FAST MULTISCALE PDE SOLVERS VIA MULTILEVEL DOMAIN DECOMPOSITION AND RANDOM FEATURES

Eray Yıldız[1*], **Chinmay Datar**[1,3,4], **Atamert Rahma**[1,3], **Victorita Dolean**[2], **Felix Dietrich**[1,3]
[1]Technical University of Munich [2]Eindhoven University of Technology
[3]Munich Center for Machine Learning [4]TUM Institute for Advanced Study

ABSTRACT

Solving multiscale partial differential equations (PDEs) accurately and efficiently remains a major challenge in computational science. Physics-informed neural networks (PINNs) have shown significant promise for solving PDEs, yet their application to multiscale problems remains limited by spectral bias and the slow convergence of gradient-descent-based optimization. Multilevel domain decomposition methods mitigate spectral bias in PINNs but leave the slow and expensive gradient-descent-based training largely unaddressed. In this work, we present a principled framework that integrates multilevel domain decomposition into random feature methods, enabling efficient, gradient-descent-free training of neural PDE solvers. We empirically demonstrate that the proposed method substantially accelerates training while preserving solution accuracy on challenging multiscale PDEs. By bridging classical multilevel methods and random-feature methods, our work opens the door to rapidly trainable, accurate neural PDE solvers for multiscale PDEs, challenging the dominance of gradient-descent-based methods.

1 INTRODUCTION

Efficient and accurate partial differential equation (PDE) solvers underpin countless scientific and engineering applications. Deep neural networks have recently gained traction for solving PDEs because of their mesh-free representations and capacity to approximate high-dimensional functions (Dissanayake & Phan-Thien, 1994; Lagaris et al., 1998; E & Yu, 2018; Han et al., 2018; Raissi et al., 2019; Zang et al., 2020), supported by powerful automatic differentiation frameworks such as PyTorch (Paszke et al., 2017) and TensorFlow (Abadi et al., 2015). In particular, Physics-informed neural networks, which integrate physical laws directly into the learning process (Karniadakis et al., 2021; Sirignano & Spiliopoulos, 2018) have played a central role in the newly emerging field of Scientific Machine Learning (SciML) (Dietrich & Schilders, 2025)This study targets two key challenges impacting the accuracy and training time of neural PDE solvers:

1. Challenges posed by training of neural PDE solvers: Despite extensive development, neural PDE solvers still struggle to match the efficiency and accuracy of classical numerical methods (Chen et al., 2025). In particular, the reliance on iterative, gradient-descent-based optimization over high-dimensional parameter spaces incurs long training times and costly hyperparameter searches (Huang et al., 2006; Werbos, 1974; Markidis, 2021). Random feature methods like Extreme learning machines (ELMs) address this bottleneck by sampling hidden-layer parameters from data-agnostic distributions and computing output weights via a single least-squares solve, eliminating the need for gradient descent (Huang et al., 2006; Rahimi & Recht, 2007). Extensions such as Sample Where It Matters (SWIM) networks further improve approximation by employing data-dependent sampling strategies while retaining non-iterative training Bolager et al. (2023). SWIM networks have recently been incorporated into physics-informed neural networks, achieving significant improvements in both training speed and solution accuracy (Datar et al., 2024; Rahma et al., 2025; 2024). However, spectral bias and limited capacity prevent single global networks from resolving high-frequency spatial variations in PDE solutions, leading to low-frequency components being learned preferentially over high-frequency features (Wang et al., 2021; Cao et al., 2020; Rahaman et al., 2019).

*Corresponding author, eray.yldz@tum.de

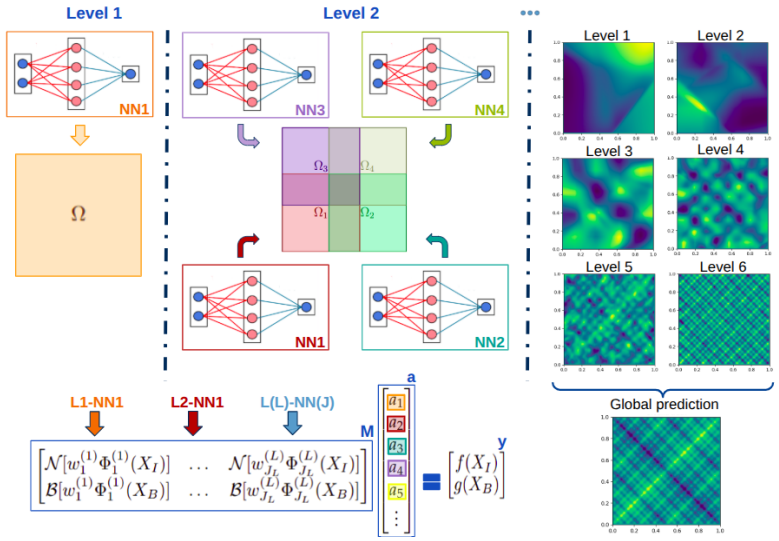


Figure 1: Overview of our neural PDE solver integrating two core ideas: (1) multi-level domain decomposition (single domain (left), four subdomains (middle)) with (2) gradient-descent-free Frozen-PINN training via a single least-squares solve to obtain the coefficients a. M encodes PDE/boundary evaluations, a the unknowns, and y the forcing/boundary data. Network predictions at each level and the ground truth are shown on the right.

2. Challenges posed by multi-scale PDEs: Accurately capturing multiscale PDE solutions amplifies the effects of spectral bias (Moseley et al., 2023). Domain decomposition methods mitigate this issue by replacing a single global network with multiple localized models, improving scalability and convergence (Heinlein et al., 2021), as exemplified by FBPINNs and their multilevel extensions (Moseley et al., 2023; Dolean et al., 2024). However, these approaches remain computationally expensive due to iterative gradient-descent-based optimization (Anderson et al., 2024). Random feature methods such as ELM-FBPINNs enable fast training through least-squares solvers and partition-of-unity (PoU)(Anderson et al., 2024; Chen et al., 2024; van Beek et al., 2026). While multilevel domain decomposition can implicitly enforce frequency separation, where coarse levels capture low-frequency components and finer levels enrich localized high-frequency structure to mitigate spectral bias, existing random feature approaches are largely restricted to single- or two-level decompositions (Sun & Chen, 2025).

The limited exploration of random feature methods with multilevel domain decomposition motivates the core questions we address in this work:

- (RQ1) Does multilevel domain decomposition improve accuracy over single-level methods, and how does the prediction error scale with the number of levels and neurons?
- (RQ2) Do random feature methods enable faster training than gradient-descent–based PINNs when both are combined with multilevel domain decomposition?
- (RQ3) How does domain decomposition affect the conditioning and singular value decay of the random-feature matrix compared to a formulation without domain decomposition?

To this end, the key contributions of our work are:

1. We propose *Frozen-PINN-DD* and *Frozen-PINN-MLDD* (see Figure 1), which incorporate single-level and multilevel domain decomposition, respectively, into Frozen-PINNs by coupling independent neural networks via partition-of-unity schemes (Lee, 2003).
2. We empirically show that hierarchical domain decomposition alleviates spectral bias and directly extends Frozen-PINNs to multiscale PDEs, dramatically reducing training time for comparable accuracy, outperforming classical PINNs trained with gradient-descent-based methods and competing random-feature methods on several multiscale PDE benchmarks.

3. We investigate how domain decomposition enhances conditioning and increases the singular values of the random-feature matrix, offering insight into why it enables the accurate representation of complex multiscale solutions.

