
Pretraining Basis Functions Accelerates Convergence and Improves Accuracy in Domain-Decomposed Physics-Informed Neural Networks

Samuel Anderson

Department of Mathematics and Statistics
University of Strathclyde
Glasgow, G1 1XH, UK

Philéas Grenade

Department of Earth Science and Engineering
Imperial College London
London, SW7 2AZ, UK

Sean De Marco

Department of Earth Science and Engineering
Imperial College London
London, SW7 2AZ, UK

Jennifer Pestana

Department of Mathematics and Statistics
University of Strathclyde
Glasgow, G1 1XH, UK

Victorita Dolean

Department of Mathematics and Computer Science
Eindhoven University of Technology
Eindhoven, 5600 MB, The Netherlands

Ben Moseley*

Department of Earth Science and Engineering
Imperial College London
London, SW7 2AZ, UK
b.moseley@imperial.ac.uk

Abstract

Physics-informed neural networks (PINNs) provide a flexible approach for solving partial differential equations (PDEs), but suffer from slow training and spectral bias. Domain-decomposed PINNs with random neural network bases can accelerate training by replacing nonlinear optimisation with linear least-squares solves, but their bases can be redundant and poorly suited to a given PDE class. We instead pretrain a shared subdomain basis on example solutions from a PDE class during an offline representation-learning stage, then freeze the basis and solve only for its coefficients at train-time on new problem instances. On 1D harmonic oscillator and 2D Helmholtz problems, pretrained bases significantly improve accuracy, conditioning, and solve efficiency over random bases. Furthermore, we show bases pretrained on low-frequency problems transfer zero-shot to higher-frequency instances by scaling the number of subdomains, without further pretraining.

1 Introduction

Solving partial differential equations (PDEs) is fundamental to computational science and engineering, and *physics-informed neural networks* (PINNs) [12, 15] have become a popular mesh-free approach.

*Corresponding author.

However, PINNs suffer from slow training and spectral bias towards low-frequency functions [14, 20, 19]. Domain decomposition strategies [10, 9, 11, 13] offer a promising way to mitigate these issues. For example, *finite basis PINNs* (FBPINNs) [13] use overlapping subdomains with small neural networks in each subdomain, reducing the complexity each network must represent and improving accuracy and efficiency [13, 5]. However, such approaches still require expensive nonlinear optimisation of the network parameters.

Recent work has instead replaced trainable subdomain networks with random neural network bases, fixing the hidden parameters and learning only the final-layer coefficients [3, 6–8, 21]. For linear PDEs, this converts training into a linear least-squares problem, enabling much faster solves. *Extreme learning machine (ELM)-FBPINNs* [2, 1], for example, achieve ~ 2 – 3 orders of magnitude reductions in solve time while maintaining similar accuracy. However, the resulting bases can be highly linearly dependent, leading to severely ill-conditioned systems. Recent preconditioning approaches [18, 16] improve the resulting solves, but do not address the quality of the underlying basis.

We instead *learn the representation*: for a given problem class, we first pretrain a shared subdomain basis to accurately represent example solutions sampled across a range of boundary/ initial conditions (BCs/ICs) in an offline stage. The basis is then frozen and reused at train-time when solving new problem instances, with only its coefficients determined through a linear least-squares solve. On 1D harmonic oscillator and 2D inhomogeneous Helmholtz problem classes, pretrained bases improve accuracy, conditioning, and solve efficiency over random bases. Furthermore, we find bases pretrained on low-frequency problems transfer *zero-shot* to higher-frequency instances by scaling the number of subdomains, without further pretraining. More broadly, our work provides a simple way to learn reusable representations that can accelerate solving families of related PDE problems.

Related Work Others have investigated the importance of representations for PINNs through, e.g., through one-shot transfer learning of learned bases [4] and other adaptive approaches [17]. We differ in specifically pretraining basis functions for domain-decomposed PINNs and demonstrating that the resulting representation transfers zero-shot to more complex problem instances.

