FB-HyDON: Parameter-Efficient Physics-Informed Operator Learning of Complex PDEs via Hypernetwork and Finite Basis Domain Decomposition

Milad Ramezankhani*
Applied Research, Quantiphi
Marlborough, MA 01752
milad.ramezankhani@quantiphi.com

Rishi Yash Parekh
Applied Research, Quantiphi
Marlborough, MA 01752
rishi.parekh@quantiphi.com

Anirudh Deodhar
Applied Research, Quantiphi
Marlborough, MA 01752
anirudh.deodhar@quantiphi.com

Dagnachew Birru
Applied Research, Quantiphi
Marlborough, MA 01752
dagnachew.birru@quantiphi.com

Abstract

Deep operator networks (DeepONet) and neural operators have gained significant attention for their ability to map infinite-dimensional function spaces and perform zero-shot super-resolution. However, these models often require large datasets for effective training. While physics-informed operators offer a data-agnostic learning approach, they introduce additional training complexities and convergence issues, especially in highly nonlinear systems. To overcome these challenges, we introduce Finite-Basis Physics-Informed HyperDeepONet (FB-HyDON), an advanced DeepONet-based operator architecture featuring intrinsic domain decomposition. By leveraging hypernetworks and finite-basis functions, FB-HyDON effectively mitigates the training limitations associated with existing physics-informed operators. We evaluated our approach on various benchmark problems including the high-frequency harmonic oscillator, Burgers' and Allen-Cahn equations. The results demonstrate substantial improvements over DeepONet and its variants, and exhibit comparable performance against the Fourier neural operator.

1 Introduction

Partial differential equations (PDEs) are integral in modeling and describing the dynamics of many complex systems in science and engineering applications. Numerical solvers such as finite element methods (FEMs) and finite difference methods (FDMs) often obtain the solution of PDEs by discretizing the domain and solving a finite-dimensional problem. However, obtaining high-resolution solutions to PDEs using numerical simulations for complex large-scale problems can be computationally expensive and prohibitive. There has been a growing interest in more efficient data-driven alternatives that can directly learn the underlying solutions from the available data without requiring explicit knowledge about the governing PDEs [1, 2]. More recently, operator learning has emerged as a promising paradigm, aiming to learn an unknown mathematical operator governing a system of PDEs [3]. They capture mappings between *infinite-dimensional* function spaces and have demonstrated potential in capturing complex solution behaviors [4, 5]. Furthermore, due to their inherent

*Corresponding author. Email address: milad.ramezankhani@quantiphi.com

Machine Learning and the Physical Sciences Workshop, NeurIPS 2024.

differentiability, they can be seamlessly applied to inverse problems, such as design optimization tasks [6]. Various architectures have been developed, including the Deep Neural Operator (DeepONet) [4], Fourier Neural Operator (FNO) [5], Graph Neural Operator [7], General Neural Operator Transformer (GNOT) [8] and Operator Transformer (OFormer) [9]. These models differ in their discretization methods and the approximation techniques they use to enhance efficiency and scalability.

Training the operators, however, relies on large datasets, which may not be readily available for many practical applications and can result in suboptimal generalization performance. One way to reduce (or eliminate) the operators' data dependency is to augment the training process with physical laws and learn the operator in a physics-informed fashion [10, 11, 12]. Despite their advantages, such physics-informed models often face challenges related to complex optimization landscapes and convergence difficulties, hindering their effectiveness [13]. In this paper, we introduce a novel operator-learning architecture designed to overcome the aforementioned limitations. We present Finite-Basis Physics-Informed HyperDeepONet (FB-HyDON), which efficiently and accurately learns the solution operator for complex, highly nonlinear, and high-frequency problems. The primary contributions of this study are as follows: 1) We propose FB-HyDON, an advanced DeepONet-based operator architecture with built-in domain decomposition functionality that effectively addresses the training limitations of existing physics-informed operators; 2) We introduce a hypernetwork-based variant of finite basis domain decomposition method [14] which facilitates learning complex PDE systems in a more parameter-efficient manner and maintains a constant number of trainable parameters for any domain decomposition, from coarse to fine; 3) We conduct a comparative analysis of FB-HyDON against various operator models, demonstrating that our model consistently achieves superior results while incorporating only a fraction of the model complexity needed in other operators.

2 Background

Operator learning. Considering a parametric PDE taking the form $\mathcal{N}(u, a) = 0$ where $u \in \mathcal{U}$ is the unknown solution, $a \in \mathcal{A} \subseteq \mathcal{V}$ is a PDE parameter and $\mathcal{N} : \mathcal{U} \times \mathcal{A} \to \mathcal{F}$ is a linear or nonlinear partial differential operator with $\mathcal{U}, \mathcal{V}, \mathcal{F}$ representing a triplet of Banach spaces. Given suitable initial and boundary conditions, for any $a \in \mathcal{A}$, we assume that there is a unique solution $u = u(a)$ to the problem. This formulation results in the solution operator $\mathcal{G} : \mathcal{A} \to \mathcal{U}$. The solution operator $\mathcal{G}$ can be approximated by an operator $\mathcal{G}_\theta$ with parameters θ [11]. Specifically in DeepONet's architecture, $\mathcal{G}_\theta$ comprises a branch net and a trunk net. The branch net encodes the input function information at discrete sensor points and the trunk net embeds the spatiotemporal coordinates of the PDE system. The networks' outputs are then merged through an element-wise multiplication followed by a summation to generate the PDE solution $\mathcal{G}_\theta(a)(y)$ [4] (more details in B.1).

HyperDeepONet. Since DeepONet approximates the target functions with a finite-dimensional linear subspace, it struggles to accurately learn complex functions with nonlinear and sharp features [15]. Various works have been proposed to address this limitation [15, 16, 12]. Recently, HyperDeepONet (HDON) [17] has been introduced as a more expressive variant of DeepONet capable of learning highly nonlinear operators with fewer network parameters. HDON replaces the branch net with a hypernetwork which infers the parameters of the trunk (target) net. Instead of propagating the input function information only through the last layer of the branch and trunk nets, HDON infuses this information into every parameter of the trunk net, enabling a more comprehensive integration of the input function across the entire network.