2 METHOD

Problem setup: We aim to compute the solution $u\colon \Omega \times \mathbb{R} \to \mathbb{R}$ of linear static PDEs of the form

$$\mathcal{N}[u(\mathbf{x})] = f(\mathbf{x}), \quad \mathbf{x} \in \Omega \subset \mathbb{R}^d, \quad f\colon \Omega \to \mathbb{R}, \tag{1}$$

$$\mathcal{B}_k[u(\mathbf{x})] = g_k(\mathbf{x}), \quad \mathbf{x} \in \Gamma_k \subset \partial\Omega, \quad g_k\colon \partial\Omega \to \mathbb{R}, \tag{2}$$

where $\mathcal{N}$ and $\mathcal{B}_k$ are linear operators, f and g_k represent the forcing and boundary condition, respectively and k indexes the boundary condition defined on the corresponding boundary $\Gamma_k \subset \partial\Omega$. Datar et al. (2024) propose using random features and fixing the single hidden layer of a neural network to approximate the target PDE. For the static case in Equations (1) and (2), their Ansatz is

$$\hat{u}_{\text{FROZEN-PINN}}(\mathbf{x}) = A\sigma(Wx + b), \tag{3}$$

where $W \in \mathbb{R}^{C\times d}, b \in \mathbb{R}^C$ are random features (e.g. using ELM or SWIM), and σ is `tanh` activation function. Given data $X \in \mathbb{R}^{N\times d}$ consisting of interior points X_I and boundary points X_B, the linear output layer weights $A \in \mathbb{R}^{1\times C}$ are obtained by solving the least-squares problem $\min_A \|MA^\top - y\|_2^2$, where $M = [\mathcal{N}[\Phi(X_I)] \quad \mathcal{B}[\Phi(X_B)]]^T \in \mathbb{R}^{N\times C}$ is the random feature matrix, $y = [f(X_I) \quad g(X_B)]^T \in \mathbb{R}^{N\times 1}$ is the ground truth, and $\Phi(z) = \sigma(zW^\top + b)$ is the hidden-layer feature map.

Frozen-PINN-MLDD: Given a partitioned domain $\{\Omega_j^{(l)}\}_{j=1}^{J_l}$ for each level $l \in \{1, \dots, L\}$, where Ω is decomposed into J_l overlapping subdomains, we apply the Frozen-PINN ansatz (eq. (3)) on each subdomain $\Omega_j^{(l)}$ and construct the global solution as

$$\hat{u}_{\text{FROZEN-PINN-MLDD}}(\mathbf{x}) = \sum_{\ell=1}^{L}\sum_{j=1}^{J_\ell} w_j^{(\ell)}(\mathbf{x})\,\hat{u}_j^{(\ell)}(\mathbf{x}), \quad \sum_{j=1}^{J_\ell} w_j^{(\ell)}(\mathbf{x}) = 1, \qquad \forall\, \mathbf{x} \in \Omega, \tag{4}$$

where $\hat{u}_j^{(l)}, w_j^{(l)}(x)$ represent a Frozen-PINN Ansatz and the corresponding window function for the j-th subdomain at level l, respectively. Note that the window functions satisfy the partition of unity (Lee, 2003) independently for each level $l = 1, \dots L$. We then jointly solve for all the output layer parameters

$$\underset{A_1,\dots,A_J}{\arg\min} \|M\,[A_1 \quad \dots \quad A_J]^T - y\|_2^2 \quad , \tag{5}$$

where

$$M = \begin{bmatrix} \mathcal{N}[w_1^{(1)}\Phi_1^{(1)}(X_I)] & \dots & \mathcal{N}[w_{J_L}^{(L)}\Phi_{J_L}^{(L)}(X_I)] \\ \mathcal{B}[w_1^{(1)}\Phi_1^{(1)}(X_B)] & \dots & \mathcal{B}[w_{J_L}^{(L)}\Phi_{J_L}^{(L)}(X_B)] \end{bmatrix} \in \mathbb{R}^{N\times JC}, y = \begin{bmatrix} f(X_I) \\ g(X_B) \end{bmatrix} \in \mathbb{R}^{N\times 1}. \tag{6}$$

We now have $J = \sum_{l=1}^{L} J_l$ Frozen-PINNs, rather than a single network as in Equation (3). We refer to the method as *Frozen-PINN-DD* for $L = 1$ and *Frozen-PINN-MLDD* for $L > 1$, where "ML" denotes "multi-level" and "DD" denotes "domain decomposition" (Table 2). The window functions and domain decomposition strategies are summarized in Appendix B.

3 EMPIRICAL RESULTS

In the first experiment, we consider the homogeneous Laplace problem adapted from Dolean et al. (2024) in Appendix C.1 (Figure 4), and observe an approximately one-order-of-magnitude decrease in normalized MAE per added decomposition level, and an exponential decay of error with respect to the number of basis functions in each network, answering (RQ1). For detailed ablations on the number of levels, the overlap ratio, and the capacity of sub-networks, please refer to Appendix C.1.

In the second experiment, we study increasingly multi-scale solutions given by $u(x_1, x_2) = \frac{1}{n}\sum_{i=1}^{n} \sin(\omega_i \pi x_1)\sin(\omega_i \pi x_2)$, for the Poisson problem, with n and ω_i controlling the number

and frequency of modes. Following the weak-scaling setup of Dolean et al. (2024), we simultaneously increase problem complexity (i.e., by increasing n), collocation points $(10 \cdot 2^{L-1})^2$, and model capacity. In Figure 2 and Table 7, we observe that Frozen-PINNs with random features achieve $120 - 184$ times faster training than gradient-descent–trained finite-basis PINNs at comparable accuracy, answering (RQ2).

The third experiment (Appendix C.2.2, Figure 5) demonstrates that multi-level decomposition slows the decay of singular values compared to single-level setups, preserving more random-feature information. Consequently, for a fixed SVD truncation ratio, the least-squares solution for last-layer coefficients attains higher accuracy on multi-scale problems, answering (RQ3).

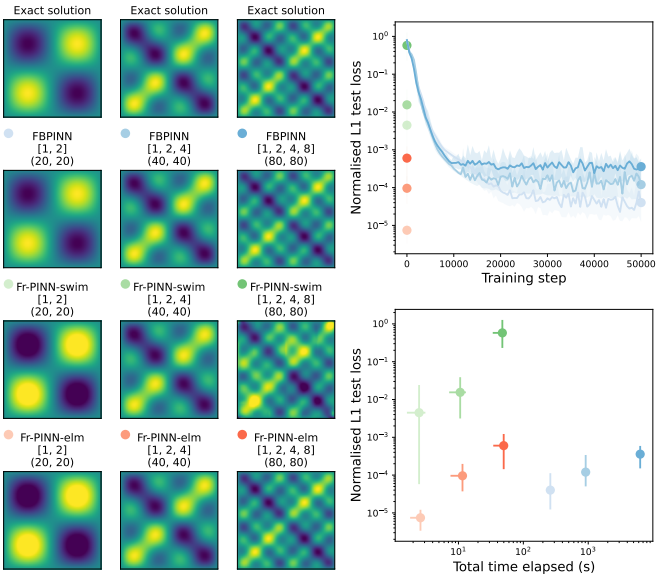


Figure 2: Weak scaling study for a multi-scale Laplacian problem: true solution with increasing frequency components (1rs row), multilevel FBPINN (2nd row), Frozen-PINN-swim (3rd row), and Frozen-PINN-elm (4th row). Titles specify the level structure and the number of collocation points. The loss curve (top right) and training time (bottom right) show that Frozen-PINNs, especially with ELM, achieve higher accuracy and faster training (see also Table 1).

