Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar for Applied Mathematics

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

V. Dolean and A. Heinlein and S. Mishra and B. Moseley

Research Report No. 2023-26
June 2023

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

MULTILEVEL DOMAIN DECOMPOSITION-BASED ARCHITECTURES FOR PHYSICS-INFORMED NEURAL NETWORKS

VICTORITA DOLEAN*, ALEXANDER HEINLEIN†, SIDDHARTHA MISHRA‡, AND BEN MOSELEY§

Abstract. Physics-informed neural networks (PINNs) are a popular and powerful approach for solving problems involving differential equations, yet they often struggle to solve problems with high frequency and/or multi-scale solutions. Finite basis physics-informed neural networks (FBPINNs) improve the performance of PINNs in this regime by combining them with an overlapping domain decomposition approach. In this paper, the FBPINN approach is extended by adding multiple levels of domain decompositions to their solution ansatz, inspired by classical multilevel Schwarz domain decomposition methods (DDMs). Furthermore, analogous to typical tests for classical DDMs, strong and weak scaling studies designed for measuring how the accuracy of PINNs and FBPINNs behaves with respect to computational effort and solution complexity are carried out. Our numerical results show that the proposed multilevel FBPINNs consistently and significantly outperform PINNs across a range of problems with high frequency and multi-scale solutions. Furthermore, as expected in classical DDMs, we show that multilevel FBPINNs improve the scalability of FBPINNs to large numbers of subdomains by aiding global communication between subdomains.

Key words. Physics-informed neural networks, overlapping domain decomposition methods, multilevel methods, multi-scale modeling, spectral bias, forward modeling, differential equations

1. Introduction.

Scientific machine learning (SciML) [3, 52, 12, 2, 35] is an emerging and rapidly growing field of research. The central goal of SciML is to provide accurate, efficient, and robust tools for carrying out scientific research by tightly combining scientific understanding with machine learning (ML). The field has provided many such tools which have enhanced traditional approaches, from accelerating simulation algorithms to discovering new scientific phenomena.

One popular SciML approach are physics-informed neural networks (PINNs) [25, 40]. PINNs solve forward and inverse problems related to differential equations by using a neural network to directly approximate the solution to the differential equation. They are trained by using a loss function which minimizes the residual of the differential equation over a set of collocation points. The initial concepts behind PINNs were introduced in [25], and later re-implemented and extended in [40]. One of the advantages of PINNs over traditional methods for solving differential equations such as finite difference (FD) and finite element methods (FEM) is that they provide a mesh-free approach, paving the way for the application of problems with complex geometry or in very high spatial dimensions; cf. [33]. Furthermore, they can easily be extended to solve inverse problems by incorporating observational data.

Since their invention, PINNs have been employed across a wide range of domains [12, 22]. For example, they have been used to solve forward and inverse problems in geophysics [36], fluid dynamics [21, 6, 41], and optics [10]. Many extensions of PINNs have also been proposed. For example, PINNs have been extended to carry out uncertainty quantification [54], learn fast surrogate models [49, 55], and carry out

*University of Strathclyde, UK.

†Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands. a.heinlein@tudelft.nl

‡Seminar for Applied Mathematics (SAM) D-MATH, and ETH AI Center, ETH Zürich, Switzerland. smishra@ethz.ch

§Seminar for Applied Mathematics (SAM) D-MATH, and ETH AI Center, ETH Zürich, Switzerland. benjamin.moseley@ai.ethz.ch

equation discovery [11].

However, PINNs suffer from a number of limitations. One is that, compared to traditional methods, their convergence properties are poorly understood, although some work have started to explore this [34, 42, 51]. Another limitation is that, compared to traditional methods, the computational cost of training PINNs is relatively high, especially when they are only used for forward modeling [22]. Finally, a major limitation of PINNs is that they often struggle to solve problems with high frequency and/or multi-scale solutions [37, 50]. Typically, as higher frequencies and multi-scale features are added to the solution, the accuracy of PINNs usually rapidly reduces and their computational cost rapidly increases in a super-linear fashion [37].

There are multiple reasons for this behavior. One is the spectral bias of neural networks, which is the well-studied property that neural networks tend to learn high frequencies much slower than low frequencies [53, 39, 4, 9]. Another is that, as higher frequencies and more multi-scale features are added, more collocation points and a larger neural network with significantly more free parameters are typically required to accurately approximate the solution. This creates a significantly more complex optimization problem when training the PINN.