2 Background: Random-Basis FBPINNs

First, we concisely define ELM-FBPINNs [2, 18, 1], which our work extends. We consider a linear PDE on a domain $\Omega \subset \mathbb{R}^d$, $\mathcal{N}[u(\mathbf{x})] = f(\mathbf{x})$, $\mathbf{x} \in \Omega$ with BCs/ICs $\mathcal{B}_b[u(\mathbf{x})] = g_b(\mathbf{x})$, $\mathbf{x} \in \Gamma_b \subset \partial\Omega$, $b = 1, \dots, B$, where $\mathcal{N}$ and the $\mathcal{B}_b$ are linear operators and f, g_b are forcing and boundary data. The ELM-FBPINN approximates u by

$$\hat{u}(\mathbf{x}, \mathbf{a}, \boldsymbol{\theta}) = \sum_{j=1}^J \omega_j(\mathbf{x}) \sum_{k=1}^K a_{jk} \phi_{jk}(\mathbf{x}, \boldsymbol{\theta}_{jk}), \quad (1)$$

where $\{\omega_j\}_{j=1}^J$ are smooth, compactly-supported window functions forming a partition of unity on an overlapping cover $\Omega = \bigcup_{j=1}^J \Omega_j$, each ϕ_{jk} is a scalar output of a small (typically fully-connected) neural network local to subdomain Ω_j with hidden weights $\boldsymbol{\theta}_{jk}$ where the input coordinates to the network are normalised over the subdomain range, and $\mathbf{a} = \{a_{jk}\}_{j,k}$ and $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{jk}\}_{j,k}$.

Crucially, the hidden weights of the subdomain networks are randomly initialised and frozen, and only the output coefficients $\mathbf{a}$ are solved for by minimizing the physics-informed loss

$$\mathcal{L}_{\text{train}}(\mathbf{a}) = \frac{1}{N_I} \sum_{n=1}^{N_I} (\mathcal{N}[\hat{u}(\mathbf{x}_n, \mathbf{a})] - f(\mathbf{x}_n))^2 + \sum_{b=1}^B \frac{\lambda_b}{N_B^{(b)}} \sum_{i=1}^{N_B^{(b)}} (\mathcal{B}_b[\hat{u}(\mathbf{x}_i^{(b)}, \mathbf{a})] - g_b(\mathbf{x}_i^{(b)}))^2, \quad (2)$$

with interior collocation points $\{\mathbf{x}_n\}_{n=1}^{N_I} \subset \Omega$, boundary collocation points $\{\mathbf{x}_i^{(b)}\}_{i=1}^{N_B^{(b)}} \subset \Gamma_b$, $N_B = \sum_{b=1}^B N_B^{(b)}$ and positive scalar weights $\lambda_b > 0$. Because $\mathcal{N}$ and the $\mathcal{B}_b$ are linear and only $\mathbf{a}$ is trainable, (2) is equivalent to a sparse linear least-squares problem $\min_{\mathbf{a} \in \mathbb{R}^{JK}} \|\mathbf{M}\mathbf{a} - \mathbf{h}\|_2^2$, where $\mathbf{M} \in \mathbb{R}^{(N_I + N_B) \times JK}$ stacks the PDE-residual and boundary evaluations row-wise, and $\mathbf{h}$ collects the corresponding right-hand-side evaluations; explicit componentwise expressions can be found in [18, Eqs. (3)–(6)]. This least-squares system is solved with a sparse solver; in this work we use LSQR. Here, we train ELM-FBPINNs, but chose to share the same set of random hidden weights $\boldsymbol{\theta} = \{\boldsymbol{\theta}_k\}_{k=1}^K$ across subdomains, and refer to this variant as random-basis FBPINNs for clarity.

Problem	Model	(ω_0, J)	(h, K)	$\kappa(\mathbf{M}) \downarrow$	Optimiser	$e_{L_1} \downarrow$	Iterations $\downarrow$	Time (s) $\downarrow$
1D Oscillator	PINN	(80, 1)	(2, 64)	N/A	Adam	$6.3 \pm 0.2\text{e-}01$	10000	$2.7 \pm 0.0\text{e+}01$
1D Oscillator	FBPINN	(80, 16)	(3, 8)	N/A	Adam	$3.6 \pm 9.0\text{e-}02$	9870	$2.0 \pm 0.1\text{e+}01$
1D Oscillator	Random-basis FBPINN	(80, 16)	(3, 8)	$1.8\text{e+}12$	LSQR	$2.9 \pm 0.5\text{e-}03$	9860	$5.0 \pm 0.2\text{e-}01$
1D Oscillator	Pretrained-basis FBPINN	(80, 16)	(3, 8)	$3.1\text{e+}05$	LSQR	$3.5 \pm 0.8\text{e-}04$	3120	$1.7 \pm 0.5\text{e-}01$
2D Helmholtz	PINN	(8, 1)	(2, 64)	N/A	Adam	2.2 ± 1.4	9500	$2.0 \pm 0.3\text{e+}03$
2D Helmholtz	FBPINN	$(8, 16 \times 16)$	(3, 8)	N/A	Adam	$1.3 \pm 1.5\text{e-}01$	10000	$2.3 \pm 0.1\text{e+}03$
2D Helmholtz	Random-basis FBPINN	$(8, 16 \times 16)$	(3, 8)	$1.6\text{e+}06$	LSQR	$5.2 \pm 5.0\text{e-}02$	10000	$3.7 \pm 0.0\text{e+}01$
2D Helmholtz	Pretrained-basis FBPINN	$(8, 16 \times 16)$	(3, 8)	$1.3\text{e+}03$	LSQR	$9.7 \pm 1.3\text{e-}03$	1690	6.0 ± 0.2