4 Conclusion

We introduce Frozen-PINN-DD and Frozen-PINN-MLDD, integrating single- and multilevel domain decomposition into Frozen-PINNs via partition-of-unity coupling. We reveal that multilevel domain decomposition combined with random feature methods enables rapid training of PINNs, achieving speedups of more than 100× for multi-scale PDEs, avoiding the accuracy bottlenecks and the high computational cost associated with gradient-descent-based training and mitigating spectral bias. We empirically demonstrate that multi-level decomposition consistently outperforms single-level by preserving more information in the random feature representation, as evidenced by slower singular value decay, enabling higher-accuracy solutions for multi-scale problems.

Limitations and future work: Our current study is limited to static PDEs and to relatively small systems that are computationally tractable on a standard desktop. Extending our methods to time-dependent PDEs is a natural next step, leveraging classical ODE solvers with adaptive time-stepping and the idea of space-time separation (Datar et al., 2024). Scaling to larger, multi-scale PDEs across larger domains and more decomposition levels is another important direction that could exploit high-performance computing and scalable iterative solvers like LSRN. Finally, improving system conditioning remains an open challenge, for which advances in preconditioning for random-feature methods may be beneficial Meng et al. (2014); van Beek et al. (2026). To conclude, this work paves the way for fast, scalable neural PDE solvers for complex multi-scale PDEs, without relying on gradient descent or specialized hardware.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

Samuel Anderson, Victorita Dolean, Ben Moseley, and Jennifer Pestana. Elm-fbpinn: efficient finite-basis physics-informed neural networks, 2024. URL https://arxiv.org/abs/2409.01949.

Erik Lien Bolager, Iryna Burak, Chinmay Datar, Qing Sun, and Felix Dietrich. Sampling weights of deep neural networks, 2023. URL https://arxiv.org/abs/2306.16830.

Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning, 2020. URL https://arxiv.org/abs/1912.01198.

Feng Chen, Yiran Meng, Kegan Li, Chaoran Yang, and Jiong Yang. Global physics-informed neural networks (gpinns): from local point-wise constraint to global nodal association, 2025. URL https://arxiv.org/abs/2503.06403.

Jingrun Chen, Weinan E, and Yifei Sun. Optimization of random feature method in the high-precision regime. Communications on Applied Mathematics and Computation, 6(2):1490–1517, 2024. doi: 10.1007/s42967-024-00389-8. URL https://link.springer.com/article/10.1007/s42967-024-00389-8.

Chinmay Datar, Taniya Kapoor, Abhishek Chandra, Qing Sun, Iryna Burak, Erik Lien Bolager, Anna Veselovska, Massimo Fornasier, and Felix Dietrich. Solving partial differential equations with sampled neural networks. arXiv preprint arXiv:2405.20836, 2024.

Felix Dietrich and Wil Schilders. Scientific machine learning. Mathematische Semesterberichte, 72 (2):89–115, 2025.

MWMG Dissanayake and Nhan Phan-Thien. Neural-network-based approximations for solving partial differential equations. communications in Numerical Methods in Engineering, 10(3):195–201, 1994.

Victorita Dolean, Alexander Heinlein, Siddhartha Mishra, and Ben Moseley. Multilevel domain decomposition-based architectures for physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering, 429:117116, September 2024. ISSN 0045-7825. doi: 10.1016/j.cma.2024.117116. URL http://dx.doi.org/10.1016/j.cma.2024.117116.

Weinan E and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. Communications in Mathematics and Statistics, 6(1):1–12, 2018.

Jie Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. Proceedings of the National Academy of Sciences of the United States of America, 115(34):8505–8510, 2018.

Alexander Heinlein, Axel Klawonn, Martin Lanser, and Janine Weber. Combining machine learning and domain decomposition methods for the solution of partial differential equations—a review. GAMM-Mitteilungen, 44(1):e202100001, March 2021. ISSN 0936-7195. doi: 10.1002/gamm.202100001. URL https://onlinelibrary.wiley.com/doi/10.1002/gamm.202100001.

Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. Neurocomputing, 70(1):489–501, 2006. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2005.12.126. URL https://www.sciencedirect.com/science/article/pii/S0925231206000385. Neural Networks.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. Nature Reviews Physics, 3:422–440, 2021. doi: 10.1038/s42254-021-00314-5.

I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. IEEE Transactions on Neural Networks, 9(5):987–1000, 1998. ISSN 1045-9227. doi: 10.1109/72.712178. URL http://dx.doi.org/10.1109/72.712178.

John M. Lee. Introduction to Smooth Manifolds, volume 218 of Graduate Texts in Mathematics. Springer, New York, NY, 2003. ISBN 978-0-387-21752-9. doi: 10.1007/978-0-387-21752-9.

Stefano Markidis. The old and the new: Can physics-informed deep-learning replace traditional linear solvers?, 2021. URL https://arxiv.org/abs/2103.09655.

Nick McGreivy and Ammar Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. Nature machine intelligence, 6 (10):1256–1269, 2024.

Xiangrui Meng, Michael A Saunders, and Michael W Mahoney. Lsrn: A parallel iterative solver for strongly over-or underdetermined systems. SIAM Journal on Scientific Computing, 36(2): C95–C118, 2014.

Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Finite basis physics-informed neural networks (fbpinns): a scalable domain decomposition approach for solving differential equations. Advances in Computational Mathematics, 49(4), July 2023. ISSN 1572-9044. doi: 10.1007/s10444-023-10065-9. URL http://dx.doi.org/10.1007/s10444-023-10065-9.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch, 2017.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2019. URL https://arxiv.org/abs/1806.08734.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis (eds.), Advances in Neural Information Processing Systems, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf.

Atamert Rahma, Chinmay Datar, and Felix Dietrich. Training hamiltonian neural networks without backpropagation. arXiv preprint arXiv:2411.17511, 2024.

Atamert Rahma, Chinmay Datar, Ana Cukarska, and Felix Dietrich. Rapid training of hamiltonian graph networks without gradient descent. arXiv preprint arXiv:2506.06558, 2025.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Justin A. Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. Journal of Computational Physics, 375:1339–1364, 2018.

Yifei Sun and Jingrun Chen. Two-level random feature methods for elliptic partial differential equations over complex domains. Computer Methods in Applied Mechanics and Engineering, 441: 117961, 2025.

Jan Willem van Beek, Victorita Dolean, and Ben Moseley. Local feature filtering for scalable and well-conditioned domain-decomposed random feature methods. Computer Methods in Applied Mechanics and Engineering, 449:118583, 2026. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2025.118583. URL https://www.sciencedirect.com/science/article/pii/S0045782525008552.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering, 384:113938, October 2021. ISSN 0045-7825. doi: 10.1016/j.cma.2021.113938. URL http://dx.doi.org/10.1016/j.cma.2021.113938.

Paul John Werbos. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.d. dissertation, Harvard University, 1974.

Yuwei Zang, Gang Bao, Xiu Ye, and Hongkai Zhou. Weak adversarial networks for high-dimensional partial differential equations. Journal of Computational Physics, 411:109409, 2020.

A NOTATION

Table 1: Model acronyms used in the study.

