Domain Decomposition for Randomized Neural Networks

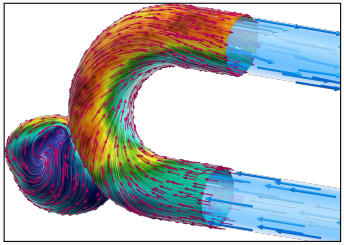
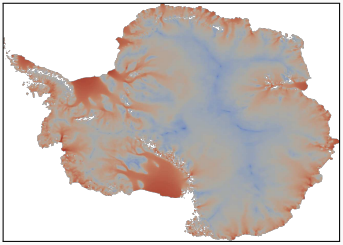
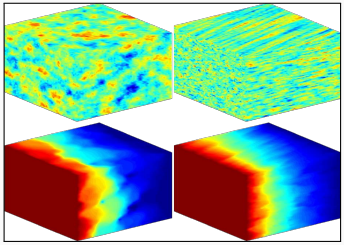
Alexander Heinlein[1]

3rd IACM Digital Twins in Engineering Conference (DTE 2025) & 1st ECCOMAS Artificial Intelligence and Computational Methods in Applied Science (AICOMAS 2025), Paris, France, February 17–21, 2025

[1]Delft University of Technology

Based on joint work with **Siddhartha Mishra** (ETH Zürich) and **Yong Shang** and **Fei Wang** (Xi'an Jiaotong University)

Numerical Analysis and Machine Learning

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

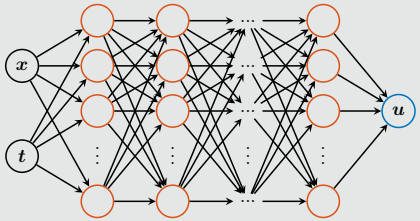
Randomized Neural Networks (RaNNs)

Neural networks

A standard **multilayer perceptron (MLP)** with L hidden layers is a **parametric** model of the form

$$u(\boldsymbol{x}, \boldsymbol{\theta}) = F_{L+1}^{\boldsymbol{A}} \cdot F_L^{\boldsymbol{W}_L, \boldsymbol{b}_L} \circ \ldots \circ F_1^{\boldsymbol{W}_1, \boldsymbol{b}_1}(\boldsymbol{x}),$$

where $\boldsymbol{A}$ is linear, and the ith hidden layer is nonlinear $F_i^{\boldsymbol{W}_i, \boldsymbol{b}_i}(\boldsymbol{x}) = \sigma(\boldsymbol{W}_i \cdot \boldsymbol{x} + \boldsymbol{b}_i)$.



In order to optimize the loss function

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}),$$

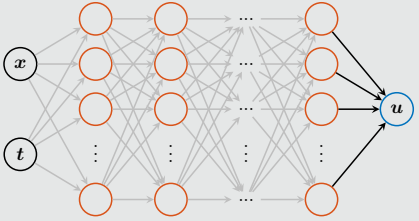
all parameters $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{W}_1, \boldsymbol{b}_1, \ldots, \boldsymbol{W}_L, \boldsymbol{b}_L)$ are **trained**.

Randomized neural networks

In **randomized neural networks (RaNNs)** as introduced by **Pao and Takefuji (1992)**,

$$u(\boldsymbol{x}, \boldsymbol{A}) = F_{L+1}^{\boldsymbol{A}} \cdot F_L^{\boldsymbol{W}_L, \boldsymbol{b}_L} \circ \ldots \circ F_1^{\boldsymbol{W}_1, \boldsymbol{b}_1}(\boldsymbol{x}),$$

the weights in the hidden layers are randomly initialized and **fixed**; only $\boldsymbol{A}$ is trainable.



The model is linear with respect to the trainable parameters $\boldsymbol{A}$, and the optimization problem reads

$$\min_{\boldsymbol{A}} \mathcal{L}(\boldsymbol{A}).$$

This can **simplify the training process**.