Table 1: Performance of the pretrained-basis FBPINN at train-time compared to random-basis FBPINNs, FBPINNs, and PINNs on the 1D harmonic oscillator and 2D inhomogeneous Helmholtz problem classes. Error bars show the mean $\pm$ one standard deviation over 10 train runs with IC/BCs sampled from the same IC/BC distribution used during pretraining, but with a different random seed.

3 Method: Pretrained-Basis FBPINNs

Instead of randomly initialised basis functions, we *pretrain a shared subdomain basis for a given problem class* offline, with the aim of learning a representation that generalises across BCs/ICs. At train-time, given a particular BC/IC from the problem class, the pretrained basis is frozen and only its coefficients are determined by the least-squares solve described in Section 2. We learn a shared basis $\{\phi_k(\boldsymbol{\theta}_k)\}_{k=1}^K$, with parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_k\}_{k=1}^K$ shared across subdomains, which also enables transfer to larger problems with more subdomains without further pretraining. We pretrain using the loss

$$\mathcal{L}_{\text{pretrain}}(\mathbf{a}_{1:N_S}, \boldsymbol{\theta}) = \frac{1}{N_S} \sum_{s=1}^{N_S} \left[(1 - i_f) \cdot \frac{\alpha}{N_I} \sum_{n=1}^{N_I} (\hat{u}_s(\mathbf{x}_n, \mathbf{a}_s, \boldsymbol{\theta}) - u_s(\mathbf{x}_i))^2 + i_f \cdot \mathcal{L}_{\text{train}}(\mathbf{a}_s, \boldsymbol{\theta}) \right]. \quad (3)$$

The first term trains the basis to represent example solutions $\{u_s\}_{s=1}^{N_S}$ sampled from a predefined distribution of BCs/ICs, while the second regularises the basis by enforcing the governing equation. We linearly transition between the two terms during pretraining, with α balancing their scales and i_f denoting the fraction of completed optimisation steps, finding this helps pretrained-basis FBPINNs converge at train-time with similar accuracy observed during pretraining. For each example solution, coefficients $\mathbf{a}_{1:N_S} = \{\mathbf{a}_1, \dots, \mathbf{a}_{N_S}\}$ for each solution example are jointly optimised with the basis to enable each model solution to be evaluated; these coefficients are discarded after pretraining.

4 Results

Benchmarking and Metrics After pretraining, at train-time, we compare the performance of pretrained-basis FBPINN with random-basis FBPINN, and standard FBPINN and PINN in which all network parameters are optimised using gradient descent. Across all problems, all models use the same problem, model, and training hyperparameters as the pretrained-basis FBPINN, except that we use a larger global network for PINN for a fair comparison, and that FBPINN and PINN are trained with Adam, while pretrained-basis and random-basis FBPINN are trained with LSQR. All optimisers use early stopping, terminating when the train loss has not reduced by more than 1% over the preceding 1,000 iterations, with 10,000 max iterations. We evaluate train-time solve performance using the normalised L1 test error, given by $e_{L_1} = \frac{1}{N_T \varsigma} \sum_{i=1}^{N_T} |\hat{u}(\mathbf{x}_i) - u(\mathbf{x}_i)|$ where N_T is the number of test points regularly sampled over the problem domain, and ς is the standard deviation of the set $\{u(\mathbf{x}_i)\}_{i=1}^{N_T}$. All models are trained on a single AMD EPYC 7742 CPU core.