ELM	Extreme Learning Machines
SWIM	Sample Where It Matters networks
PINN	Physics-Informed Neural Networks
FBPINN	Finite Basis Physics-Informed Neural Networks
Frozen-PINN	Frozen Physics-Informed Neural Networks
Frozen-PINN-elm	ELM Sampled Frozen Physics-Informed Neural Networks
Frozen-PINN-swim	SWIM Sampled Frozen Physics-Informed Neural Networks
Frozen-PINN-elm-DD	ELM Sampled One-level Domain Decomposed Frozen-PINN
Frozen-PINN-swim-DD	SWIM Sampled One-level Domain Decomposed Frozen-PINN
Frozen-PINN-elm-MLDD	ELM Sampled Multilevel Domain Decomposed Frozen-PINN
Frozen-PINN-swim-MLDD	SWIM Sampled Multilevel Domain Decomposed Frozen-PINN

Table 2: Notation used in Section 2 and in the appendix.

Problem setup	
N	Number of data points
d	Size of spatial dimension
Ω	Spatial domain
$\partial\Omega$	Boundary set of the spatial domain
Γ_k	Boundary domain for the boundary k
u	Ground truth global solution of a PDE
$\mathcal{N}$	Linear operator for definition of the interior governing equation
$\mathcal{B}_k$	linear boundary operator corresponding k-th boundary segment
$f\colon \Omega \to \mathbb{R}$	Forcing term
$g_k\colon \Gamma_k \to \mathbb{R}$	Boundary term
Frozen-PINN-MLDD setup	
$X \in \mathbb{R}^{\mathbf{N}\times\mathbf{d}}$	Dataset inputs
$X_I \subset X \in \mathbb{R}^{N_I\times d}$	Dataset inputs corresponding to interior domain
N_I	Number of interior input points in the dataset
$X_B \subset X \in \mathbb{R}^{N_B\times d}$	Dataset inputs corresponding to boundary domain
N_B	Number of boundary input points in the dataset
$y \in \mathbb{R}^{\mathbf{N}\times\mathbf{1}}$	Dataset targets
$\sigma = \tanh\colon \mathbb{R} \to \mathbb{R}$	Nonlinear smooth activation function applied element-wise
C	Number of neurons in a single network
L	Number of levels
J_l	Number of subdomains at level l
$J = \sum_{l=1}^{L} J_l$	Total number of subdomains given L levels
$\Omega_j^{(l)}$	Subdomain j at level l
$w_j^{(l)}\colon \Omega \to \mathbb{R}$	Window function for subdomain j at level l
$f(X_I) \in \mathbb{R}^{N_I\times 1}$	Forcing term evaluated at all interior points
$g(X_B) \in \mathbb{R}^{N_B\times 1}$	Boundary condition evaluated at all boundary points

B METHODOLOGICAL DETAILS

The Frozen-PINN framework encompasses various methodologies for solving different types of PDE problems, from solving linear static PDEs to solving inverse PDE problems Datar et al. (2024). In this study, we focus on the simplest variant, which is solving linear static PDEs, explain how

standard Frozen-PINN models are used to approximate their solutions, and describe how the Frozen-PINN framework is extended using domain decomposition techniques to develop both one-level and multilevel Frozen-PINN models.

B.1 FROZEN-PINN FRAMEWORK FOR SOLVING LINEAR STATIC PDES

Frozen-PINN models Datar et al. (2024) use a SWIM neural network Bolager et al. (2023) with one hidden layer to parameterize the approximation of the PDE solution. The hidden layer parameters are initialized using sampling heuristics and kept fixed during the training. Only the final layer weights and bias are trained Bolager et al. (2023). The solution approximation of the SWIM network defined as in Equation (3). *tanh* basis function is used in Frozen-PINN due to its infinite differentiability property. Since the operator evaluations require higher-order derivative evaluations of the basis functions due to the chain rule, the tanh function can be used as a basis function for operators of any differential degree.

Loss function: The Frozen-PINN model treats the interior and boundary points in the problem domain differently during their training phase. For interior points, the model evaluates the differential operator defined by the governing PDE as in Equation Equation (1), while for boundary points, it evaluates the boundary condition operator as defined in Equation Equation (2) Datar et al. (2024). Based on this methodology, the Frozen-PINN approximant is trained minimizing the following loss function definition

$$\mathcal{L}(\mathbf{a}) = \mathcal{L}_{\text{interior}}(\mathbf{a}) + \mathcal{L}_{\text{boundary}}(\mathbf{a}), \tag{7}$$

where the loss functions for interior and boundary point evaluations are

$$\mathcal{L}_{\text{interior}}(\mathbf{a}) = \frac{1}{2}\sum_{i=1}^{N_{\text{int}}} \left(\mathcal{N}[u_{\text{FROZEN-PINN}}(\mathbf{x}_i^{\text{int}}, \mathbf{a})] - f(\mathbf{x}_i^{\text{int}})\right)^2, \tag{8}$$

$$\mathcal{L}_{\text{boundary}}(\mathbf{a}) = \frac{1}{2}\sum_{i=1}^{N_{\text{bdry}}} \left(\mathcal{B}[u_{\text{FROZEN-PINN}}(\mathbf{x}_i^{\text{bdry}}, \mathbf{a})] - h(\mathbf{x}_i^{\text{bdry}})\right)^2. \tag{9}$$

As can be seen from Equation Equation (7), the same set of trainable coefficients "$\mathbf{a}$" is used for both loss functions that contain interior and boundary point evaluations given in Equation Equation (8) and Equation Equation (9). This shared parameterization results in a joint minimization problem, where the total loss, comprising contributions from the interior residuals and boundary condition violations, is minimized simultaneously. The solution to this problem gives the optimal coefficients "$\mathbf{a}^*$" that best satisfy the governing PDE and boundary conditions across the domain. Since it is assumed that the operators are linear operators, the same minimization problem is equivalent to the linear least-squares problem and can be compactly formulated as given in Equation (5) and Equation (6). This structured linear system enables Frozen-PINN models to approximate solutions of linear static PDEs efficiently by solving a single least-squares problem, eliminating the need for iterative optimization. Pseudo code of this framework is given in Algorithm 1.

B.2 INTEGRATING DOMAIN DECOMPOSITION INTO SWIM-PDE FRAMEWORK

In this section, a detailed mathematical description of the proposed domain-decomposed Frozen-PINN models will be provided.

B.2.1 ONE-LEVEL FROZEN-PINN (FROZEN-PINN-DD) MODELS

Considering the problem domain Ω divided into J overlapping subdomains $\{\Omega_j\}_{j=1}^{J}$, the global solution $u_{\text{FROZEN-PINN-DD}}(\mathbf{x})$ of the one-level Frozen-PINN model can be expressed as a weighted sum of local solutions as

$$u_{\text{FROZEN-PINN-DD}}(\mathbf{x}) = \sum_{j=1}^{J} w_j(\mathbf{x}) u_j(\mathbf{x}), \tag{10}$$

where $u_j(\mathbf{x})$ is the local solution predicted by the SWIM network trained on subdomain Ω_j as defined in Equation 3, and $w_j(\mathbf{x})$ is a window function supported on Ω_j, satisfying the partition of

unity condition Lee (2003)

$$\sum_{j=1}^{J} w_j(\mathbf{x}) = 1, \quad \forall \mathbf{x} \in \Omega. \tag{11}$$

This formulation enables each local SWIM network to specialize in a specific region of the domain by confining its contribution through a window function. The global solution is then obtained by smoothly aggregating the local predictions using a weighted sum by corresponding window functions. The use of overlapping subdomains, together with window functions that satisfy the partition of unity condition, ensures both consistency and smoothness across subdomain boundaries.