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

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

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

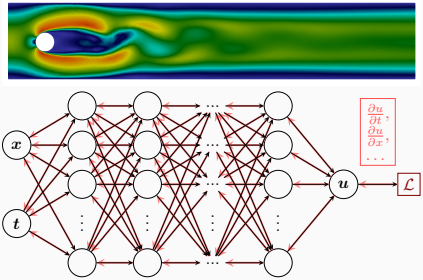
See also **Dissanayake and Phan-Thien (1994); Lagaris et al. (1998)**.

Advantages

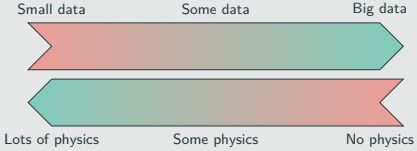
- "Meshfree"
- Small data
- Generalization properties
- High-dimensional problems
- Inverse and parameterized problems

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

Scaling of PINNs for a Simple ODE Problem

Solve

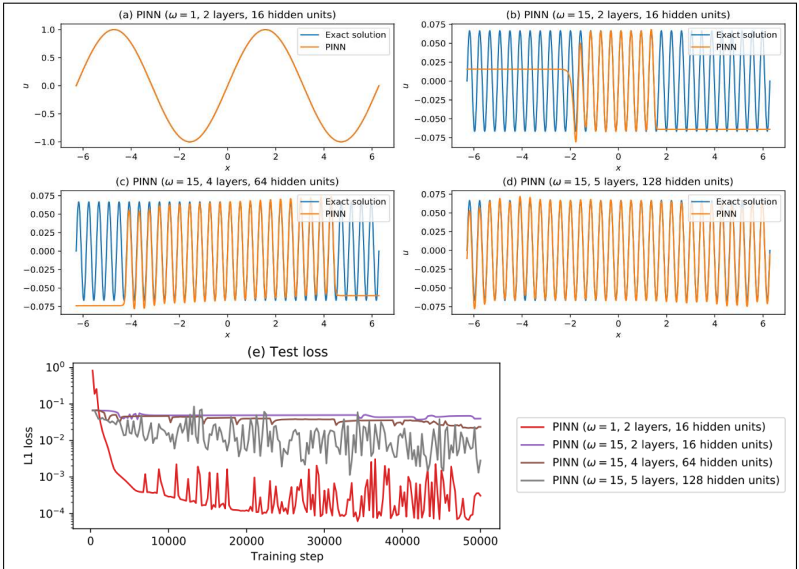
$$\begin{aligned} u' &= \cos(\omega x), \\ u(0) &= 0, \end{aligned}$$

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

Scaling of PINNs for a Simple ODE Problem

Solve

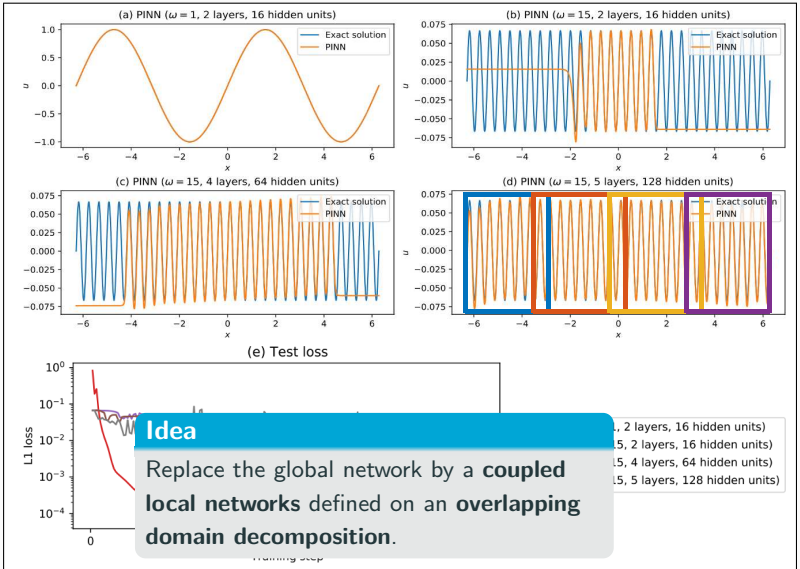
$$\begin{aligned} u' &= \cos(\omega x), \\ u(0) &= 0, \end{aligned}$$

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 **coupled local networks** defined on an **overlapping domain decomposition**.