Benefit of Pretrained Basis Functions First, we pretrain basis functions on two problem classes: a damped 1D harmonic oscillator and a 2D inhomogeneous Helmholtz equation; full problem definitions are given in Section A. For the 1D harmonic oscillator, we construct $N_S = 10$ example solutions $\{u_s\}_{s=1}^{N_S}$ by uniformly sampling the amplitude and phase of the oscillator while keeping its frequency fixed at $\omega_0 = 80$. We use $K = 8$ basis functions, each a fully connected network with $h = 2$ hidden layers with 8 hidden units and tanh activations, within an FBPINN with $J = 16$ regularly spaced subdomains and cosine window functions. We use $N_I = 288$ regularly spaced interior collocation points and optimise the pretraining loss with Adam. For the 2D Helmholtz problem, we similarly generate $N_S = 10$ examples by sampling the solution frequency parameter

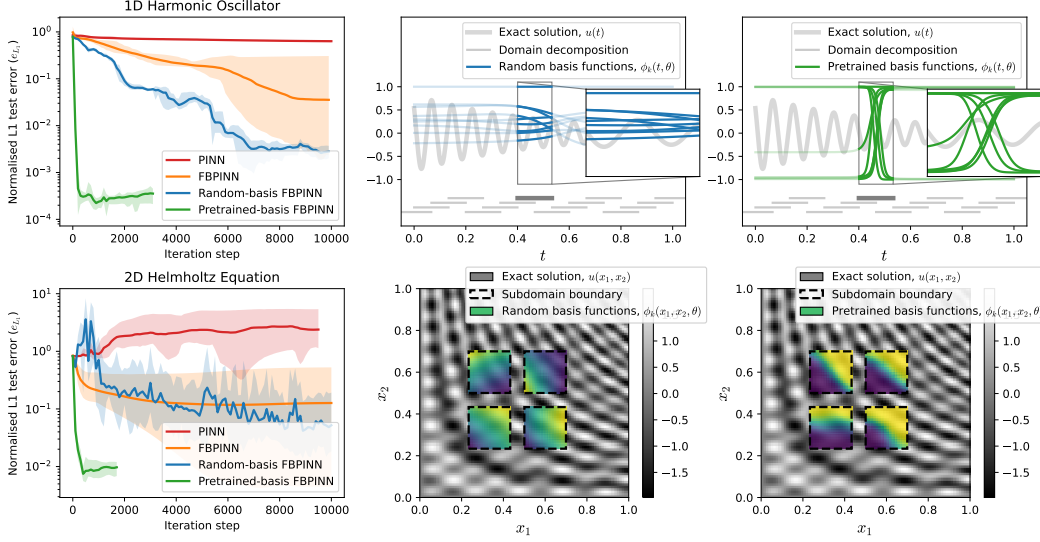


Figure 1: First column: Convergence of the pretrained-basis FBPINN at train-time on the 1D harmonic oscillator and 2D inhomogeneous Helmholtz problem classes, compared with other benchmarks. Error bars show the min–max range over 10 train runs with IC/BCs sampled from the same IC/BC distribution used during pretraining, but with a different random seed. Second and third columns: Comparison of pretrained and randomly initialised basis functions for each problem class, showing one exact solution from the train set. Only 4 of 8 basis functions are displayed for the 2D inhomogeneous Helmholtz problem for clarity.

from a normal distribution centred at $\omega = 8$, using the same basis architecture, $J = 16 \times 16$ regularly spaced rectangular subdomains, and $N_I = 68 \times 68$ regularly spaced interior points. Figure 1 shows the resulting pretrained bases. Compared with random initialisation, the pretrained bases better capture solution frequency and other problem-specific structure, such as the different orientations of Helmholtz standing waves, while the random bases appear highly correlated.

We then insert the pretrained bases into FBPINNs with the same number of subdomains and train them to solve 10 new problem instances sampled from the same IC/BC distributions but with a different random seed. The convergence and performance of the train runs are shown in Figure 1 and Table 1. For both problem classes, pretrained-basis FBPINNs significantly outperform all baselines, reducing least-squares conditioning, test error, iteration count, and training time. Both pretrained- and random-basis FBPINNs are substantially faster than FBPINNs and PINNs due to their linear least-squares solves, while PINNs fail to accurately converge on both problems.

Zero-Shot Transfer Learning To Larger Problems Next, we test whether pretrained bases transfer zero-shot to more complex problem instances without further pretraining. For both the 1D harmonic oscillator and 2D inhomogeneous Helmholtz equation problem classes, we reuse the pretrained bases and train FBPINNs with larger numbers of subdomains and higher-frequency solutions, increasing the solution frequency and scaling J proportionally. At each frequency, we sample 10 problem instances with ICs/BCs drawn from the same distributions used during pretraining, but at the increased frequency. Resulting test errors, solve times, and convergence curves are summarised in Figures 2 and 3. We find the pretrained bases consistently outperform random bases, converging to significantly lower errors with an order of magnitude fewer iterations.