Physics-informed operator learning and domain decomposition. Physics-informed operator learning allows learning the operator using only the form of the PDE and its initial and boundary conditions, without the need for large datasets. By embedding the governing physical laws as constraints within the training process, operating learning methods can be trained in a fully data-agnostic [4], or hybrid manner [11]. However, similar to physics-informed neural networks (PINNs) [18], physics-informed operator learning is not without limitations. One major challenge is the convergence issues when dealing with complex systems, such as those with nonlinear time-varying characteristics and sharp transitions, which can severely impact the performance [13, 19, 20]. Domain decomposition, an effective way to tackle such challenges, involves breaking down the PDE domain into smaller, more manageable subproblems, and thus enhancing convergence. Two notable approaches in this context are extended PINN (XPINN) [21] and finite-basis PINN (FBPINN). FBPINN [14] is particularly interesting since, unlike XPINN, it does not introduce new loss terms, thereby maintaining a simpler optimization landscape. More background on FBPINNs is provided in B.2.

3 Methodology

3.1 FB-HyPINN

FBPINN partitions the problem domain into smaller, overlapping subdomains, each associated with a dedicated subnetwork that learns a basis function with compact support [14]. The overall solution is constructed by combining the basis functions (analogous to FEMs), where the prediction for each query point is achieved by the weighted sum of outputs from subnetworks whose corresponding subdomains encompass that point (subnetwork weights are determined by predefined window functions). However, this approach presents a significant drawback: the number of subnetworks (i.e., trainable parameters) increases with the number of subdomains, leading to potential inefficiencies in training. Rather than employing separate subnetworks for each subdomain, we propose a novel strategy utilizing a hypernetwork architecture, termed FB-HyPINN. In FB-HyPINN, we utilize a hypernetwork $\hat{h}(d, \theta_H)$ [22] that computes the parameters $\{\theta_j\}_{j=1}^J$ of the subnetworks $\{\hat{u}_j\}_{j=1}^J$ corresponding to J overlapping subdomains. The input d to the hypernetwork is a vector that uniquely defines a subdomain. In our approach we pick the subdomain's midpoint m_j and the distance s_j between the midpoint and subdomain's bounds as the input to the hypernetwork. Hence, the output of each subnetwork $\hat{u}_j(x_i^{norm}, \theta_j)$ is calculated via obtaining the parameters from the hypernetwork $\theta_j = \hat{h}((m_j, s_j), \theta_H)$ and feeding in the normalized input variables $x_i^{norm} = Norm(x_i, \Omega_j)$. Subsequently, the global solution $\hat{u}(x_i, \theta_H)$ for a collocation point x_i is obtained as

$$\hat{u}(x_i, \theta_H) = \sum_{j=1}^{J} \omega_j(x_i) \hat{u}_j(x_i^{norm}, \theta_j). \tag{1}$$

where ω_j denotes the window function. More details on the functionality and physics-informed training of FB-HyPINN are provided in B.2.

3.2 FB-HyDON

Dual-hypernetwork module. As depicted in Figure 1, the proposed FB-HyDON model has two hypernetworks with equal output dimensions followed by a target network which generates the output solution for any given query point y on the spatiotemporal domain. Operator hypernet h_O is responsible for mapping the input function observations $[a(x_1), a(x_2), \ldots, a(x_m)]$ at sensor points to a feature representation $[b_1, b_2, ..., b_q]^T$. Domain hypernet h_D on the other hand takes in the subdomain coordinates $[s, m]^T$ as the input and generates a feature embedding d. The hypernets' outputs are then merged via the Hadamard product, generating the weights of the target network. While one hypernet might be sufficient to take in and process both input functions and subdomain information, we observed that separate hypernets considerably improve the model's overall performance.

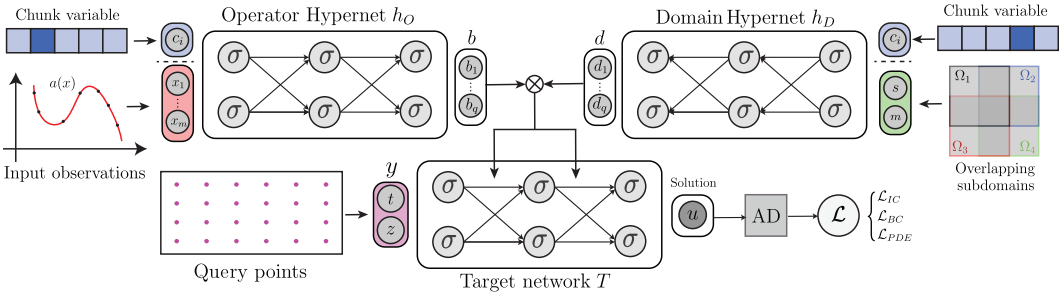

Figure 1: Schematic of proposed FB-HyDON model. The inputs to the hypernetworks consist of a chunk identifier and a task-specific set of variables (sensory observations for Operator hypernet and subdomain information for Domain hypernet.) The outputs of the hypernets are merged together via the Hadamard product to produce the weights of the Target net. Physics-informed losses are obtained at query points and used to train the hypernets' parameters.

Finite basis domain decomposition with hypernetwork. As described in Section 3.1, a hypernetwork (h_D in Figure 1) is utilized to generate the parameters for each subdomain's network, reducing

Table 1: Comparitive analysis on 1D Sinusoidal and 1D viscous Burgers equations for FBPINN and FB-HyPINN. Mean relative L_2 error (Rel. L_2) and mean absolute error (MAE) are reported for each method. Size refers to the total number of trainable parameters.