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

Finite Basis Physics-Informed Neural Networks (FBPINNs)

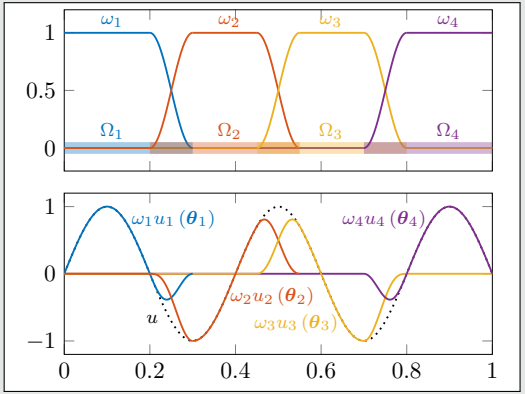
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

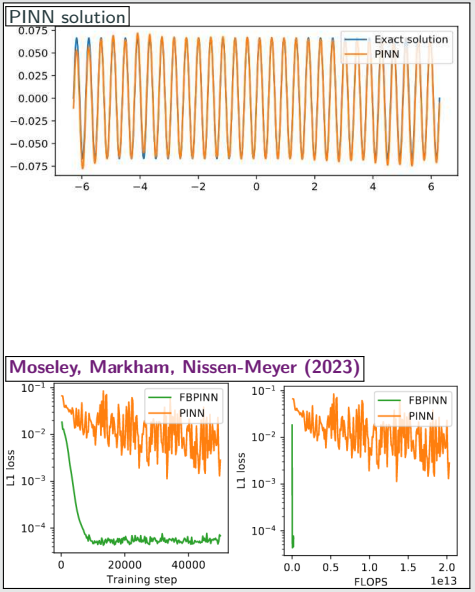
$$u(\theta_1, \dots, \theta_J) = \sum_{j=1}^{J} \omega_j u_j(\theta_j)$$

and the **loss function**

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



1D single-frequency problem



Finite Basis Physics-Informed Neural Networks (FBPINNs)

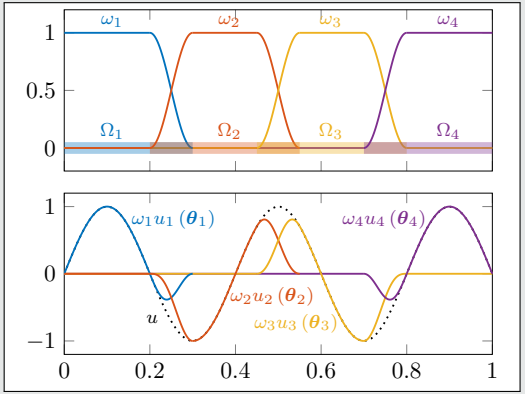
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

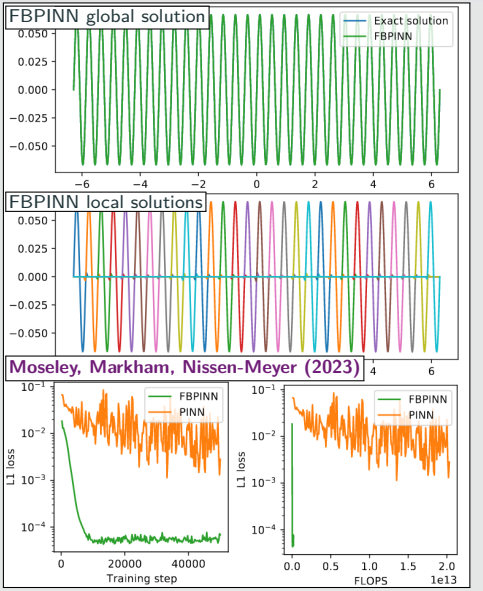
$$u(\theta_1, \dots, \theta_J) = \sum_{j=1}^{J} \omega_j u_j(\theta_j)$$

and the **loss function**

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



1D single-frequency problem



Physics-Informed Randomized Neural Networks (PIRaNNs)