The training of one-level Frozen-PINN models can still be performed by solving a single least-squares problem, as formulated in Equation (5). The only modification required is updating the random feature matrix $\mathbf{M}$, originally defined in Equation 6, as given in following formulation

$$\mathbf{M} = \begin{bmatrix} \mathcal{N}[w_1(\mathbf{x}_1^{\text{int}}) \cdot \sigma(\mathbf{w}_1^1 \cdot \mathbf{x}_1^{\text{int}} + b_1^1)] & \cdots & \mathcal{N}[w_j(\mathbf{x}_1^{\text{int}}) \cdot \sigma(\mathbf{w}_C^J \cdot \mathbf{x}_1^{\text{int}} + b_C^J)] \\ \vdots & \ddots & \vdots \\ \mathcal{B}[w_1(\mathbf{x}_N^{\text{bdry}}) \cdot \sigma(\mathbf{w}_1^1 \cdot \mathbf{x}_N^{\text{bdry}} + b_1^1)] & \cdots & \mathcal{B}[w_j(\mathbf{x}_N^{\text{bdry}}) \cdot \sigma(\mathbf{w}_C^J \cdot \mathbf{x}_N^{\text{bdry}} + b_C^J)] \end{bmatrix}, \tag{12}$$

by stacking operator evaluations weighted by corresponding window functions for each local network horizontally in matrix $\mathbf{M} \in \mathbb{R}^{(N \times (J \times C))}$ to build the full design matrix. Pseudo code for creating global design matrix $\mathbf{M}$ is equivalent to the case where we consider there is only one decomposed domain in Algorithm 2.

After solving the constructed least-squares problem, each found coefficient is distributed to the corresponding local SWIM network as trained outer linear layer weight values.

B.2.2 Multilevel Frozen-PINN (Frozen-PINN-MLDD) Models

For problems exhibiting rich multiscale behavior, the one-level domain decomposition approach can be extended to a multilevel framework by introducing a hierarchy of overlapping domain decompositions at different spatial resolutions.

Assuming there are L decomposition levels, in the first step of this multilevel approach, at each level ℓ, the domain Ω is partitioned into J_ℓ overlapping subdomains $\{\Omega_j^{(\ell)}\}_{j=1}^{J_\ell}$. After the partition, for each of the subdomains, a corresponding SWIM network is constructed, and parameter initialization is performed using the collocation points evaluations within the subdomain itself. Then, all SWIM networks are trained jointly on each subdomain $\Omega_j^{(\ell)}$, producing local predictions $u_j^{(\ell)}(\mathbf{x})$ as given in Equation Equation (3). Finally, the smooth global approximation is obtained by hierarchically aggregating contributions from all local SWIM network predictions by ensuring the partition of unity for each level as given in the Equation (4) such that each level independently satisfies the partition of unity Lee (2003).

The training of multilevel Frozen-PINN models can still be performed by solving a single least-squares problem similar to the one-level Frozen-PINN models, as defined in Equation (5). The only modification required is updating the random feature matrix $\mathbf{M}$ again, originally defined in Equation (6) by horizontally stacking the operator evaluations weighted by corresponding window functions for each local network into the matrix $\mathbf{M} \in \mathbb{R}^{(N \times (J \times C))}$ where $J = \sum_{\ell=1}^{L} J^{(\ell)}$ to build the full random feature matrix for solving the single least-squares problem to train the multilevel Frozen-PINN models. Pseudo code for creating global random feature matrix $\mathbf{M}$ is given in Algorithm 2.

By decomposing the domain from coarse to fine resolutions, multilevel Frozen-PINN models can better capture both global and local features of the solution. Coarse levels capture broad, low-frequency components, while finer levels refine the solution in regions requiring higher resolution. This multi-resolution formulation enhances the scalability, accuracy, and robustness of Frozen-PINN models when applied to complex multiscale PDEs.

B.2.3 Mathematical Details of Domain Partitioning

We consider a bounded hyperrectangular domain

$$\Omega = \prod_{d=1}^{D} [a_d, b_d] \subset \mathbb{R}^D. \tag{13}$$

At each level ℓ, the domain is decomposed uniformly into overlapping subdomains with $J^{(\ell)}$ partitions per spatial dimension. The overlap is controlled by a fixed overlap ratio $\delta > 0$.

One-dimensional decomposition: For a given level ℓ and dimension d, the one-dimensional subdomains are defined in normalized coordinates as

$$\Omega_j^{(\ell,d)} = \begin{cases} \left[\frac{1}{2} - \frac{\delta}{2},\ \frac{1}{2} + \frac{\delta}{2}\right], & J^{(\ell)} = 1, \\ \left[\frac{j-1}{J^{(\ell)}-1} - \frac{\delta}{2(J^{(\ell)}-1)},\ \frac{j-1}{J^{(\ell)}-1} + \frac{\delta}{2(J^{(\ell)}-1)}\right], & J^{(\ell)} > 1, \end{cases} \qquad j = 1, \ldots, J^{(\ell)}. \tag{14}$$

Mapping back to physical coordinates yields

$$x^{(d)} = a_d + \xi(b_d - a_d), \qquad \xi \in \Omega_j^{(\ell,d)}. \tag{15}$$

All subdomains have equal width $\frac{\delta}{J^{(\ell)}-1}(b_d - a_d)$, while adjacent centers are spaced by $\frac{1}{J^{(\ell)}-1}(b_d - a_d)$. Thus, for $\delta > 1$, neighboring subdomains overlap by a fixed relative fraction proportional to $\delta - 1$. Boundary subdomains are allowed to extend beyond $[a_d, b_d]$ to ensure symmetric overlap near the global boundary.

Multi-dimensional subdomains: Let $\mathbf{j} = (j_1, \ldots, j_D) \in \{1, \ldots, J^{(\ell)}\}^D$ be a multi-index. Each D-dimensional subdomain is defined as the Cartesian product

$$\Omega_{\mathbf{j}}^{(\ell)} = \prod_{d=1}^{D} \Omega_{j_d}^{(\ell,d)}. \tag{16}$$

This construction yields $(J^{(\ell)})^D$ overlapping hyperrectangular subdomains satisfying

$$\Omega \subset \bigcup_{\mathbf{j}} \Omega_{\mathbf{j}}^{(\ell)}. \tag{17}$$

B.2.4 Mathematical Details of Window Functions

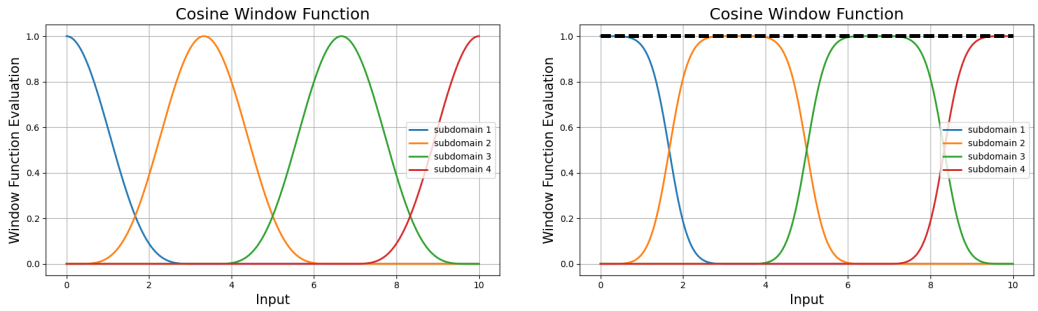
The purpose of the window function is to locally confine each neural network solution to its subdomain. Any differentiable function can be used as a window function, as long as it is (negligibly close to) zero outside of the subdomain and greater than zero within it. In this study, cosine based window function is preferred to be used as a window function. Owing to its infinite differentiability, non-vanishing interior behavior, and rapid exponential decay near the boundaries, this window function is well-suited for use in overlapping domain decomposition methods where smooth partition-of-unity properties are required, as in domain decomposed SWIM-PDE models.