Recently, [37] proposed finite basis physics-informed neural networks (FBPINNs), which aim to improve the performance of PINNs in this regime by using an overlapping domain decomposition (DD) approach. In particular, instead of using a single neural network to approximate the solution to the differential equation, many smaller neural networks were placed in overlapping subdomains and summed together to represent the solution. On the one hand, FBPINNs can be seen as a domain decomposition-based network architecture for PINNs. On the other hand, by taking this "divide and conquer" approach, the global PINN optimization problem is transformed into many smaller local optimization problems, which are coupled implicitly due to the overlap of the subdomains and their globally defined loss function. The results in [37] show that this significantly improves the accuracy and reduces the training cost of PINNs when solving differential equations with high frequency and multi-scale solutions.

In this work, we significantly extend FBPINNs by incorporating *multilevel modeling* into their design. In particular, instead of using a single domain decomposition in their solution ansatz, we add multiple levels of overlapping domain decompositions. This idea is inspired by classical DDMs, where coarse levels are required for numerical scalability when using large numbers of subdomains. Furthermore, to assess the performance of multilevel FBPINNs, we define strong and weak scaling tests for measuring how the accuracy of PINNs and FBPINNs scale with computational effort and solution complexity, analogous to the strong and weak scaling tests commonly used in classical DDMs.

Given these extensions, the performance of PINNs, (one-level) FBPINNs, and multilevel FBPINNs across a range of high frequency and multi-scale problems is investigated. Across all these tasks, we find that multilevel FBPINNs significantly outperform both PINNs and FBPINNs in terms of accuracy and computational cost. As expected in classical DDMs, we show that multilevel FBPINNs improve the scalability of FBPINNs when a large number of subdomains are used by aiding global communication between subdomains.

The remainder of this work is structured as follows. In subsection 1.1, we discuss related work on combining ML, PINNs, and DD, and in subsections 1.2 and 1.3, we give a brief overview of neural networks and PINNs. Then, we define FBPINNs and extend them to multilevel FBPINNs in section 2. Our strong and weak scaling tests and corresponding numerical results on the performance of PINNs, FBPINNs, and multilevel FBPINNs across a range of high frequency and multi-scale problems are discussed in section 3. Finally, in section 4, we discuss the implications and limitations of our work and further research directions.

1.1. Related work. In general, the idea of combing ML with classical DDMs is not new; for early works on using ML to predict the geometrical location of constraints in adaptive finite element tearing and interconnecting (FETI) and balancing domain decomposition by constraints (BDDC) methods; see [17]. An overview of the first attempts on combining DD and ML can be found in [18].

For specifically combining PINNs with DD, some of the first methods in this area were the deep domain decomposition method (D3M) [27], the deep-learning-based domain decomposition method (DeepDDM) [29, 28], and its two-level variant [32], which use PINNs to solve local problems and overlapping Schwarz steps to iteratively connect them based on Lions' parallel Schwarz algorithm [30]. At the same time, a series of other extensions, like cPINNs and XPINNs [20] were proposed, which similarly divide the domain and use PINNs to solve each local problem; here, typically a nonoverlapping domain decomposition is used. The advantage of all these methods is their high potential for parallelism, but the downside is the increasing complexity of the local loss functions as additional terms are required to enforce coupling between subdomains.

In contrast, FBPINNs do not require local loss functions nor any additional loss terms since they use a globally-defined solution ansatz and loss function [37]. Gated-PINNs introduced in [45] are perhaps most similar to FBPINNs, where several local networks, called experts, are used for training and the domain decomposition itself is learned for better efficiency. The idea of learnable domains was also recently exploited in XPINNs to improve their performance [19].

1.2. Neural networks. We first provide a basic definition of a neural network. For the purpose of this work, we simply consider a neural network to be a mathematical function with some learnable parameters. More precisely, the network is defined as $u(\mathbf{x}, \boldsymbol{\theta}) : \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_u}$, where $\mathbf{x}$ are some inputs to the network, $\boldsymbol{\theta}$ are a set of learnable parameters, and d_x, d_θ, and d_u are the dimensionality of the network's inputs, parameters, and outputs. In a traditional supervised learning setting, learning typically consists of fitting the network function to some training data containing example inputs and outputs, by minimizing a loss function with respect to $\boldsymbol{\theta}$ which penalizes the difference between the network's outputs and the training data.