Model	Burgers			Sinusoidal		
	Size	Rel. L_2 ($\times 10^{-7}$)	MAE ($\times 10^{-2}$)	Size	Rel. L_2 ($\times 10^{-2}$)	MAE ($\times 10^{-2}$)
FBPINN [14]	8425	2.3	5.1	8988	1.07	5.4
FB-HyPINN (ours)	5221	0.36	1.02	6865	3.45	5.39

the overall parameter count while maintaining the model performance. Each subdomain is encoded into a unique vector $[m_j, s_j]$ that captures its spatial information (e.g., bounds coordinates). This vector serves as input to the hypernetwork, which generates the embedding d for the corresponding subnetwork. The embedding is then used to generate the parameters of the target network T which predicts the operator output.

Chunked hypernetwork. As the size of the target network increases, the output layer of the hypernetwork can get prohibitively large, potentially hindering both training efficiency and model performance. *Chunked hypernetworks* [23] offer an effective solution to this challenge. In this method, the parameters of the target network are generated in smaller, manageable chunks over multiple steps via iteratively invoking the hypernetwork. To differentiate between chunks, an additional input $\mathcal{C} = \{c\}_{i=1}^{N_c}$ is introduced to the hypernetwork which allows generating chunk-specific outputs for a given fixed task t. The full set of target network parameters can then be obtained by concatenating the outputs of the hypernetwork for each chunk: $\theta_t = [h(t, c_1), ..., h(t, c_{N_c})]$. In the proposed framework, both hypernetworks (h_O and h_D) leverage the chunking strategy (Figure 1).

4 Results

Physics-informed learning with finite basis domain decomposition. First, we evaluate the performance of the proposed hypernetwork architecture (FB-HyPINN) tasked to perform finite basis domain decomposition for physics-informed learning. In particular, we investigate two highly nonlinear case studies 1) high-frequency sinusoidal wave and 2) 1D viscous Burgers equation. We compare the performance of FB-HyPINN with vanilla FBPINN as the baseline model. The results are presented in Table 1. We used 28 subdomains and 25 subdomains for the sinusoidal and Burgers' equations respectively. We achieve comparable performance for the sinusoidal case with 23% less trainable parameters and outperform the baseline model in the Burgers' equation case. Figure A.4 demonstrates that in comparison to FBPINN, our model was able to more accurately predict the solution near the discontinuity. Details regarding the experiments and model architectures are provided in C and D.

Learning solution operators with parameter-efficient FB-HyDON. We evaluate the performance of our operator on three benchmark problems, namely, harmonic oscillator (Appendix A), 1D Burgers' equation and 1D Allen-Cahn equation, and compare it against four physics-informed baselines, DeepONet (DON) [4], modified DeepONet (MDON) [10], HyperDeepONet (HDON) [17] and FNO [11]. For all cases, the task is to learn the solution operator mapping the initial condition a to the solution u. For Burgers' and Allen-Cahn equations, we implement two variants of our model, one with a single hypernetwork (FB-HyDON-1) which takes in both input function and subdomain information as input, and one with a dual-hypernetwork module (FB-HyDON-2). Details of baseline models' architecture and training as well as the benchmark problems' setting are provided in C and D.

Table 2 and Figures A.2 and A.3 showcase the performance of the models for 1D Burgers' equation (high and low viscosities) and Allen-Cahn equation. Both versions of our models significantly outperform DeepONet and its variants, resulting in at least a 36% percent reduction in Relative L_2 error compared to the next best performer, HyDON. Our model also achieves remarkable efficiency in terms of network size. With only 95k parameters in the Burgers' high viscosity case, it matches the compact architecture of HyDON, while considerably outperforming the larger DON (215k parameters) and MDON (203k parameters). Additionally, we observe that utilizing the dual-hypernetwork module

Table 2: Comparitive analysis on 1D viscous Burgers' equations at high and low viscosity values and 1D Allen-Cahn equation. FB-HyDON-1 and FB-HyDON-2 are trained using 4 subdomains for $v = 0.01$, 16 subdomains for $v = 0.005$ and 8 subdomains for Allen-Cahn equation.

Model	**Burgers** ($\nu = 0.01$)		**Burgers** ($\nu = 0.005$)		**Allen-Cahn**	
	Size	Rel. L_2	Size	Rel. L_2	Size	Rel. L_2
DON [4]	215k	0.14	215k	0.23	278k	0.93
MDON [10]	203k	0.11	203k	0.19	291k	0.91
HyDON [17]	95k	0.089	128k	0.11	229k	0.87
FB-HyDON-1 (ours)	95k	0.057	128k	0.063	197k	0.24
FB-HyDON-2 (ours)	99k	**0.048**	130k	**0.051**	202k	**0.17**

Table 3: Comparitive analysis of FB-HyDON and FNO on 1D viscous Burgers' equations at various viscosities and 1D Allen-Cahn equation.

Model	**Burgers**						**Allen-Cahn**	
	$\nu = 0.01$		$\nu = 0.005$		$\nu = 0.001$		$\epsilon = 0.0001$	
	Size	Rel. L_2	Size	Rel. L_2	Size	Rel. L_2	Size	Rel. L_2
FNO-small [11]	113k	0.015	113k	0.027	165k	0.22	217k	0.27
FNO-large [11]	165k	0.017	165k	0.022	217k	0.14	270k	0.27
FB-HyDON-2	99k	0.048	130k	0.051	202k	0.18	202k	0.17

improved the model's generalization performance while only marginally increasing the network size. For lower viscosity (higher nonlinearities and sharper transitions), unlike DeepONet methods, our models are able to maintain their high performance thanks to the built-in domain decomposition feature. Similarly, FB-HyDON effectively utilizes domain decomposition to address the Allen-Cahn equation's nonlinearities and stiff terms, significantly outperforming DeepONet models.

We also compared the performance of FB-HyDON against FNO across different network sizes by varying the number of layers (details on the FNO architecture and training can be found in C). As shown in Table 3, the FNO models achieve higher accuracy on simpler and less complex problems, such as the high-viscosity Burgers' equation. However, for highly nonlinear systems like the low-viscosity Burgers' and Allen-Cahn equations, the proposed FB-HyDON model can match or even outperform FNO. This superior performance is attributed to FB-HyDON's built-in domain decomposition, which enhances its ability to effectively handle complex nonlinearities (Figure A.3). This is noteworthy, as it is generally observed that DeepONet-based methods tend to be outperformed by FNO models [5, 11]. Our findings demonstrate that FB-HyDON not only matches but can exceed FNO's performance in complex scenarios when the number of trainable parameters is similar, highlighting its efficiency and competitive edge.