Physics-informed randomized neural networks (PIRaNNs) make use of the aforementioned linearization of the model with respect to the trainable parameters as well as the fact that RaNNs retain **universal approximation properties**, as shown in **Igelnik and Pao (1995)**.

Consider a linear differential operator $\mathcal{A}$. Then, solving the PDE

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

using PIRaNNs yields the linear equation system

$$\mathcal{A}[u](\boldsymbol{x}_i) = f(\boldsymbol{x}_i), \quad i = 1, \dots, N_{\text{PDE}},$$

where N_{PDE} is the number of collocation points.

Hard enforcement of boundary conditions

We construct u to explicitly satisfy bcs:

$$u(\boldsymbol{x}, \boldsymbol{A}) = G(\boldsymbol{x}) + L(\boldsymbol{x}) NN(\boldsymbol{x}, \boldsymbol{A})$$

- NN is a **feedforward neural network** with **trainable parameters** $\boldsymbol{A}$
- G and L are **fixed functions**, chosen such that u satisfies the boundary conditions

The resulting linear equation system

$$\boldsymbol{H}\boldsymbol{A} = \boldsymbol{f}$$

generally does not have a unique solution. In fact, $\boldsymbol{H}$ is typically **rectangular**, **dense**, and **ill-conditioned**.

Solving the system using least squares corresponds to applying the classical PINN loss function to the RaNN model u. As we will see, this approach offers a **potentially efficient alternative**.

(Non-Exhaustive) Literature Overview

Randomized neural networks

- RaNNs: Pao, Takefuji (1992); Pao Park, Sobajic (1994); Igelnik, Pao (1995)
- Extreme Learning Machines (ELMs): Huang, Zhu, Siew (2006); Liu, Lin, Fang, Xu (2014); Gallicchio, Scardapane (2020); Calabrò, Fabiani, Siettos (2021); Ni, Dong (2023); Wang, Dong (2024)

Domain decomposition for neural networks and randomized neural networks

- cPINNs, XPINNs: Jagtap, Kharazmi, Karniadakis (2020); Jagtap, Karniadakis (2020)
- Schwarz it. for PINNs or DeepRitz (D3M, DeepDDM, etc):: Li, Tang, Wu, Liao (2019); Li, Xiang, Xu (2020); Mercier, Gratton, Boudier (arXiv 2021); Dolean, H., Mercier, Gratton (arXiv 2024); Li, Wang, Cui, Xiang, Xu (2023); Sun, Xu, Yi (arXiv 2023, 2024); Kim, Yang (2023, 2024, 2024)
- FBPINNs, FBKANs: Moseley, Markham, Nissen-Meyer (2023); Dolean, H., Mishra, Moseley (2024, 2024); H., Howard, Beecroft, Stinis (acc. 2024); Howard, Jacob, Murphy, H., Stinis (arXiv 2024)
- DD for RaNNs, ELMS, Random Feature Method: Dong, Li (2021); Dang, Wang (2024); Sun, Dong, Wang (2024); Sun, Wang (2024); Chen, Chi, E, Yang (2022); Shang, H., Mishra, Wang (subm. 2024)

An overview of the state-of-the-art in early 2021:

A. Heinlein, A. Klawonn, M. Lanser, J. Weber

Combining machine learning, 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, domain decomposition methods – a survey