The exact form of the network function is determined by the neural network's architecture. In this work, we solely use feedforward fully connected networks (FCNs) [16]. In this case, the network function is given by

$$u(\mathbf{x}, \boldsymbol{\theta}) = f_n \circ ... \circ f_i \circ ... \circ f_1(\mathbf{x}, \boldsymbol{\theta}) \tag{1.1}$$

where now $\mathbf{x} \in \mathbb{R}^{d_0}$ is the input to the FCN, $u \in \mathbb{R}^{d_n}$ is the output of the FCN, n is the number of layers (depth) of the FCN, and $f_i(\mathbf{x}, \boldsymbol{\theta}) = \sigma_i(W_i\mathbf{x}+\mathbf{b}_i)$ where $\boldsymbol{\theta}_i = (W_i, \mathbf{b}_i)$, $W_i \in \mathbb{R}^{d_i \times d_{i-1}}$ are known as weight matrices, $\mathbf{b}_i \in \mathbb{R}^{d_i}$ are known as bias vectors, σ_i are element-wise activation functions commonly chosen as rectified linear unit (ReLU), hyperbolic tangent, or identity functions, and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_i, ..., \boldsymbol{\theta}_n)$ are the set of learnable parameters of the network. Note that only the nonlinear activation functions σ_i facilitate nonlinearity of the network function.

1.3. Physics-informed neural networks. Physics-informed neural networks (PINNs) [25, 40] use neural networks to solve problems related to differential equations. In particular, PINNs focus on solving boundary value problems of the form

$$\begin{aligned} \mathcal{N}[u](\mathbf{x}) &= f(\mathbf{x}), \qquad \mathbf{x} \in \Omega \subset \mathbb{R}^d, \\ \mathcal{B}_k[u](\mathbf{x}) &= g_k(\mathbf{x}), \qquad \mathbf{x} \in \Gamma_k \subset \partial\Omega \end{aligned} \tag{1.2}$$

where $\mathcal{N}[u](\mathbf{x})$ is some differential operator, $u(\mathbf{x})$ is the solution, and $\mathcal{B}_k(\cdot)$ are a set of boundary conditions (BCs) which ensure uniqueness of the solution. For the sake of simplicity, we consider BCs in a broad sense; we do not explicitly distinguish between initial and boundary conditions, and the $\mathbf{x}$ variable can include time. Equation (1.2) can describe many different differential equation problems, including linear and non-linear problems, time-dependent and time-independent problems, and those with irregular, higher-order, and cyclic boundary conditions.

To solve (1.2), PINNs use a neural network to directly approximate the solution, i.e., $u(\mathbf{x}, \boldsymbol{\theta}) \approx u(\mathbf{x})$. Note, for simplicity throughout this work, we use the same notation for the true solution and the neural network. It is important to note that PINNs provide a functional approximation to the solution, and not a discretized solution similar to that provided by traditional methods such as finite difference methods, and as such PINNs are a mesh-free approach for solving differential equations. Following the approach proposed by [40], the following loss function is minimized to train the PINN,

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{\lambda_I}{N_I} \sum_{i=1}^{N_I} \big(\underbrace{\mathcal{N}[u](\mathbf{x}_i, \boldsymbol{\theta}) - f(\mathbf{x}_i)}_{\text{PDE residual}} \big)^2 + \sum_{k=1}^{N_k} \frac{\lambda_B^k}{N_B^k} \sum_{i=1}^{N_B^k} \big(\underbrace{\mathcal{B}_k[u](\mathbf{x}_i^k, \boldsymbol{\theta}) - g_k(\mathbf{x}_i^k)}_{\text{BC residual}} \big)^2. \tag{1.3}$$

where $\{\mathbf{x}_i\}_{i=1}^{N_I}$ is a set of collocation points sampled in the interior of the domain, $\{\mathbf{x}_j^k\}_{j=1}^{N_B^k}$ is a set of points sampled along each boundary condition, and λ_I and λ_B^k are well-chosen scalar weights that ensure the terms in the loss function are well balanced. Intuitively, one can see that by minimizing the PDE residual, the method tries to ensure that the solution learned by the network obeys the underlying PDE, and by minimizing the BC residual, the method tries to ensure that the learned solution is unique by matching it to the BCs. Importantly, a sufficient number of collocation and boundary points must be chosen such that the PINN is able to learn a consistent solution across the domain.