5 Practical Considerations, Limitations and Future Work

The zero-shot transferability test demonstrates that the learned representation transfers robustly to increasingly complex problem instances, without requiring further pretraining. This is practically important because the cost of pretraining is not negligible: generating the pretraining reference solutions and performing pretraining takes 1.5 min for the 1D harmonic oscillator problem class and 181 min for the 2D inhomogeneous Helmholtz problem class. Thus, for our method to be practically

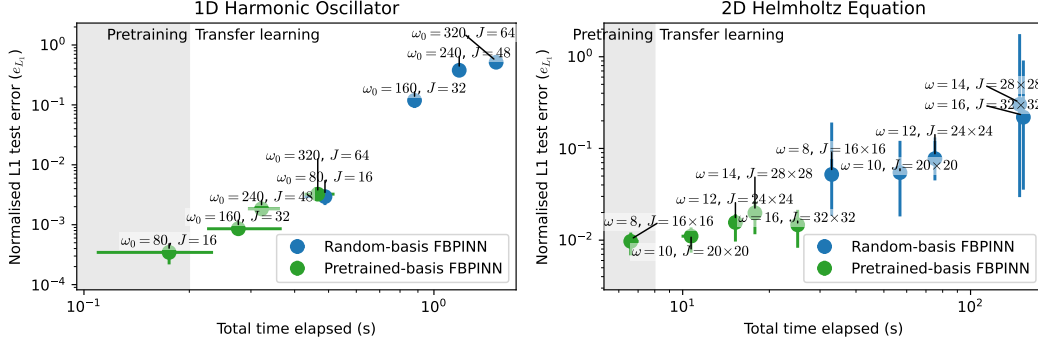


Figure 2: Transferability of pretrained bases to higher-frequency solutions. We take bases pretrained on the low-frequency $\omega_0 = 80$ 1D harmonic oscillator and $\omega = 8$ 2D inhomogeneous Helmholtz problems, and test their zero-shot transferability to higher-frequency solutions without further pre-training. The pretrained bases are inserted into FBPINNs with the number of subdomains, J , scaled proportionally with the exact solution frequency, and their performance is compared with random bases. Error bars show the min–max range over 10 train runs with IC/BCs sampled from the same IC/BC distribution used during pretraining, but at the increased frequency. Corresponding convergence curves are shown in Figure 3.

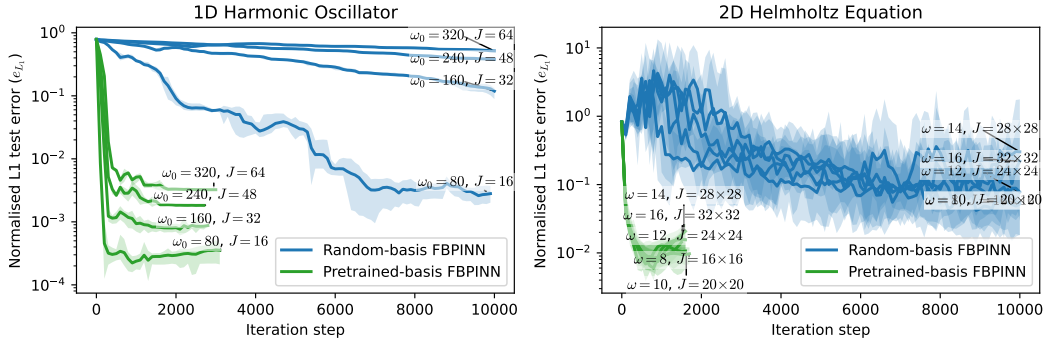


Figure 3: Convergence curves of the pretrained-basis and random-basis FBPINNs shown in Figure 2. Error bars show the min–max range over 10 train runs with IC/BCs sampled from the same IC/BC distribution used during pretraining, but with a different random seed.

useful, the cost of pretraining must be outweighed by the computational speedup provided by the pretrained basis at train-time. This is likely if the pretrained basis can be transferred zero-shot to larger problems and/or if many similar problems must be solved at train-time (as is often the case, for example, when solving inverse problems). The pretraining cost could potentially be reduced by skipping the generation of reference solutions, which could be computationally expensive, and instead pretraining using only the physics-informed loss (the second term in Equation (3)), which we leave for future work. It is also important to note that zero-shot transfer is likely only when the subdomain size is scaled in proportion to the solution frequency, as done here. Under normalized local subdomain coordinates, each subdomain then sees a local solution of similar complexity to those encountered during pretraining, effectively acting as a scale-matched domain refinement. Furthermore, we have not yet tested whether a basis pretrained on one BC/IC distribution can generalise outside this distribution, except in this scale-matched refinement setting.