5 Conclusion

This paper introduced FB-HyDON, a novel DeepONet-based operator architecture, that addresses key limitations in existing physics-informed operators. By incorporating a built-in domain decomposition feature via a hypernetwork, our model achieves improved parameter efficiency and enhanced performance, particularly for highly nonlinear systems. The model's performance was evaluated on various nonlinear benchmarks including the high-frequency harmonic oscillator and 1D Burgers' and Allen-Cahn equations. The results demonstrate substantial improvements over DeepONet-based operators as well as comparable performance against FNO. FB-HyDON's ability to increase the number of subdomains without increasing the parameter count allows it to learn and represent complex dynamics across various scales effectively.

References

[1] Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *international conference on machine learning*, pages 2402–2411. PMLR, 2020.

[2] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artificial neural networks. *European Journal of Applied Mathematics*, 32(3):421–435, 2021.

[3] Nicolas Boullé and Alex Townsend. A mathematical guide to operator learning. *arXiv preprint arXiv:2312.14688*, 2023.

[4] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[5] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[6] Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, pages 1–9, 2024.

[7] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[8] Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pages 12556–12569. PMLR, 2023.

[9] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022.

[10] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

[11] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 1(3):1–27, 2024.

[12] Milad Ramezankhani, Anirudh Deodhar, Rishi Yash Parekh, and Dagnachew Birru. An advanced physics-informed neural operator for comprehensive design optimization of highly-nonlinear systems: An aerospace composites processing case study. *arXiv preprint arXiv:2406.14715*, 2024.

[13] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

[14] 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):62, 2023.

[15] Patrik Simon Hadorn. Shift-deeponet: Extending deep operator networks for discontinuous output functions. ETH Zurich, Seminar for Applied Mathematics, 2022.

[16] Jacob Seidman, Georgios Kissas, Paris Perdikaris, and George J Pappas. Nomad: Nonlinear manifold decoders for operator learning. *Advances in Neural Information Processing Systems*, 35:5601–5613, 2022.

[17] Jae Yong Lee, Sung Woong Cho, and Hyung Ju Hwang. Hyperdeeponet: learning operator with complex target function space using the limited resources via hypernetwork. *arXiv preprint arXiv:2312.15949*, 2023.

[18] Maziar Raissi, Paris Perdikaris, and George 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.

[19] Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. An expert's guide to training physics-informed neural networks. *arXiv preprint arXiv:2308.08468*, 2023.

[20] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116813, 2024.

[21] Ameya D Jagtap and George Em 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), 2020.

[22] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[23] Johannes Von Oswald, Christian Henning, Benjamin F Grewe, and João Sacramento. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

[24] 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, 2024.

[25] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[26] C Basdevant, M Deville, P Haldenwang, J.M Lacroix, J Ouazzani, R Peyret, P Orlandi, and A.T Patera. Spectral and finite difference solutions of the burgers equation. *Computers Fluids*, 14(1):23–41, 1986.

A Additional resutls

High-frequency harmonic oscillator. We considered a case of high-frequency harmonic oscillator ($\omega_0 = 80$) and trained the operator to learn the mapping between the initial condition (various pairs of initial displacement and initial velocity $[u_0, v_0]$) and the mass displacement over time. As illustrated in Figure A.1, both the DON and HDON models were unsuccessful in learning the solution operator and capturing the high-frequency characteristics of the system. In contrast, the FB-HyDON model effectively harnessed its domain decomposition capabilities to accurately predict the solution and fully encompass the system's nonlinearities. For this example, 12 subdomains were used for the training of FB-HyDON.

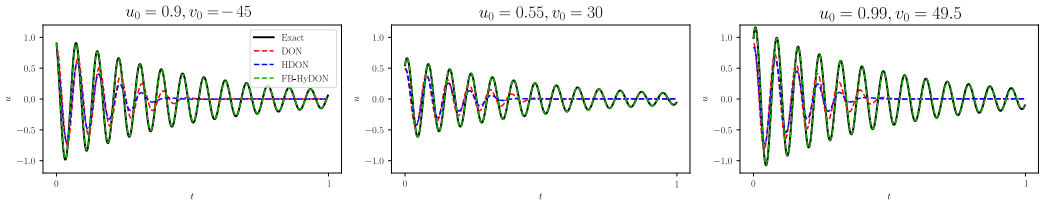

Figure A.1: Comparison of operators' prediction for solving the high-frequency harmonic oscillator with $w_0 = 80$ via physics-informed training. Each subplot represents a unique initial condition pair $[u_0, v_0]$. Both DON and HDON failed to learn the solution operator, while FB-HyDON was able to capture the system's nonlinearities and accurately predict the solution leveraging its domain decomposition capabilities. 12 subdomains were used to train FB-HyDON.

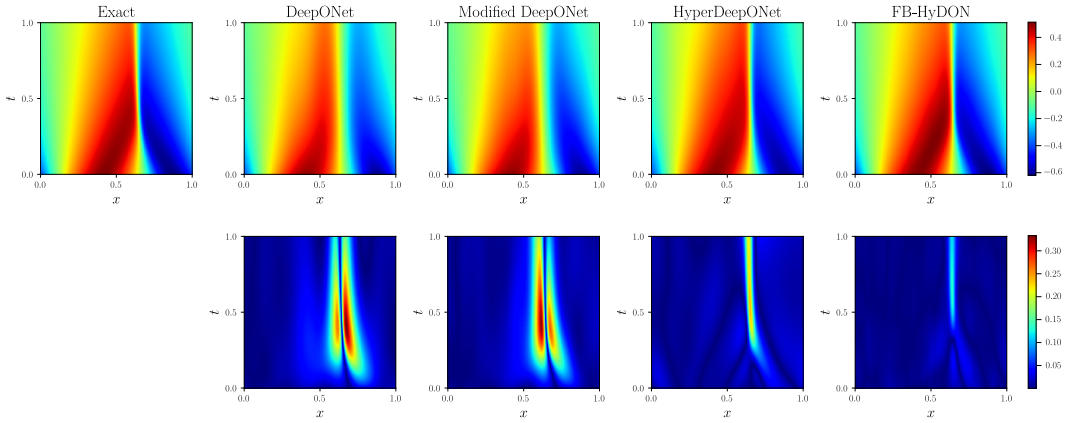

Figure A.2: Comparison of operators' prediction (top row) and absolute error (bottom row) for solving the Burger's equation with $\nu = 0.005$ via physics-informed training. DeepONet and its modified version produced the largest error while having more trainable parameters. FB-HyDON (ours) outperformed other models and generated the most accurate predictions.