Iterative schemes are typically used to optimize this loss function. Usually, variants of the gradient descent (GD) method, such as the Adam optimizer [24], or quasi-Newton methods, such as the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [31] are employed. These methods require the computation of the gradient of the loss function with respect to the network parameters, which can computed easily and efficiently using automatic differentiation [23] provided in modern deep learning libraries [1, 38, 5]. Note that gradients of the network output with respect to its inputs are also typically required to evaluate the PDE residual in the loss function, and can similarly be obtained and further differentiated through to update the network's parameters using automatic differentiation.

1.3.1. Hard constrained PINNs. A downside of training PINNs with the loss function given by (1.3) is that the BCs are *softly* enforced. This means the learned solution may deviate from the BCs because the BC term may not be fully minimized. Furthermore, it can be challenging to balance the different objectives of the PDE and BC terms in the loss function, which can lead to poor convergence and solution accuracy [51, 46]. An alternative approach, as originally proposed by

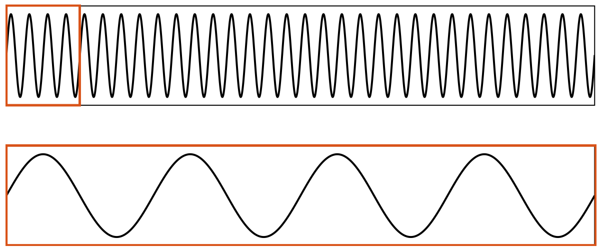

FIG. 1. *Scaling high frequency problems to low frequency problems using domain decomposition. FBPINNs decompose the domain into many subdomains, and use neural networks within each subdomain to learn the local solution. The input coordinates to each network are normalized to the range [-1,1] over their individual subdomains. When solving problems with high frequency solutions, this effectively scales each local problem from a high frequency problem to a lower frequency problem, and helps reduce the network's spectral bias.*

[25], is to enforce BCs in a *hard* fashion by using the neural network as part of a solution ansatz. More precisely, the solution to the differential equation is instead approximated by $[\mathcal{C}u](\mathbf{x}, \boldsymbol{\theta}) \approx u(\mathbf{x})$ where $\mathcal{C}$ is an appropriately selected constraining operator which analytically enforces the BCs [37, 26].

To give a simple example, suppose we want to enforce $u(x = 0) = 0$ when solving a one-dimensional ordinary differential equation (ODE). The constraining operator and solution ansatz could be chosen as $[\mathcal{C}u](x, \boldsymbol{\theta}) = \tanh(x)u(x, \boldsymbol{\theta}) \approx u(\mathbf{x})$. The rationale behind this is that the function $\tanh(x)$ is zero at 0, forcing the BC to always be obeyed, but non-zero away from 0, allowing the network to learn the solution away from the BC.

In this approach, the BCs are always satisfied and therefore the BC term in the loss function (1.3) can be removed, meaning that the PINN can be trained using the simpler unconstrained loss function,

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} (\mathcal{N}[\mathcal{C}u](\mathbf{x}_i, \boldsymbol{\theta}) - f(\mathbf{x}_i))^2. \tag{1.4}$$

where $\{\mathbf{x}_i\}_{i=1}^N$ is a set of collocation points sampled in the interior of the domain. Note that, in general, there is no unique way of choosing the constraining operator, and the definition of a suitable constraining operator for complex geometries and/or complex BCs may be difficult or sometimes even impossible, i.e., this strategy is problem dependent; in this case, one may resort to the soft enforcement of boundary conditions (1.3) instead.

2. Methods. In this section, we define FBPINNs (subsection 2.1) and extend them to multilevel FBPINNs (subsection 2.2). We also discuss the similarities and differences of FBPINNs and multilevel FBPINNs to classical DDMs (subsection 2.2.2).