The subdomain window functions which form a partition of unity for each level are defined as

$$\omega_j^{(\ell)}(x) = \frac{\hat{\omega}_j^{(\ell)}(x)}{\sum_{j=1}^{J} \hat{\omega}_j^{(\ell)}(x)}, \qquad \hat{\omega}_j^{(\ell)}(x) = \prod_{d=1}^{n} \left(\frac{1 + \cos\left(\pi \frac{x_d - \mu_d}{\sigma_d}\right)}{2} \right)^2, \quad \text{for } \ell \geq 1, \tag{18}$$

where the window functions are compactly supported and satisfy $\hat{\omega}_j^{(\ell)}(x) = 0$ whenever $|(x_d - \mu_d)/\sigma_d| > 1$ for any dimension d. Here, $\mu_d = (a_d + b_d)/2$ and $\sigma_d = (b_d - a_d)/2$ denote the center and half-width of each subdomain along dimension d, respectively.

Window function evaluations over a partitioned 1D domain are demonstrated in Figure 3. It can be seen that each window function evaluates only the collocation points within their corresponding subdomains as nonzero, and their normalized forms obey the partition of unity as described before, when they are summed up over the entire domain.



(a) (b)

Figure 3: Evaluation of window functions for the domain $\Omega \in [0, 10]$ with a 1.9 overlap ratio and 4 subdomains. (a) Unnormalized Gaussian window function evaluations. (b) Normalized Gaussian window function evaluations obtained considering the partition of unity using Equation (18).

B.2.5 Mathematical Details of Constraining Operator

As an option, we introduce the possibility of defining a constraining operator that enforces boundary conditions in a hard manner, rather than determining coefficients that only softly satisfy the boundary conditions in the resulting least-squares problem, as is commonly done in FBPINN frameworks Moseley et al. (2023); Dolean et al. (2024). Although defining an appropriate constraining operator may not be possible for all problems, all problems considered in this study are defined on a two-dimensional domain $\Omega = [0, 1] \times [0, 1]$ and are subject to homogeneous Dirichlet boundary conditions. Therefore, we employ the constraining operator

$$[\mathcal{C}u](\mathbf{x}, \theta) = \tanh\left(\frac{x_1}{\sigma}\right) \tanh\left(\frac{1 - x_1}{\sigma}\right) \tanh\left(\frac{x_2}{\sigma}\right) \tanh\left(\frac{1 - x_2}{\sigma}\right) u(\mathbf{x}, \theta) \tag{19}$$

that automatically satisfies boundary conditions at the domain boundaries for all the examples we present in this study.

C Additional Experimental Details

We first compare standard, one-level, and multilevel Frozen-PINN models on a homogeneous Laplacian problem to illustrate how domain decomposition and hierarchical multilevel constructions improve approximation accuracy. We then analyze the influence of key parametric choices, such as the overlap ratio and the number of basis functions, on prediction accuracy on the same problem.

Next, we consider multiscale Laplacian problem and compare the proposed multilevel Frozen-PINN-swim and Frozen-PINN-elm models against multilevel FBPINN models in terms of accuracy and runtime.

Finally, we study how domain decomposition and multilevel strategies affect the conditioning of the random-feature matrices arising in the associated least-squares problems by evaluating different SWIM model variants with equal total network capacity on a fixed multiscale Laplacian configuration.

C.1 Homogeneous Laplacian Experiment

This experiment is adapted from Dolean et al. (2024) to test and compare the approximation performance of one-level and multilevel Frozen-PINN models.

The homogeneous Laplacian problem is formulated as

$$\begin{aligned} -\Delta u = f, \quad \text{in } \Omega = [0, 1]^2, \quad u = 0, \quad \text{on } \partial\Omega, \\ f(x_1, x_2) = 32\left(x_1(1 - x_1) + x_2(1 - x_2)\right), \end{aligned} \tag{20}$$

where the exact solution can be obtained by

$$u(x_1, x_2) = 16\, x_1(1 - x_1)\, x_2(1 - x_2). \tag{21}$$

For the given homogeneous PDE, we conduct three separate experiments to investigate how the approximation accuracy is affected by (i) the number of levels and subdomains, (ii) the overlap ratio, and (iii) the capacity of local subdomain networks. These experiments are carried out using one-level and multilevel Frozen-PINN-swim models with varying configurations.

In each experiment, new models are derived from a baseline configuration, where all parameters are kept consistent with the baseline model (as defined in Table 3) except for the specific parameter being studied. The number of partitions at each level is determined using the formulation $2^{d(l-1)}$, where l is the level index and d is the number of dimensions of the problem. As the baseline, we select a Frozen-PINN-swim-MLDD model with $L = 3$ levels, 16 tanh basis functions per subdomain network, and an overlap ratio of 1.9.

Table 3: Frozen-PINN-swim-MLDD base model parameters

Parameters	**Value**
Number of levels	3
Partition per level per dimension	[1, 2, 4]
Number of basis functions	16
Overlapping ratio	1.9

For the level and partitioning experiment, the number of levels L in the Frozen-PINN-swim-MLDD models is increased from $L = 2$ to $L = 5$, following the partitioning configuration defined in Table 7. For each multilevel model configuration, a corresponding one-level Frozen-PINN-swim-DD model is also constructed, using the same number of total subdomains as the final level of the multilevel model. This comparison is performed to evaluate whether multilevel domain decomposition offers any advantages over the one-level decomposition approach in terms of approximation accuracy.

Table 4: Leveling and partitioning configurations for one-level and multi-level models

Configuration	**Frozen-PINN-swim-DD**	**Frozen-PINN-swim-MLDD**
1	[2]	[1, 2]
2	[4]	[1, 2, 4]
3	[8]	[1, 2, 4, 8]
4	[16]	[1, 2, 4, 8, 16]

For the second experiment on the homogeneous Laplacian problem, the overlap ratio of the between neighboring subdomains is varied from 1.1 to 2.7 in increments of 0.4, . This experiment aims to investigate how increasing the overlap between adjacent subdomains at each level influences the approximation accuracy of the global solution.

In the third and final experiment, the number of basis functions used in each subdomain SWIM network is varied from 2 to 32. This setup is used to evaluate how the approximation accuracy scales with increasing local network capacity, while keeping all other parameters of the baseline model unchanged.

All models are trained using $80 \times 80 = 6400$ uniformly spaced collocation points and evaluated on $350 \times 350 = 122500$ test points distributed across the domain. As the accuracy metric, the normalized mean absolute error, defined in Equation 25, is used to assess model performance.