Computational Science, Engineering. 2024

Domain Decomposition-Based PIRaNNs

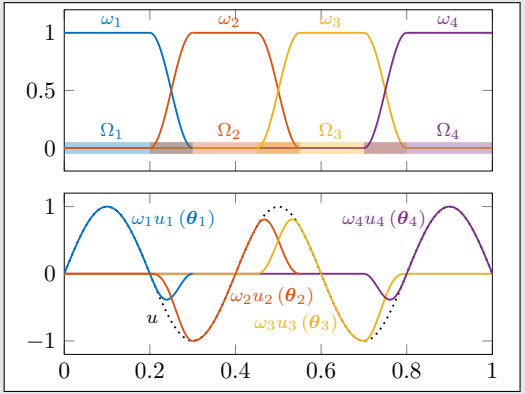
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

$$u(\theta_1, \dots, \theta_J) = \sum_{j=1}^{J} \omega_j u_j(\theta_j)$$

and the **loss function**

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

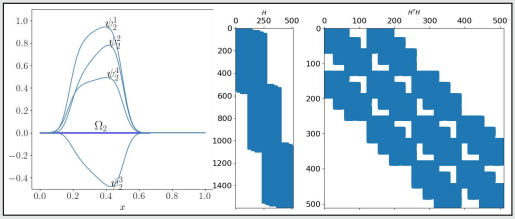


Domain decomposition for RaNNs

We employ the FBPINNs approach; cf. **Shang, Heinlein, Mishra, Wang (subm. 2024)**. This is closely related to the **random feature method (RFM)** by **Chen, Chi, E, Yang (2022)**. In particular, we solve

$$\mathcal{A}[\sum_{j=1}^{J} \omega_j u_j(A_j)](x_i) = f(x_i),$$

for $i = 1, \dots, N_{\text{PDE}}$; the boundary condtions are incorporated directly into the u_j.

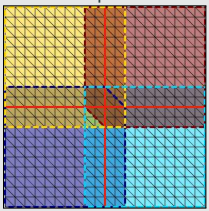


The hidden weights are randomly initialized, the resulting matrices H and $H^\top H$ are block-sparse.

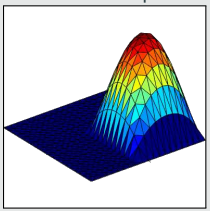
Preconditioning for Domain Decomposition-Based PIRaNNs

One-level Schwarz preconditioner

Overlap $\delta = 1h$



Solution of local problem



Based on an **overlapping domain decomposition**, we define a **one-level Schwarz operator** for $\boldsymbol{K} := \boldsymbol{H}^\top \boldsymbol{H}$

$$\boldsymbol{M}_{\text{OS-1}}^{-1}\boldsymbol{K} = \sum_{i=1}^{N} \boldsymbol{R}_i^\top \boldsymbol{K}_i^{-1} \boldsymbol{R}_i \boldsymbol{K},$$

where $\boldsymbol{R}_i$ and $\boldsymbol{R}_i^\top$ are restriction and prolongation operators corresponding to Ω_i', and $\boldsymbol{K}_i := \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^\top$.

Here, the matrix $\boldsymbol{K}_i$ could be singular in which case we use a **pseudo inverse** $\boldsymbol{K}_i^+$ instead of $\boldsymbol{K}_i^{-1}$.

We also consider **restricted and scaled additive Schwarz preconditioners**; cf. **Cai, Sarkis (1999)**.