Effect of the number of subdomains. Our results reveal a nuanced relationship between the number of subdomains and FB-HyDON's performance, given a fixed parameter count, as shown in Table 4. Initially, increasing the subdomain count leads to consistent performance improvements. However, this trend eventually reverses, with performance degrading beyond a certain threshold (here, 16 subdomains). This means that finer subdomain partitioning does not necessarily lead to model improvement. We hypothesize that there are two main factors contributing to this phenomenon. First, as the number of subdomains grows, the hypernetwork requires greater capacity to effectively manage the intricacies of each subdomain. With a fixed parameter budget, this increased demand may not be adequately met. Second, in configurations with numerous subdomains, the flow of information from the boundary and initial subdomains to central regions becomes increasingly challenging, which can result in suboptimal performance, as observed in [24].

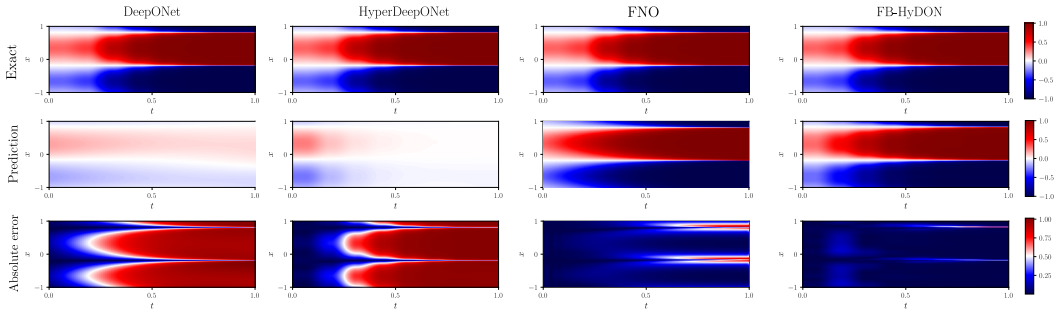

Figure A.3: Comparison of operators' prediction (middle row) and absolute error (bottom row) for solving the Allen-Cahn equation with $\epsilon = 0.0001$ via physics-informed training. The ground truth solution is depicted on the top row. FB-HyDON is trained using 8 subdomains.

# of Subdomains	4	8	16	32
Rel. L_2	0.0486	0.0482	**0.0416**	0.0534
MAE	0.0343	**0.0341**	0.0369	0.0392

Table 4: Effect of number of subdomains on the performance of FB-HyDON.

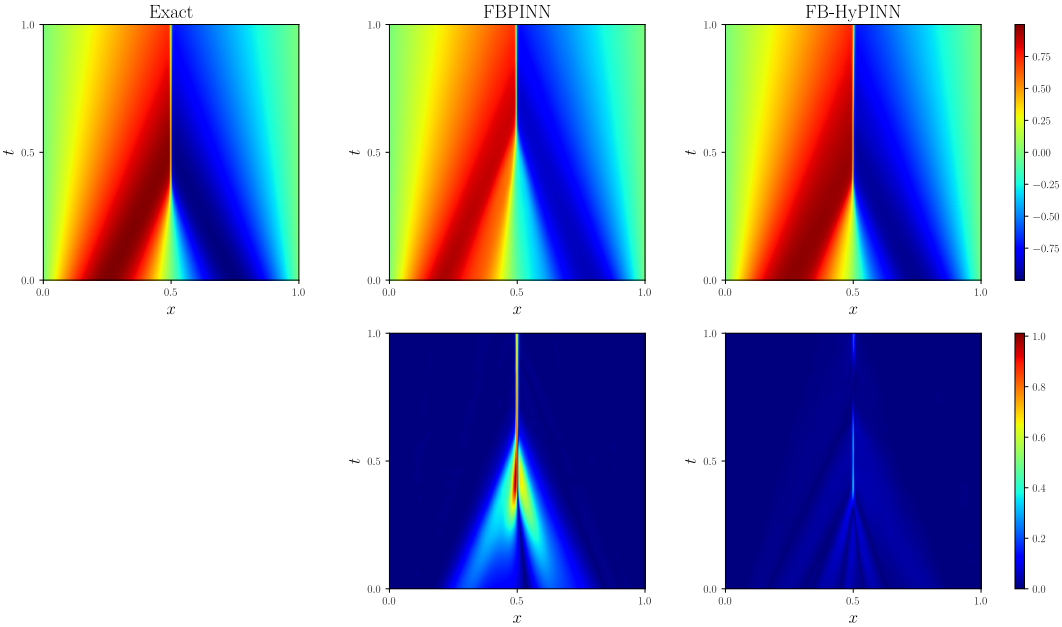

Figure A.4: Comparison of FBPINN and FB-HyPINN's predictions (top row) and absolute error (bottom row) for solving the Burger's equation with $\nu = 0.001/\pi$ via physics-informed learning with domain decomposition. Both model has 25 subdomains. The prediction of FB-HyPINN near the discontinuity is better in terms of absolute error.