The results of the conducted experiments on the homogeneous Laplacian problem are presented in Figure 4. From the level and partitioning experiment shown in Figure 4 (b), it can be observed that multilevel models achieve higher accuracy compared to one-level models with the same number of partitions at their final level. This demonstrates that multilevel decomposition effectively captures features at different resolutions, thereby improving global predictions. Figure 4 (c) shows that accuracy improves as the overlap ratio increases, likely due to enhanced communication between neighboring subdomain networks through bigger overlapping regions. Finally, as depicted in Figure

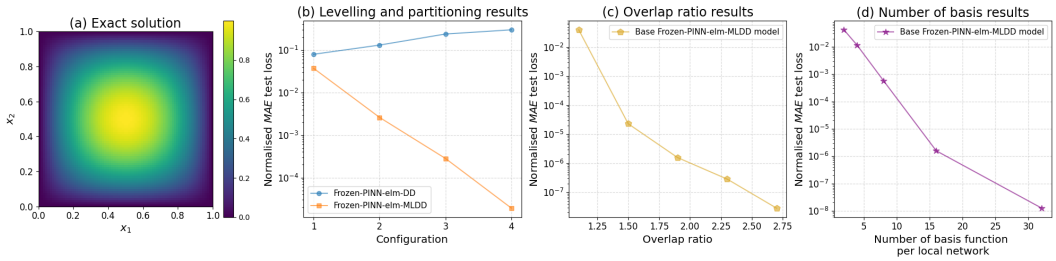


Figure 4: Results of the homogeneous Laplacian experiments. (a) Exact solution of the homogeneous Laplacian problem. (b) Results of the level and partitioning experiment using the configurations defined in Table 7. (c) Results of the overlapping ratio experiment using the baseline Frozen-PINN-swim-MLDD model. (d) Results of the basis function experiment using the baseline Frozen-PINN-swim-MLDD model.

4 (d), the approximation error exponentially decreases with the number of basis functions used in each subdomain network, as expected, due to the increased model representation capacity.

C.2 Multiscale Laplacian Experiment

This example is adapted from Dolean et al. (2024) to evaluate how the performance of multilevel Frozen-PINN models scales as additional multiscale components are introduced into the solution of a PDE, and also to compare these model with the multilevel FBPINN models on multiscale setup. The PDE examined in this experiment is a multiscale variant of the Laplacian problem discussed in the previous section (Equation 20). The corresponding forcing function for this PDE is defined as

$$f(x_1, x_2) = \frac{2}{n} \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x_1) \sin(\omega_i \pi x_2), \tag{22}$$

where the corresponding exact solution can be obtained by

$$u(x_1, x_2) = \frac{1}{n} \sum_{i=1}^{n} \sin(\omega_i \pi x_1) \sin(\omega_i \pi x_2). \tag{23}$$

In the given problem, the solution contains multiple frequency components, where the parameters n and ω_i control the number and frequency of these components, respectively. The normalized mean absolute error given in Equation 25 is used as the evaluation metric to assess model performance in weak scaling experiments Dolean et al. (2024) presented in the following section.

C.2.1 Weak Scaling Experiment

In the weak scaling experiment, the problem complexity, number of collocation points, and model capacity are all increased at the same rate to assess how the prediction performance of the multilevel Frozen-PINN models scales with problem size. The model capacity is increased in the same manner as in the strong scaling experiment described in the previous section by increasing the number of levels from $L = 2$ to 4. The complexity of the underlying PDE is also scaled. Specifically, for each test case, the number of frequency components is set to $n = L - 1$, with the frequency of each component defined as $\omega_i = 2^i$, for $i = 1, ..., n$.

Although each configuration is trained using a grid of $(10 \times 2^{L-1}) \times (10 \times 2^{L-1})$ uniformly spaced collocation points, the test evaluation is performed on a grid of 350×350 uniformly-spaced points. Consequently, the number of test points, the number of subdomains, and the frequency content of the solution all grow exponentially with increasing L. The overlapping ratio 1.9 used for all models, and the number of basis functions for Frozen-PINN models adjusted where possible such that similar fixed normalized L_1 loss obtained for all model at each level to perform a fair comparison as suggested in McGreivy & Hakim (2024).

Table 5: Leveling and partitioning configurations for one-level and multi-level models

Configuration	n	L	Partitioning	Training points	Test points
1	1	2	[1, 2]	20 x 20	350 x 350
2	2	3	[1, 2, 4]	40 x 40	350 x 350
3	3	4	[1, 2, 4, 8]	80 x 80	350 x 350

Table 6: Multiscale Laplacian week scaling test results with utilized configurations

Model	Level partitioning	Number of basis	Training time (s)	Normalized training time	Normalized L_1 loss
FBPINN	[1, 2]	16	2.605e+02 ± 7.7e+00	100.28	4.008e-05 ± 3.0e-05
Fr-PINN-elm	[1, 2]	32	2.598e+00 ± 4.9e-01	1.00	7.436e-06 ± 2.9e-06
Fr-PINN-swim	[1, 2]	32	2.474e+00 ± 5.2e-01	0.95	4.479e-03 ± 7.2e-03
FBPINN	[1, 2, 4]	16	9.135e+02 ± 2.2e+01	79.17	1.199e-04 ± 1.0e-04
Fr-PINN-elm	[1, 2, 4]	32	1.154e+01 ± 2.5e+00	1.00	9.600e-05 ± 4.3e-05
Fr-PINN-swim	[1, 2, 4]	32	1.063e+01 ± 2.2e+00	0.92	1.547e-02 ± 1.2e-02
FBPINN	[1, 2, 4, 8]	16	6.321e+03 ± 1.4e+02	126.95	3.587e-04 ± 1.3e-04
Fr-PINN-elm	[1, 2, 4, 8]	32	4.979e+01 ± 9.8e+00	1.00	6.018e-04 ± 3.4e-04
Fr-PINN-swim	[1, 2, 4, 8]	32	4.747e+01 ± 8.2e+00	0.95	5.784e-01 ± 2.8e-01

The weak scaling experiment results are shown in Figure 2 as well as in Table 7, based on the configurations listed in Table 5.

C.2.2 Conditioning Experiment

The conditioning property of the random feature matrix $\mathbf{M}$ significantly affects the runtime of the least squares solver during training and the resulting approximation error of the RFMs. Better conditioning property implies faster runtime and lower approximation error Chen et al. (2024), and it is indicated from the slow decay of the singular values of the matrix $\mathbf{M}$. This experiment is designed to investigate how the condition property of the random feature matrix $\mathbf{M}$ changes for all types of introduced Frozen-PINN models.

For this experiment, the second configuration of the multi-scale Laplacian problem discussed in the weak scaling experiment section (see Table 5) is selected as PDE, which contains two multi-scale components in its solution. To create a comparable random feature matrix $\mathbf{M}$ of the same size, regular, one-level and multilevel Frozen-PINN-swim models are constructed such that they all have the same total representational capacity by tuning their number of basis functions per SWIM network regarding to the determined domain partition and leveling strategy for them.