Singular Value Decomposition

As discussed before, on each subdomain Ω_j, the RaNN is

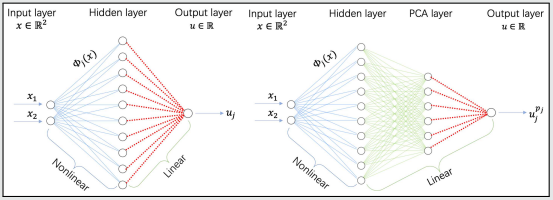
$$\begin{aligned} u_j(\boldsymbol{x}, \boldsymbol{A}_j) &= F_{L+1}^{\boldsymbol{A}} \cdot F_L^{W_L, b_L} \circ \ldots \circ F_1^{W_1, b_1}(\boldsymbol{x}) \\ &= \boldsymbol{A}_j \begin{bmatrix} \Phi_1(\boldsymbol{x}) & \cdots & \Phi_k(\boldsymbol{x}) \end{bmatrix}^\top, \end{aligned}$$

where k is the width of the last hidden layer and the Φ_k are the randomized basis functions.

Consider a **reduced SVD** $\boldsymbol{\Phi} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, where the entries of the matrix are $\boldsymbol{\Phi}_{i,k} = \Phi_k(\boldsymbol{x}_i)$. Then, we consider

$$\hat{u}_j(\boldsymbol{x}, \boldsymbol{A}_j) = \boldsymbol{A}_j \hat{\boldsymbol{V}}^\top \begin{bmatrix} \Phi_1(\boldsymbol{x}) & \cdots & \Phi_k(\boldsymbol{x}) \end{bmatrix}^\top,$$

where $\hat{\boldsymbol{V}}^\top$ is obtained by omitting the right singular vectors corresponding to small singular values.



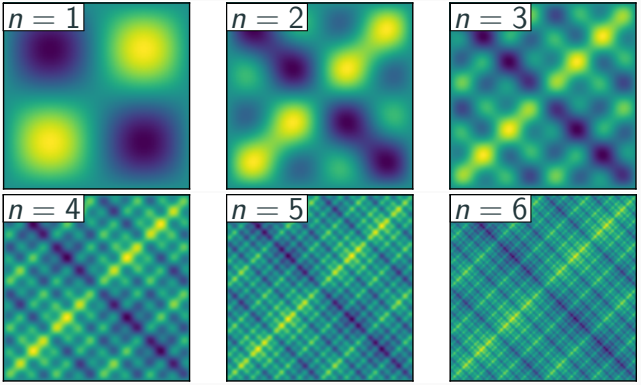
Multi-Frequency Problem

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

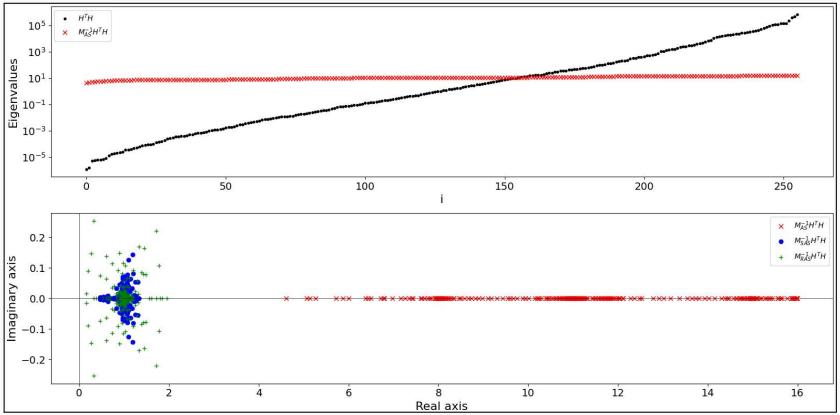
$$
\begin{aligned}
-\Delta u &= 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) && \text{in } \Omega = [0,1]^2, \\
u &= 0 && \text{on } \partial\Omega,
\end{aligned}
$$

with $\omega_i = 2^i$.