2.1. Finite basis physics-informed neural networks. As discussed in section 1, a major challenge when training PINNs is that, when higher frequencies and multi-scale features are added to the solution, the accuracy of PINNs usually rapidly reduces and their computational cost rapidly increases in a super-linear fashion [37, 50].

In the FBPINN approach [37], instead of using a single neural network to represent the solution, many smaller neural networks are confined in overlapping subdomains and summed together to represent the solution. By taking this "divide and conquer"

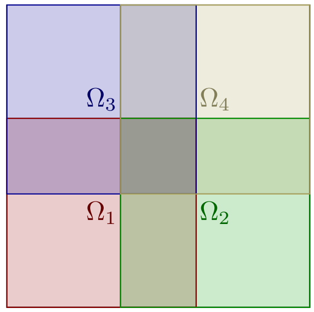

Fig. 2. *Plot of a square domain Ω decomposed into four overlapping subdomains, using a uniform rectangular decomposition.*

approach, the global PINN optimization problem is transformed into many smaller coupled local optimization problems.

Furthermore, FBPINNs ensure that the inputs to each subdomain network are normalized over their individual subdomain. When solving problems with high frequency solutions, this effectively scales each local problem from a high frequency problem to a lower frequency problem, and helps limit the effect of spectral bias; Figure 1 explains this effect further.

2.1.1. Mathematical definition. We now provide a mathematical definition of FBPINNs. First, the global solution domain Ω is decomposed into J overlapping subdomains $\{\Omega_j\}_{j=1}^{J}$; cf. Figure 2. Then, for each subdomain Ω_j, a space of network functions is defined,

$$\mathcal{V}_j = \{v_j(\mathbf{x}, \boldsymbol{\theta}_j) \mid \mathbf{x} \in \Omega_j, \boldsymbol{\theta}_j \in \Theta_j\},$$

where $v_j(\mathbf{x}, \boldsymbol{\theta}_j)$ is a neural network placed in each subdomain and $\Theta_j = \mathbb{R}^{K_j}$ is the linear space of all possible network parameters. Here, K_j is the number of local network parameters which is determined by the network architecture.

Next, each subdomain network is confined to its subdomain by multiplying each network with a window function $\omega_j(\mathbf{x})$, where $\mathrm{supp}(\omega_j) \subset \Omega_j$. Note the neural network functions used in $\mathcal{V}_j$ generally can have global support, and the window functions are used to restrict them to their individual subdomains. Furthermore, we impose that the window functions form a partition of unity, i.e.,

$$\sum_{j=1}^{J} \omega_j \equiv 1 \quad \text{on } \Omega.$$

Given the space of network functions and the window functions, we define a global space decomposition given by $\mathcal{V}$ as

$$\mathcal{V} = \sum_{j=1}^{J} \omega_j \mathcal{V}_j.$$

This space decomposition allows for decomposing any given function $u \in \mathcal{V}$ as follows

$$u = \sum_{j=1}^{J} \omega_j v_j \quad \text{or} \quad u(\mathbf{x}, \boldsymbol{\theta}) = \sum_{j=1}^{J} \omega_j v_j(\mathbf{x}, \boldsymbol{\theta}_j), \tag{2.1}$$

respectively.

FBPINNs solve the boundary value problem (1.2) by using equation (2.1) to approximate the solution, and we refer to (2.1) as the FBPINN solution. From a PINN perspective, the FBPINN solution can simply be thought of as a specific type of neural network architecture for the PINN which sums together many locally-confined networks to generate the output solution.

The same scheme for training PINNs is used to train the FBPINN. More specifically, the FBPINN solution (2.1) is substituted into the PINN loss function (1.3) and the same iterative optimization scheme is used to learn the parameters $\{\boldsymbol{\theta}_j\}_{j=1}^{J}$ of each subdomain network. FBPINNs can also be trained with hard BCs by using the same constraining operator approach described in subsection 1.3.1. In particular, substituting the FBPINN solution (2.1) into the hard-constrained loss function (1.4) yields the loss function

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(\mathcal{N}[\mathcal{C}\sum_{j=1}^{J}\omega_j v_j](\mathbf{x}_i, \boldsymbol{\theta}_j) - f(\mathbf{x}_i))^2.$$

Note, naively computing the FBPINN solution (2.1) can be very expensive as it requires a summation over all subdomain networks at each collocation point. However, this cost can be significantly reduced by noting that only subdomains which contain the respective collocation point need to be included; more details on our software implementation are provided in Appendix A.