A further limitation of this work is that we have only tested pretraining on two problem classes. Future work will extend these experiments to a wider range of problems, including 3D PDEs, and test the robustness of the pretrained bases by assessing their sensitivity to the number of basis functions and subdomain size. Finally, we note our learned basis effectively represents a PDE solution space and could potentially be applied to related tasks such as operator learning and inverse problems, which we leave for future work.

References

- [1] S. Anderson, V. Dolean, B. Moseley, and J. Pestana. ELM-FBPINNs: an efficient multilevel random feature method. *Machine Learning for Computational Science and Engineering*, 2(1): 23, 2026.
- [2] S. Anderson, V. Dolean, B. Moseley, and J. Pestana. ELM-FBPINN: efficient finite-basis physics-informed neural networks. In P. Bjørstad, X.-C. Cai, V. Dolean, D. E. Keyes, R. Kornhuber, and J. Xu, editors, *Domain Decomposition Methods in Science and Engineering XXVIII*, volume 155 of *Lecture Notes in Computational Science and Engineering*. Springer Nature Switzerland AG, 2026. doi: 10.1007/978-3-032-26946-1.
- [3] J. Chen, X. Chi, W. E, and Z. Yang. Bridging Traditional and Machine Learning-Based Algorithms for Solving PDEs: The Random Feature Method. *Journal of Machine Learning*, 1(3):268–298, 2022. ISSN 2790-2048. doi: 10.4208/jml.220726. URL <https://global-sci.com/index.php/jml/article/view/13180>.
- [4] S. Desai, M. Mattheakis, H. Joy, P. Protopapas, and S. Roberts. One-Shot Transfer Learning of Physics-Informed Neural Networks. In *2nd AI4Science Workshop at the 39th International Conference on Machine Learning (ICML)*, oct 2022. URL <https://arxiv.org/pdf/2110.11286>.
- [5] V. Dolean, A. Heinlein, S. Mishra, and B. Moseley. Multilevel domain decomposition-based architectures for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 429:117116, Sept. 2024. ISSN 0045-7825. doi: 10.1016/j.cma.2024.117116. URL <https://www.sciencedirect.com/science/article/pii/S0045782524003724>.
- [6] S. Dong and Z. Li. Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 387:114129, Dec. 2021. ISSN 0045-7825. doi: 10.1016/j.cma.2021.114129. URL <https://www.sciencedirect.com/science/article/pii/S0045782521004606>.
- [7] V. Dwivedi and B. Srinivasan. Physics Informed Extreme Learning Machine (PIELM)—A rapid method for the numerical solution of partial differential equations. *Neurocomputing*, 391:96–118, May 2020. ISSN 0925-2312. doi: 10.1016/j.neucom.2019.12.099. URL <https://www.sciencedirect.com/science/article/pii/S0925231219318144>.
- [8] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1):489–501, Dec. 2006. ISSN 0925-2312. doi: 10.1016/j.neucom.2005.12.126. URL <https://www.sciencedirect.com/science/article/pii/S0925231206000385>.
- [9] A. D. Jagtap and G. E. Karniadakis. Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020. ISSN 1991-7120. doi: 10.4208/cicp.OA-2020-0164. URL <https://global-sci.com/index.php/cicp/article/view/6911>.
- [10] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, June 2020. ISSN 0045-7825. doi: 10.1016/j.cma.2020.113028. URL <https://www.sciencedirect.com/science/article/pii/S0045782520302127>.
- [11] E. Kharazmi, Z. Zhang, and G. E. M. Karniadakis. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, Feb. 2021. ISSN 0045-7825. doi: 10.1016/j.cma.2020.113547. URL <https://www.sciencedirect.com/science/article/pii/S0045782520307325>.
- [12] I. Lagaris, A. Likas, and D. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, Sept. 1998. ISSN 10459227. doi: 10.1109/72.712178. URL <http://ieeexplore.ieee.org/document/712178/>.