B Additional background

B.1 DeepONet

In DeepONet's architecture, $\mathcal{G}_\theta$ comprises a branch net and a trunk net. The branch net takes the input function values at sensor points $a = [a(x_1), a(x_2), \ldots, a(x_m)]$ as input and generates a finite-dimensional feature embedding $b = [b_1, b_2, \ldots, b_q]^T \in \mathbb{R}^q$ as output. Similarly, the trunk net encodes the spatiotemporal coordinates of the PDE system y to a feature representation $t = [t_1, t_2, \ldots, t_q]^T \in \mathbb{R}^q$ dimensionally consistent with the branch net's output. The output of the branch and trunk nets are then merged through an element-wise multiplication followed by a summation to generate $G_\theta(a)(y) = \sum_{k=1}^{q} b_k t_k + b_0$. The DeepONet can be trained in a supervised learning manner by minimizing the error between the model's predicted output and the actual operator solution across a range of training input functions [4].

B.2 Finite basis domain decomposition using hypernetwork

In FBPINN's framework, the input domain Ω is partitioned into J overlapping subdomains such that $\Omega = \bigcup_{j=1}^{J} \Omega_j$ and the intersection of any two adjacent subdomains is non-empty, i.e., $\Omega_j \cap \Omega_k \neq \emptyset$ for k being an adjacent subdomain to j. A family of sub-networks, denoted as V, is defined as:

$$V = \{\hat{u}_j(x, \theta_j) \mid x \in \Omega, \theta_j \in \Theta_j\}_{j=1}^{J} \tag{2}$$

Each subnetwork $\hat{u}_j(x, \theta_j)$ corresponds to a specific subdomain Ω_j. These subnetworks operate independently to learn the solution relevant to their assigned subdomain. According to the FBPINN scheme [14], the inputs to the subnetworks are normalized separately over each subdomain, which mitigates the issue of spectral bias. For every collocation point x_i, the outputs of the subnetworks are generated and multiplied with a window value $\omega_j(x_i)$ determined by their respective window function and summed across all subdomains. An example and description of the window functions and subdomains are provided in Figure A.5. The primary role of window functions is to bound each subnetwork to its subdomain by introducing a higher weight near the center of the corresponding subdomain and zero outside of it.

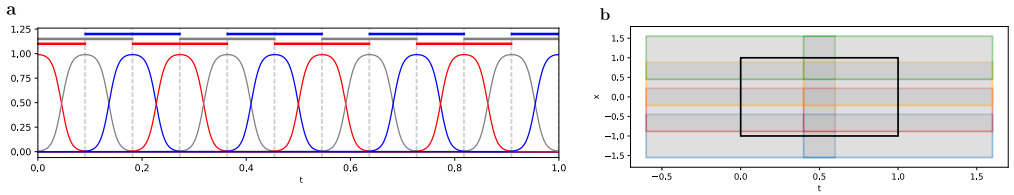

Figure A.5: a) Visualization of a set of window functions for harmonic oscillator case where time domain is divided into 12 subdomains. Different colors represent different subdomains and all subdomains are associated with window function based on their bounds. X-axis represents the time domain and y-axis shows window function values. Horizontal bars on the top represent the length of subdomains. b) Visualization of domain decomposition for time-space domain of 1D Allen-Cahn example. Black box represents the problem domain and overlapping subdomains are shown with gray rectangles.

As described in Section 3.1, in FB-HyPINN, a hypernetwork $\hat{h}(d, \theta_H)$ is used to generate the parameters $\{\theta_j\}_{j=1}^{J}$ of the subnetworks $\{\hat{u}_j\}_{j=1}^{J}$ assigned to the subdomains. Thus the global solution on the entire domain is achieved by a weighted summation of subnetworks output $\hat{u}_j(x_i^{norm}, \theta_j)$. In other words, for each collocation point x_i the solution is calculated as:

$$\hat{u}(x_i, \theta_H) = \sum_{j=1}^{J} \omega_j(x_i) \hat{u}_j(x_i^{norm}, \theta_j) \tag{3}$$

Table 5: Architecture of FB-HyPINN model used for training on sinusoidal wave and 1D Burgers' equation.

Problem	Number of Subdomains	Hypernet	AF	Target Net	AF	Params
Sinusoidal	28	$[16] \times 6$	ReLU	$[16] \times 2$	tanh	6865
1D Burgers	25	$[12] \times 6$	ReLU	$[16] \times 2$	tanh	5221

Table 6: Architecture of the baseline FBPINN model used for training on sinusoidal wave and 1D Burgers' equation.

Problem	Number of Subdomains	AF	Target Net	AF	Params
Sinusoidal	28	ReLU	$[16] \times 2$	tanh	8988
1D Burgers	25	ReLU	$[16] \times 2$	tanh	8425

where ω_j denotes the window function. This significantly improves the efficiency and scalability of the domain decomposition in learning nonlinear PINNs. Particularly, in a given PDE of the form

$$\mathcal{N}[u](x) = 0, x \in \Omega \tag{4}$$
$$u(x) = g(x), x \in \partial\Omega, \tag{5}$$

$\mathcal{N}[.]$ denotes a differential operator, $u(x) : \overline{\Omega} \to \mathbb{R}$ is the unknown solution, $x = \{x_i\}_{i=1}^{N_I}$ denotes a collection of collocation points sampled within the domain interior and $\{x_j\}_{j=1}^{N_B}$ represents a set of points sampled along each boundary condition. Additionally, λ_I and λ_B are appropriately selected scalar weights. Just like a regular PINN framework, here, the solution $u(x)$ is approximated by $\hat{u}(x_i; \theta_H)$ via the defined physics and boundary losses and optimizing the hypernetwork parameters θ_H. The loss is represented as:

$$\mathcal{L}(\theta_H) = \frac{\lambda_I}{N_1} \sum_{i=0}^{N_1} (\mathcal{N}[\hat{u}(x_i, \theta_H)])^2 + \frac{\lambda_B}{N_B} \sum_{i=0}^{N_B} (\hat{u}(x_i, \theta_H) - g(x_i))^2. \tag{6}$$