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



Numerical Results for the Multi-Frequency Problem ($n = 2$)



	$M^{-1} = I$		$M^{-1} = M_{AS}^{-1}$		$M^{-1} = M_{RAS}^{-1}$		$M^{-1} = M_{SAS}^{-1}$	
	iter	e_{L^2}	iter	e_{L^2}	iter	e_{L^2}	iter	e_{L^2}
CG	> 2000	$1.95{\cdot}10^{-2}$	8	$5.03{\cdot}10^{-3}$	—	—	—	—
CGS	> 2000	$2.63{\cdot}10^{-2}$	4	$5.04{\cdot}10^{-3}$	24	$5.03{\cdot}10^{-3}$	6	$5.04{\cdot}10^{-3}$
BICG	> 2000	$1.03{\cdot}10^{-2}$	8	$5.08{\cdot}10^{-3}$	32	$5.05{\cdot}10^{-3}$	11	$5.09{\cdot}10^{-3}$
GMRES	> 2000	$8.68{\cdot}10^{-2}$	13	$5.07{\cdot}10^{-3}$	31	$5.06{\cdot}10^{-3}$	11	$5.08{\cdot}10^{-3}$

4×4 subdomains; DoF $= 256$; $N = 1600$; $\boldsymbol{\theta}^0 \mathcal{U}(-1, 1)$; stop.: $\|\boldsymbol{M}^{-1}\boldsymbol{r}^k\|_{L^2} / \|\boldsymbol{M}^{-1}\boldsymbol{r}^0\|_{L^2} \leq 10^{-5}$

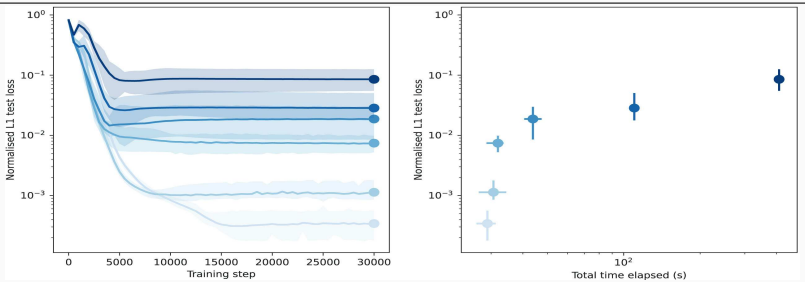
Numerical Results for the Multi-Frequency Problem ($n = 2$) – Effect of the SVD

We now investigate the effect of omitting right singular vectors associated with singular values below a varying tolerance τ.

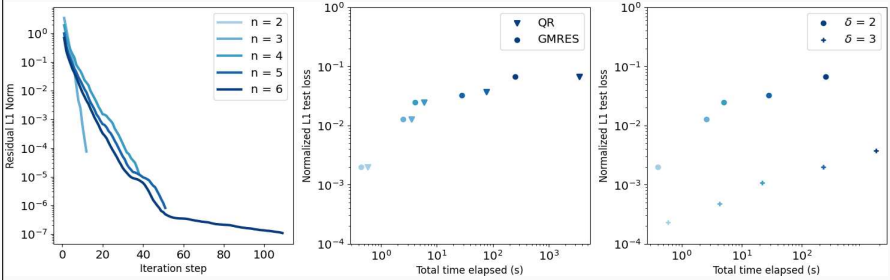
τ	DoF	M^{-1}	σ_{min}	σ_{max}	iter	e_{L^2}
10^{-4}	512	*None*	10^{-10}	10^{6}	> 2000	3.72e-2
		M_{AS}^{-1}	10^{-6}	10^{6}	27	5.46e-5
		M_{SAS}^{-1}	10^{-7}	10^{5}	30	5.49e-5
10^{-3}	436	*None*	10^{-8}	10^{5}	> 2000	3.75e-2
		M_{AS}^{-1}	10^{-5}	10^{5}	16	1.28e-4
		M_{SAS}^{-1}	10^{-6}	10^{4}	18	1.28e-4
10^{-2}	335	*None*	10^{-5}	10^{5}	> 2000	4.51e-2
		M_{AS}^{-1}	10^{-3}	10^{4}	14	7.14e-4
		M_{SAS}^{-1}	10^{-4}	10^{3}	13	7.11e-4
10^{-1}	212	*None*	10^{-3}	10^{6}	> 2000	5.01e-2
		M_{AS}^{-1}	10^{-2}	10^{3}	12	7.13e-3
		M_{SAS}^{-1}	10^{-3}	10^{2}	11	7.10e-3