- [13] B. Moseley, A. Markham, and T. Nissen-Meyer. Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations. *Advances in Computational Mathematics*, 49(4):62, July 2023. ISSN 1572-9044. doi: 10.1007/s10444-023-10065-9. URL <https://doi.org/10.1007/s10444-023-10065-9>.
- [14] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the Spectral Bias of Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5301–5310. PMLR, May 2019. URL <https://proceedings.mlr.press/v97/rahaman19a.html>.
- [15] 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, Feb. 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045. URL <https://www.sciencedirect.com/science/article/pii/S0021999118307125>.
- [16] Y. Shang, A. Heinlein, S. Mishra, and F. Wang. Overlapping Schwarz preconditioners for randomized neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 442(1), July 2025. doi: 10.1016/j.cma.2025.118011.
- [17] E. Torres, J. Schiefer, and M. Niepert. Adaptive Physics-informed Neural Networks: A Survey. *Transactions on Machine Learning Research*, 2025-March, mar 2025. ISSN 28358856. URL <https://arxiv.org/pdf/2503.18181>.
- [18] J. W. van Beek, V. Dolean, and B. Moseley. Local feature filtering for scalable and well-conditioned domain-decomposed random feature methods. *Computer Methods in Applied Mechanics and Engineering*, 449(118583), Feb. 2026. ISSN 0045-7825. doi: 10.1016/j.cma.2025.118583. URL <https://www.scopus.com/pages/publications/105022795998>.
- [19] S. Wang, Y. Teng, and P. Perdikaris. Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, Jan. 2021. ISSN 1064-8275. doi: 10.1137/20M1318043. URL <https://epubs.siam.org/doi/10.1137/20M1318043>.
- [20] S. Wang, H. Wang, and P. 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, Oct. 2021. ISSN 0045-7825. doi: 10.1016/j.cma.2021.113938. URL <https://www.sciencedirect.com/science/article/pii/S0045782521002759>.
- [21] E. Yıldız, C. Datar, A. Rahma, V. Dolean, and F. Dietrich. Fast multiscale PDE solvers via multilevel domain decomposition and random features. In *ICLR 2026 Workshop on AI for PDEs*, 2026. URL <https://openreview.net/forum?id=wA6SsMmftX>.

A Mathematical Problem Definitions

In this section, we mathematically define the problems studied, their training loss functions, and the solution sampling procedure used during pretraining. The problem definitions closely follow those used in [1].

A.1 1D Harmonic Oscillator

Problem Definition

We consider the displacement $u(t)$ of a damped harmonic oscillator governed by

$$\begin{cases} mu_{tt} + \mu u_t + ku(t) = 0, & t \in (0, 1], \\ u(0) = u_0, & u_t(0) = v_0, \end{cases} \quad (4)$$

where m is the mass, μ the damping coefficient, and k the spring constant. Defining

$$\beta = \frac{\mu}{2m}, \quad \omega_0 = \sqrt{\frac{k}{m}},$$

we model the solution in the under-damped regime, which corresponds to $\beta < \omega_0$, where the exact solution is given by

$$u(t) = e^{-\beta t} 2A \cos(\phi + \omega t),$$

where

$$\omega = \sqrt{\omega_0^2 - \beta^2},$$

and

$$A = \frac{u_0}{2 \cos(\phi)}, \quad \phi = \arctan\left(-\frac{(v_0/u_0) + \beta}{\omega}\right).$$

For all problems studied, we fix $m = 1$ and $\mu = 4$ (hence $\beta = 2$), and vary ω_0 to increase the complexity of the solution.

Training Loss Function

To solve this equation, we use the physics-informed loss function

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{phys}} + \lambda_{u_0} \mathcal{L}_{\text{ic}}^{(u_0)} + \lambda_{v_0} \mathcal{L}_{\text{ic}}^{(v_0)}, \quad (5)$$

with

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_I} \sum_{i=1}^{N_I} (m\hat{u}_{tt} + \mu\hat{u}_t + k\hat{u})^2(t_i), \quad t_i \in (0, 1], \quad (6)$$

$$\mathcal{L}_{\text{ic}}^{(u_0)} = (\hat{u}(0) - u_0)^2, \quad (7)$$

$$\mathcal{L}_{\text{ic}}^{(v_0)} = (\hat{u}_t(0) - v_0)^2, \quad (8)$$

where N_I denotes the number of interior collocation points and $\hat{u}$ is the model solution. We set $\lambda_{u_0} = k^2 = \omega_0^4$ and $\lambda_{v_0} = k = \omega_0^2$, which normalises all loss terms to the same physical unit and we find provides a reasonable balance between across different solution frequencies.