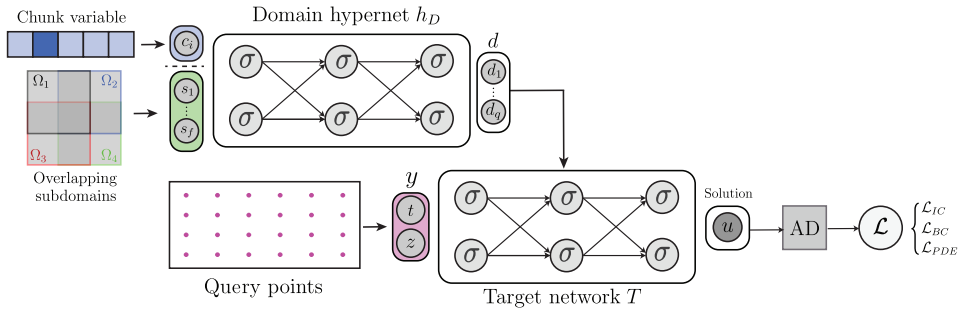

Figure A.6: Schematic of proposed FB-HyPINN, a finite basis domain decomposition approach with hypernetwork. The hypernetwork is responsible for generating the weights of the subnetwork associated with each overlapping subdomain ω_i. As opposed to FB-HyDON which is designed for operator learning, FB-HyPINN is a parameter-efficient variant of FBPINN, used as part of the FB-HyDON architecture to facilitate the domain decomposition task.

C Model implementation and experiments details

Physics-informed domain decomposition experiments. The architecture details of FB-HyPINN and FBPINN are given in Table 5 and 6 respectively. Both models are implemented in JAX [25]. For FBPINN, we developed a version of the model taking FBPINN's open-source code repository [2] as reference.

For the high-frequency sinusoidal case, we trained both models for 50k epochs. Both models were trained using Adam optimizer with a learning rate of 1×10^{-3} and a decay rate of 0.95 per 1000

[2]https://github.com/benmoseley/FBPINNs

Table 7: Architecture of FB-HyDON used for training on harmonic oscillator, 1D Burgers' equation and 1D Allen-Cahn equation. 1 and 2 in the model name refer to the number of hypernetworks used in the model architecture.

Problem	Model	Hypernet	AF	Target Net	AF	Params
Oscillator	FB-HyDON-1	$[32] \times 5$	ReLU	$[16] \times 2$	tanh	6.5k
1D Burgers	FB-HyDON-1	$[90] \times 6$	ReLU	$[32] \times 4$	tanh	95k
$\nu = 0.01$	FB-HyDON-2	$[64] \times 5 + [64] \times 3$	ReLU	$[32] \times 4$	tanh	99k
1D Burgers	FB-HyDON-1	$[100] \times 6$	ReLU	$[32] \times 5$	tanh	128k
$\nu = 0.005$	FB-HyDON-2	$[64] \times 5 + [64] \times 5$	ReLU	$[32] \times 5$	tanh	130k
1D Allen-Cahn	FB-HyDON-1	$[100] \times 6$	ReLU	$[45] \times 5$	tanh	197k
	FB-HyDON-2	$[90] \times 5 + [90] \times 5$	ReLU	$[32] \times 5$	tanh	202k

Table 8: Architecture of baseline models used for training on harmonic oscillator and 1D Burgers' equation and 1D Allen-Cahn equation.

Problem	Model	Branch/hypernet	AF	Trunk/target Net	AF	Params
Oscillator	DON	$[32] \times 4$	tanh	$[32] \times 4$	tanh	7.7k
	MDON	$[32] \times 4$	tanh	$[32] \times 4$	tanh	10k
	HDON	$[32] \times 5$	ReLU	$[16] \times 2$	tanh	6.1k
1D Burgers	DON	$[128] \times 7$	tanh	$[128] \times 7$	tanh	203k
	MDON	$[128] \times 7$	tanh	$[128] \times 7$	tanh	215k
$\nu = 0.01$	HDON	$[90] \times 6$	ReLU	$[32] \times 4$	tanh	95k
1D Burgers	DON	$[128] \times 7$	tanh	$[128] \times 7$	tanh	203k
	MDON	$[128] \times 7$	tanh	$[128] \times 7$	tanh	215k
$\nu = 0.005$	HDON	$[100] \times 6$	ReLU	$[32] \times 5$	tanh	128k
1D Allen-Cahn	DON	$[150] \times 7$	tanh	$[150] \times 7$	tanh	278k
	MDON	$[150] \times 7$	tanh	$[150] \times 7$	tanh	291k
	HDON	$[100] \times 6$	ReLU	$[50] \times 5$	tanh	229k

epochs. We randomly sample 200 collocation points from the input domain for training and 1000 equally spaced points for testing. The domain was decomposed into 28 overlapping subdomains with a subdomain length of 0.47. The frequency ω of the sinusoidal function was taken as 15.

For 1D Burgers Equations, we trained both models for 50k epochs. Both models were trained using Adam optimizer with a learning rate of 1×10^{-3} and a decay rate of 0.95 per 5000 epochs. For training randomly we sample 40,000 collocation points, 200 BC points, and 200 IC points from the input domain. Testing was done on 160,000 equally spaced points along both input dimensions (400×400). The domain was decomposed into 5 subdomains along the spacial dimension and 5 across the temporal dimension giving a total of 25 overlapping subdomains. The length of the subdomains was selected as 0.35 along temporal dimension and 0.7 across the spacial dimension

Operator learning experiments. Details regarding the architectures of FB-HyDON-1 and FB-HyDON-2 are provided in Table 8. Both models are developed in Jax [25]. We implemented the baseline models using the open-source implementation of physics-informed DeepONet and modified DeepONet [3]. For HDON, we developed a physics-informed implementation based on the paper [17] in Jax.

