\theta}} \left[\omega_{\text{data}} \mathcal{L}_{\text{data}}(\boldsymbol{\theta}) + \max_{\boldsymbol{X} \subset \Omega} \omega_{\text{PDE}} \mathcal{L}_{\text{PDE}}(\boldsymbol{X}, \boldsymbol{\theta}) \right].$$

Different from the other residual-based adaptive sampling methods, the **existing collocation points are moved** using a gradient-based optimizers, such as **gradient ascent**, **RMSprop** (**Hinton (2018)**), **Adam** (**Kingma, Ba (2017)**), or others.



Algorithm 1: PACMANN with iteration counts P and T and stepsize s

Sample a set $\boldsymbol{X}$ of N_{PDE} collocation points using a uniform sampling method;
while *stopping criterion not reached* **do**
  Train the PINN for P iterations;
  for $k = 1, \ldots, T$ **do**
    Compute squared residual $\mathcal{R}(\boldsymbol{x}_i) = (\mathcal{N}[u](\boldsymbol{x}_i, \boldsymbol{\theta}) - f(\boldsymbol{x}_i))^2$ for all $\boldsymbol{x}_i \in \boldsymbol{X}$;
    Compute gradient $\nabla_{\boldsymbol{x}} \mathcal{R}(\boldsymbol{x}_i)$ for all $\boldsymbol{x}_i \in \boldsymbol{X}$;
    Move the points in $\boldsymbol{X}$ according to the chosen optimization algorithm and stepsize s;
  end
  Resample points in $\boldsymbol{X}$ that moved outside Ω based on a uniform probability distribution;
end

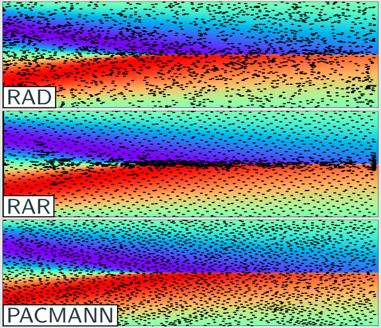
Numerical Results – Burger's Equation in 1D

Varying the optimizer in PACMANN

sampling method	L_2 relative error		mean	hyper parameters	
	mean	1 SD	runtime [s]	stepsize s	# steps T
PACMANN–gradient ascent	0.30%	0.17%	**436**	10^{-6}	1
PACMANN–RMSprop	0.10%	**0.03%**	442	10^{-6}	10
PACMANN–Adam	**0.07%**	0.05%	461	10^{-5}	15

Comparison against different methods

sampling method	L_2 relative error		mean
	mean	1 SD	runtime [s]
uniform grid	25.9%	14.2%	425
Hammersley grid	0.61%	0.53%	443
random resampling	0.40%	0.35%	**423**
RAR	0.11%	**0.05%**	450
RAD	0.16%	0.10%	463
RAR-D	0.24%	0.21%	503
PACMANN–Adam	**0.07%**	**0.05%**	461



Cf. **Visser, Heinlein, and Giovanardi (arXiv:2411.19632)**.

Numerical Results – Poisson Equation in 5D

Furthermore, we show that our method **scales well to higher dimensions**, such as a **Poisson equation in five dimensions**:

$$-\Delta u = f, \quad \text{in } \Omega = [-1,1]^5,$$
$$u = 0, \quad \text{on } \partial\Omega.$$

Here, f is chosen such that $u = \prod_{i=1}^{5} \sin(\pi x_i)$.

Comparison against different methods

sampling method	L_2 relative error mean	L_2 relative error 1 SD	mean runtime [s]
uniform grid	17.89%	0.94%	742
Hammersley grid	82.08%	3.23%	**734**
random resampling	11.03%	0.69%	772
RAR	56.84%	4.46%	753
RAD	10.07%	0.75%	851
RAR-D	88.30%	1.53%	774
PACMANN–Adam	**5.93%**	**0.46%**	778

Cf. **Visser, Heinlein, and Giovanardi (arXiv:2411.19632)**.

Numerical Results – Parameter Identification for the Navier–Stokes Equations

Finally, we consider an **inverse problem** involving the **Navier-Stokes equations in two dimensions** of an **incompressible flow past a cylinder** discussed by **Raissi et al. (2019)**:

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}), \quad \boldsymbol{x} \in [1,8] \times [-2,2], t \in [0,7],$$

$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}), \quad \boldsymbol{x} \in [1,8] \times [-2,2], t \in [0,7].$$

Here, (u, v) and p are the velocity and pressure fields. The scalar parameter λ_1 scales the convective term, and λ_2 represents the dynamic (shear) viscosity. The true values of λ_1 and λ_2 are 1 and 0.01.

sampling method	L_2 relative error				mean runtime [s]
	λ_1		λ_2		
	mean	1 SD	mean	1 SD	
uniform grid	0.05 %	**0.01 %**	0.72 %	0.43 %	1506
Hammersley grid	0.08 %	0.04 %	0.89 %	0.52%	**1492**
random resampling	0.12 %	0.05 %	0.65 %	0.46 %	1514
RAR	0.30 %	0.06 %	1.44 %	0.90 %	1520
RAD	0.23 %	0.06 %	1.38 %	0.79 %	1583
RAR-D	0.08 %	0.05 %	0.84 %	0.57 %	1525
PACMANN–Adam	**0.03 %**	0.03 %	**0.53 %**	**0.19 %**	1559

Cf. **Visser, Heinlein, and Giovanardi (arXiv:2411.19632)**.

Annual Meeting of EMS activity group on Scientific Machine Learning

Organizing committee: P.F. Antonietti, S. Pagani, F. Regazzoni, M. Verani (chair), P. Zunino

Scientific committee: Members of the EMS activity group on Scientific Machine Learning

- **Dates:** March 24 – 28, 2025
- **Event:** First Annual Meeting of the EMS-AI Scientific Machine Learning (SciML) activity group
- **Focus:** Bridging mathematics, computer science, and applications in SciML
- **Program Highlights:**
  - 18 Invited Talks
  - 2 Industrial Sessions
  - Poster Session
  - Roundtable Discussion on *Interplay between machine learning, applied mathematics, and scientific computing*; chair: Wil Schilders (ICIAM President)

Deadline for registration: January 31, 2025!

CWI Research Semester: Synergies in Numerical Linear Algebra and Machine Learning

Co-organizers: Victorita Dolean (TU/e), Alexander Heinlein (TU Delft), Benjamin Sanderse (CWI), Jemima Tabbeart (TU/e), Tristan van Leeuwen (CWI)

- **Autumn School** (October 27–31, 2025):
  - Chris Budd (University of Bath)
  - Ben Moseley (Imperial College London)
  - Gabriele Steidl (Technische Universität Berlin)
  - Andrew Stuart (California Institute of Technology)
  - Andrea Walther (Humboldt-Universität zu Berlin)
- **Workshop** (December 1–3, 2025):
  - 3 days with plenary talks (academia & industry) and an industry panel
  - Confirmed plenary speakers:
    - Marta d'Elia (Meta)
    - Benjamin Peherstorfer (New York University)
    - Andreas Roskopf (Fraunhofer Institute)

CWI Centrum Wiskunde & Informatica

Join us for inspiring talks, hands-on sessions, and industry collaboration!

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.

Stacking Multifidelity FBPINNs

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

PACMANN Sampling Method

- Adaptive movement of the collocation points along the gradient yields comparable or better performance **compared to state-of-the-art sampling approaches**; standard optimizers can be employed.
- In particular, for **high-dimensional problems**, the performance is clearly better.

Thank you for your attention!

Topical Activity Group
Scientific Machine Learning