Pretraining Solution Sampling

During pretraining, we generate example solutions by randomly sampling

$$A_s \sim \mathcal{U}(0.5, 1.5), \quad \phi_s \sim \mathcal{U}\left(-\frac{\pi}{2}, \frac{\pi}{2}\right), \quad (9)$$

The corresponding initial conditions are then computed from the exact solution as

$$u_0^{(s)} = A_s \cos(\phi_s), \quad v_0^{(s)} = -A_s (\beta \cos(\phi_s) + \omega \sin(\phi_s)), \quad (10)$$

These initial conditions define the s -th example solution and are used when evaluating u_s and $\mathcal{L}_{\text{train}}(\mathbf{a}_s, \boldsymbol{\theta})$ in Equation (3). During pretraining we set $\alpha = \lambda_{u_0} = k^2$ which we find provides a reasonable balance between the terms.

A.2 2D Inhomogeneous Helmholtz Equation

Problem Definition

The problem is given by

$$\begin{cases} \Delta u + k^2 u = f, & \text{in } \Omega = (0, 1)^2, \\ u = g, & \text{on } \partial\Omega, \end{cases} \quad (11)$$

where $k > 0$ is the wavenumber and is fixed at $k = 1$. We define a manufactured solution

$$u(x_1, x_2) = \sin(\omega\pi x_1) \sin(2\omega\pi x_2) + \sin(3\omega\pi x_1 x_2), \quad (12)$$

from which the source term is derived as

$$\begin{aligned} f(x_1, x_2) &= \Delta u(x_1, x_2) + k^2 u(x_1, x_2) \\ &= -5(\omega\pi)^2 \sin(\omega\pi x_1) \sin(2\omega\pi x_2) - 9(\omega\pi)^2 (x_1^2 + x_2^2) \sin(3\omega\pi x_1 x_2) \\ &\quad + k^2 \left(\sin(\omega\pi x_1) \sin(2\omega\pi x_2) + \sin(3\omega\pi x_1 x_2) \right), \end{aligned} \quad (13)$$

and non-homogeneous Dirichlet boundary conditions are imposed as $g = u|_{\partial\Omega}$. We choose this manufactured solution to contain both high-frequency and non-separable spatial components, yielding a non-trivial solution with spatially varying boundary conditions. Increasing the frequency parameter ω increases the oscillatory complexity of both the solution and corresponding source term, thereby controlling the difficulty of the problem.

Training Loss Function

To solve this equation, we use the physics-informed loss function

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{phys}} + \lambda_{\text{bc}} \mathcal{L}_{\text{bc}}, \quad (14)$$

with

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_I} \sum_{i=1}^{N_I} \left(\Delta \hat{u} + k^2 \hat{u} - f \right)^2(\mathbf{x}_i), \quad \mathbf{x}_i \in \Omega, \quad (15)$$

$$\mathcal{L}_{\text{bc}} = \frac{1}{N_B^{(\text{bc})}} \sum_{i=1}^{N_B^{(\text{bc})}} \left(\hat{u}(\mathbf{x}_i) - g(\mathbf{x}_i) \right)^2, \quad \mathbf{x}_i \in \partial\Omega, \quad (16)$$

where N_I and $N_B^{(\text{bc})}$ denote the number of interior and boundary collocation points, respectively, and $\hat{u}$ is the model solution. We choose $\lambda_{\text{bc}} = 25\omega^4$, which normalises all loss terms to the same physical unit and we find provides a reasonable balance between across different solution frequencies.

Pretraining Solution Sampling

During pretraining, we generate example solutions by randomly sampling the frequency parameter from a normal distribution centred on a central frequency ω ,

$$\omega_s \sim \mathcal{N}(\mu = \omega, \sigma = \omega/10). \quad (17)$$

The corresponding boundary conditions are then computed from the exact solution as $g_s = u_s|_{\partial\Omega}$. These boundary conditions define the s -th example solution and are used when evaluating u_s and $\mathcal{L}_{\text{train}}(\mathbf{a}_s, \boldsymbol{\theta})$ in Equation (3). During pretraining we set $\alpha = \lambda_{\text{bc}} = 25\omega^4$ which we find provides a reasonable balance between the terms.