Table 7: Conditioning number experiment settings

Models	Partition per dim	Basis per NN	Training points	M size
SWIM-PDE	-	336	350 x 350	(121104, 336)
SWIM-PDE-DD	[4]	21	350 x 350	(121104, 336)
SWIM-PDE-MLDD	[1, 2, 4]	16	350 x 350	(121104, 336)

The results of the test are shown in Figure 5 for the parametric configurations used, as defined in Table 7. The singular value spectra reveal a substantial improvement in the conditioning of the random feature matrix $\mathbf{M}$ when domain decomposition with a partition of unity is incorporated into the Frozen-PINN framework. While the regular model exhibits a rapid decay of singular values, indicating strong column correlations and a low effective numerical rank, the domain-decomposed variants display a markedly flatter spectrum with sustained singular values across a larger index range. This effect is slightly further amplified in the multilevel configuration, suggesting that distributing random features across both spatial subdomains and resolution levels promotes a more

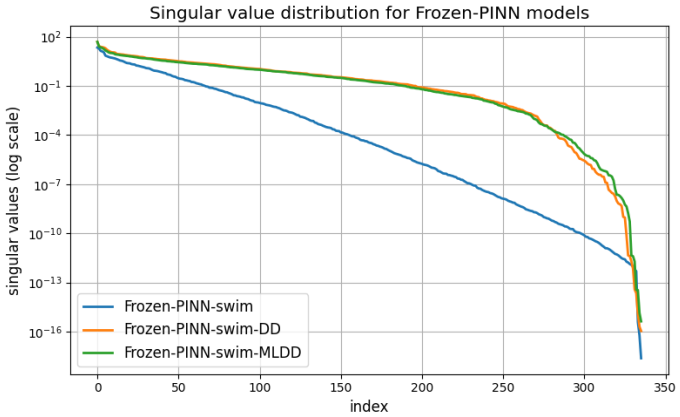


Figure 5: Conditioning experiment results for different Frozen-PINN models. The singular values of the random feature matrices $\mathbf{M}$ are plotted for regular (blue), one-level (orange), and multilevel (green) Frozen-PINN models in decaying order.

balanced, decorrelated feature representation. From a numerical standpoint, the improved spectral behavior implies a better-conditioned least-squares system, reduced sensitivity to perturbations, and more stable coefficient estimation. These results demonstrate that domain-decomposed SWIM-PDE models improve not only approximation accuracy and scalability but also numerical robustness and training efficiency, aligning with theoretical expectations for localized random feature methods Chen et al. (2024).

C.3 Pseudo Codes

C.3.1 Frozen-PINN

Algorithm 1 Frozen-PINN training

1: Randomly assign input weights $\mathbf{w}_c$ and biases b_c, $c = 1, \ldots, C$ according to SWIM network heuristics Bolager et al. (2023).
2: Calculate the hidden layer output matrix $\mathbf{M}$ as defined in Equation (6).
3: Calculate the output weight vector $\mathbf{a}$ by solving the least-squares problem in Equation (5).

C.3.2 Frozen-PINN-MLDD

Algorithm 2 Frozen-PINN-(ML)DD random feature matrix construction

Inputs: Decomposed domains $\{\mathcal{D}_\ell\}_{\ell=1}^{N_D}$, forcing function f, boundary function b, local ansatz template $\mathcal{A}$, PDE operator $\mathcal{N}$, BC operator $\mathcal{B}$, interior points within global domain $\mathbf{x}_{\text{int}}$, boundary points of global domain $\mathbf{x}_{\text{bdry}}$
Initialize: Local ansatzs list $\mathcal{A}_{\text{list}}$, local ansatz evalutions list $\mathcal{M}_{\text{list}}$
Evaluate interior points $\mathbf{b_{int}} \leftarrow f(\mathbf{x}_{\text{int}})$
Evaluate boundary points $\mathbf{b_{bd}} \leftarrow f(\mathbf{x}_{\text{bdry}})$
Stack domain evaluations $\mathbf{b} \leftarrow \begin{bmatrix} \mathbf{b}_{\text{int}} \\ \mathbf{b}_{\text{bd}} \end{bmatrix}$
for each decomposed domain $\mathcal{D}_\ell$ **do**
  for each subdomain $\{\mathcal{S}_j\}_{j=1}^{N_J} \subset \mathcal{D}_\ell$ **do**
    Clone ansatz template $\mathcal{A}_{\ell j} \leftarrow \texttt{Copy}(\mathcal{A})$
    Attach domains to local ansatz $\mathcal{A}_{\ell j}.\texttt{attach_domain}(\mathcal{S}_j)$, $\mathcal{A}_{\ell j}.\texttt{attach_domain}(\mathcal{D}_\ell)$
    Perform local sampling $(\mathcal{S}_{\text{sampled}}, \mathbf{b}_{\text{sampled}}) \leftarrow \texttt{SampleAndEval}(\mathcal{D}_\ell, f, \texttt{index} = j)$
    Perform SWIM initialization $\mathcal{A}_{\ell j}.\texttt{fit}(\mathcal{S}_{\text{sample}}, \mathbf{b}_{\text{sampled}})$) Bolager et al. (2023)
    Compute ansatz PDE evaluations $\mathbf{M}_{\text{eq}} \leftarrow \mathcal{N}(\mathcal{A}_{\ell j}, \mathcal{D}_{\text{int}})$
    Add equation bias $\mathbf{M}_{\text{int}} \leftarrow [\mathbf{M}_{\text{eq}} \mid \nVdash \cdot \mathcal{N}_{\text{bias}}]$
    Compute ansazt BC evalutions $\mathbf{M}_{\text{bnd}} \leftarrow \mathcal{B}(\mathcal{A}_{\ell j}, \mathcal{D})$
    Add BC bias $\mathbf{M}_{\text{bdry}} \leftarrow [\mathbf{M}_{\text{bnd}} \mid \nVdash \cdot \mathcal{B}_{\text{bias}}]$
    Stack ansatz evaluations $\mathbf{M}_{\ell j} \leftarrow \begin{bmatrix} \mathbf{M}_{\text{int}} \\ \mathbf{M}_{\text{bd}} \end{bmatrix}$
    Append local ansatz evaluations $\mathcal{M}_{\text{list}} \leftarrow \mathcal{M}_{\text{list}} \cup \{\mathbf{M}_{\ell j}\}$
    Detach subdomain from local ansatz $\mathcal{A}_{\ell j}.\texttt{detach_domain}(\texttt{type} = \text{"subdomain"})$
    Append initialized ansatz $\mathcal{A}_{\text{list}} \leftarrow \mathcal{A}_{\text{list}} \cup \{\mathcal{A}_{\ell j}\}$
  end for
end for
Construct global design matrix as defined in Eq. 6 $\mathbf{M} \leftarrow \texttt{Concat}(\mathcal{M}_{\text{list}}, \texttt{axis} = 1)$
Find all basis coefficients $\mathbf{a}$ defined in Eq. 5 $\mathbf{a} \leftarrow \texttt{SolveLeastSquares}(\mathbf{M}, \mathbf{b})$
Create fully trained local SWIM networks SWIMs $\leftarrow \texttt{CreateLocalSWIMs}(\mathcal{A}_{\text{list}}, \mathbf{a})$
Create final trained model Frozen-PINN-MLDD $\leftarrow \texttt{Create}(\text{SWIMs})$
Outputs: Frozen-PINN-MLDD, $\mathbf{M}$

C.4 Code and Hardware

Code for our paper is publicly available under the branch name called *workshop* at:

`https://gitlab.lrz.de/erayyildiz/swim-pde-thesis.git`.

All experiments were conducted on a single-socket machine equipped with an 11th-generation Intel® Core™ i7-11800H CPU (8 physical cores, 16 threads) running Ubuntu Linux. Note that introduced Frozen-PINN–based methods do not include explicit parallel implementations for execution across multiple CPUs.

C.5 Loss Functions

C.5.1 Mean Absolute Error Loss

$$\text{MAE} = \frac{1}{N_X} \sum_{i=1}^{N_X} \left| u_{\text{exact}}(x^i_{\text{test}}) - u_{NN}(x^i_{\text{test}}) \right| \tag{24}$$

C.5.2 Normalized Mean Absolute Error Loss

$$\text{Normalized MAE} = \frac{1}{N_X} \sum_{i=1}^{N_X} \frac{\left| u_{\text{exact}}(x^i_{\text{test}}) - u_{NN}(x^i_{\text{test}}) \right|}{\sigma} \tag{25}$$