4×4 subdomains; $N = 1600$; $\boldsymbol{\theta}^0$; $\mathcal{U}(-1, 1)$; stop.: $\|\boldsymbol{M}^{-1}\boldsymbol{r}^k\|_{L^2} / \|\boldsymbol{M}^{-1}\boldsymbol{r}^0\|_{L^2} \leq 10^{-5}$

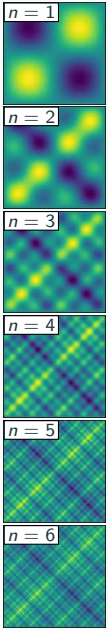
Numerical Results for the Multi-Frequency Problem



Multi-level FBPINNs; cf. Dolean, Heinlein, Mishra, Moseley (2024)



DD-PIRaNNs; cf. Shang, Heinlein, Mishra, Wang (subm. 2024)

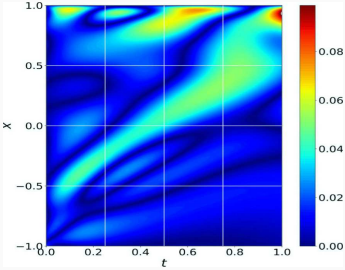


Numerical Results for 1D Advection-Diffusion Problem

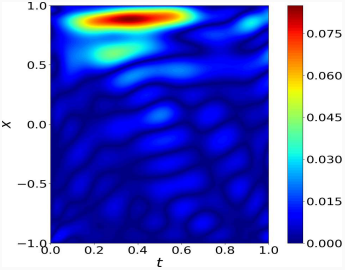
Given $\Omega = (-1, 1)$, we consider a **one-dimensional advection-diffusion equation**

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} &= \kappa \frac{\partial u^2}{\partial x^2} && \text{in } \Omega \times I, \\ u(-1, t) = u(1, t) &= 0 && \text{in } I, \\ u(x, 0) &= -\sin(\pi x) && \text{in } \Omega, \end{aligned}$$

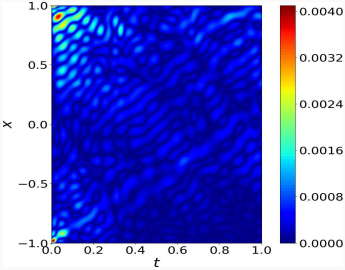
with $I = (0, 1)$. Here, $\kappa = 0.1/\pi$ is the diffusivity coefficient, complicating the solution near $x = 1$ due to the no-slip boundary; see **Kharazmi, Zhang, Karniadakis (2021)**.



hp-**VPINNs**: error results from **Kharazmi, Zhang, Karniadakis (2021)**



DD-PIRaNNs (weak bc. enf.): error; relative L^2 error of 2.88×10^{-2}; 0.12 s



DD-PIRaNNs (hard bc. enf.): error; relative L^2 error of 7.40×10^{-4}; 0.08 s

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!

Summary

Physics-informed neural networks and randomized neural networks

- **PINNs** incorporate physics the loss, improving accuracy and generalization. **RaNNs** use random weights and biases, while only training the last layer. This can **simplify the training** and **reduce computational cost**.
- Nonetheless, both approaches suffer from **ill-conditioning** leading to **a challenging training process**.

Domain decomposition architectures and preconditioning

- **Domain decomposition-based architectures** **improve the scalability of PINNs** to **large domains** / **high frequencies**, **keeping the complexity of the local networks low**.
- **Preconditioning**, using a combination of **Schwarz preconditioning** and **SVD**, can **improve the conditioning of the RaNN problem** and **significantly reduce the number of iterations** needed for convergence.

Thank you for your attention!

Topical Activity Group

Scientific Machine Learning