For both 1D Burgers' and Allen-Cahn equations, we trained DeepONet and its variants for 25k iterations with a batch size of 1000. 1000 training and 100 test input functions (initial conditions) are randomly generated using a mean-zero Gaussian random field GRF $\sim \mathcal{N}\left(0, 25^2\left(-\Delta + 5^2 I\right)^{-4}\right)$ and used for training and evaluation of the operators. 40 equally-spaced sensor points were used for encoding the input function information. All models were trained using Adam optimizer with an initial learning rate of 1×10^{-3} and an exponential decay rate of 0.9 per 1000 epochs. For each training input function, 2500 collocation points, 100 IC points and 100 BC points were uniformly

[3] https://github.com/PredictiveIntelligenceLab/Physics-informed-DeepONets

Table 9: Architecture of FNO models used for training on 1D Burgers' and Allen-Cahn equations.

Problem	Model	Fourier layers	Mode 1	Mode 2	AF	Params
1D Burgers	FNO-small	$[32] \times 3$	5	5	GeLU	113k
$\nu = 0.01$	FNO-large	$[32] \times 4$	5	5	GeLU	165k
1D Burgers	FNO-small	$[32] \times 3$	5	5	GeLU	113k
$\nu = 0.005$	FNO-large	$[32] \times 4$	5	5	GeLU	165k
1D Burgers	FNO-small	$[32] \times 4$	5	5	GeLU	165k
$\nu = 0.001$	FNO-large	$[32] \times 5$	5	5	GeLU	217k
1D Allen-Cahn	FNO-small	$[32] \times 5$	5	5	GeLU	217k
	FNO-large	$[32] \times 6$	5	5	GeLU	270k

sampled and used for training. For models with hypernetwork, we used the chunking strategy and set the number of chunks to 6.

For FNO models, we followed the training procedure used in [11]. In particular, all models were trained for 500 epochs with a batch size of 20 (each batch representing a separate training sample.) Adam optimizer with a learning rate of 0.001 and a decay rate of $1/2$ per 100 epochs was used. To ensure a fair comparison with DeepONet-based models, we trained and evaluated the FNO models on resolution 50×50. This results in a total of 2500 collocation points, equal to that of DeepONet models. Similarly, FNO models were exposed to the same 1000 GRF initial conditions sampled at 50 locations. Details regarding the architecture of FNO models are summarized in Table 9.

For the harmonic oscillator, the models were trained for 100k iterations on the full batch of data. We sampled 200 initial condition pairs $[u_0, v_0]$ for training and three for testing from $[0, 1]$ for initial displacement u_0 and $[-50, +50]$ for initial velocity v_0. Adam optimizer with an initial learning rate of 1×10^{-3} and a decay rate of 0.9 per 10k epochs was used for all cases. 500 collocation points and 1 initial condition point were used to minimize the physics-informed losses.

D Benchmark problems setting

High-frequency sinusoidal. Following [14], this case represents a 1D scenario where neural networks often encounter spectral bias, necessitating a domain decomposition approach. Considering the following problem:

$$\frac{du}{dx} = \cos(\omega x),$$
$$u(0) = 0,$$

where $x, u, \omega \in \mathbb{R}$ is the frequency of the wave. The exact solution to this problem is given by:

$$u(x) = \frac{1}{\omega} \sin(\omega x).$$

1D Burgers' equation for FB-HyPINNs. We tested the proposed FB-HyPINN and FBPINNs on 1D viscous time-dependent Burgers equation, given by:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2},$$

with initial and boundary conditions,

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

where $x, t, u, \nu \in \mathbb{R}$, and the problem domain is $x \in [-1, 1]$ and $t \in [0, 1]$. Notably, for small values of the viscosity parameter ν, specifically $\nu = \frac{0.01}{\pi}$, the solution develops a discontinuity at $x = 0$ as

time progresses. The exact solution can be obtained from open source code repository [4], that uses the Hopf-Cole transform as explained in [26].

Harmonic Oscillator for operator learning. We solved the 1D damped harmonic oscillator, where we aim to model the displacement $u(t)$ of the oscillator over time. The problem is defined by the following ordinary differential equation (ODE):

$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + ku = 0, \tag{7}$$

where m is the mass of the oscillator, μ is the damping coefficient, and k is the spring constant. Considering $\delta = \frac{\mu}{2m}$ and $\omega_0 = \sqrt{\frac{k}{m}}$, we investigated the under-damped state where $\delta < \omega_0$ [14]. With the initial conditions $u(0) = u_0$ and $\frac{du}{dt}(0) = v_0$ the exact solution will be:

$$u(t) = e^{-\delta t}\left(u_0\cos(\omega t) + \frac{v_0 + \delta u_0}{\omega}\sin(\omega t)\right). \tag{8}$$

1D Burgers' equation for operator learning. To demonstrate the capability of our proposed operator in addressing nonlinearities in governing PDEs, we utilized the 1D Burgers' equation benchmark, following the setup described in [5]. The equation is given by:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0, \quad (x,t) \in (0,1)\times(0,1], \tag{9}$$

$$u(x,0) = a, \quad x \in (0,1), \tag{10}$$

with periodic boundary conditions:

$$u(0,t) = u(1,t), \tag{11}$$

$$\frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(1,t). \tag{12}$$

In our experiments, we set the viscosity to $\nu = 0.01$ and $\nu = 0.005$ and generated the initial condition a from a GRF. To generate the solution test data, we followed the procedure outlined in [10]. We solved the 1D Burgers' equation using conventional spectral methods, assuming periodic boundary conditions. The initial condition $s(x,0) = u(x)$ was integrated up to a final time $t = 1$, with temporal snapshots of the solution recorded at regular intervals.

1D Allen-Cahn for operator learning. For 1D Allen-Cahn equation with periodic boundary conditions, we followed the setup described in [18]:

$$\frac{\partial u}{\partial t} - 0.0001\frac{\partial^2 u}{\partial x^2} + 5u^3 - 5u = 0, \quad x \in [-1,1], \quad t \in [0,1], \tag{13}$$

$$u(x,0) = a, \tag{14}$$

$$u(-1,t) = u(1,t), \tag{15}$$

$$\frac{\partial u}{\partial x}(-1,t) = \frac{\partial u}{\partial x}(1,t). \tag{16}$$

[4] https://github.com/benmoseley/FBPINNs/tree/main/fbpinns/traditional_solutions/analytical