2.2. Multilevel FBPINNs. Multilevel FBPINNs extend FBPINNs by adding multiple levels of domain decompositions to their solution ansatz. They are inspired by classical multilevel DD methods, where coarse levels are generally required for numerical scalability when using large numbers of subdomains, and multilevel approaches may significantly improve performance; see, for instance, [47, 15]. Our hypothesis is that multilevel modeling similarly improves the performance of FBPINNs. The generalization of FBPINNs to two levels was briefly discussed in [14] and we fully introduce the concept here.

A multilevel FBPINN is defined as follows. First, we define L levels of domain decompositions, where each level, l, defines an overlapping domain decomposition of Ω with $J^{(l)}$ subdomains, i.e.,

$$D^{(l)} = \left\{\Omega_j^{(l)}\right\}_{j=1}^{J^{(l)}},$$

for $j = 1, \dots, J^{(l)}$. Without loss of generality, let $J^{(1)} = 1$, that is, on the first level, we only have one subdomain $\Omega_j^{(1)} = \Omega$. Moreover, we let $J^{(1)} < J^{(2)} < \dots < J^{(l)}$, meaning that the number of subdomains increases from one to the next level.

Next, we define spaces of network functions for each level,

$$\mathcal{V}_j^{(l)} = \left\{v_j^{(l)}(\mathbf{x}, \boldsymbol{\theta}_j^{(l)}) \mid \mathbf{x} \in \Omega_j^{(l)}, \boldsymbol{\theta}_j^{(l)} \in \Theta_j^{(l)}\right\}, \quad j = 1, \dots, J^{(l)},\ l = 1, \dots, L,$$

as well as a partition of unity for each level using window functions, $\omega_j^{(l)}$, with

$$\text{supp}\big(\omega_j^{(l)}\big) \subset \Omega_j^{(l)} \quad \text{and} \quad \sum_{j=1}^{J^{(l)}} \omega_j^{(l)} \equiv 1 \quad \text{on } \Omega \quad \forall l.$$

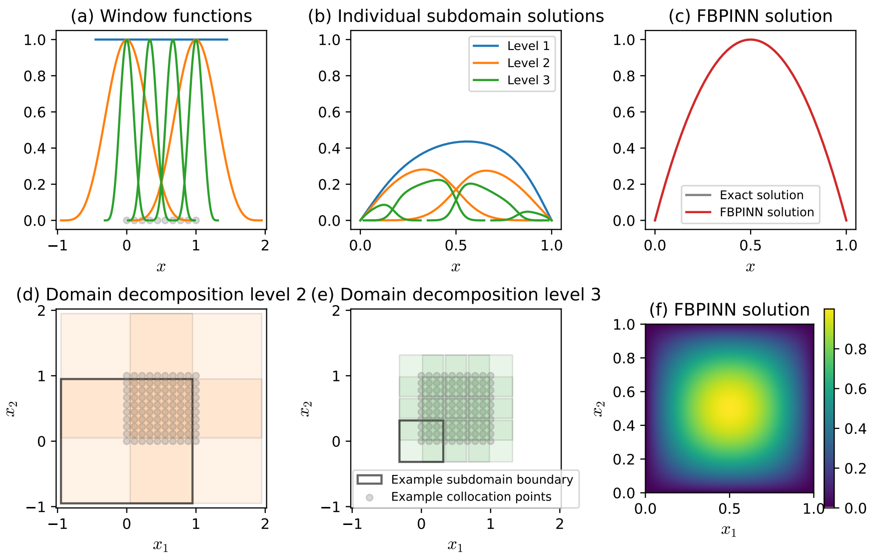

FIG. 3. *Example of a multilevel FBPINN solving Laplace's equation in one and two dimensions. For the 1D problem, the multilevel FBPINN uses $L = 3$ levels, where each level has 1, 2 and 4 subdomains respectively. The window functions, $\hat{\omega}_j^{(l)}(x)$, used for each level are shown in (a), the individual solutions learned by each subdomain network are shown in (b), and the multilevel FBPINN solution is shown in (c). For the 2D problem, the multilevel FBPINN uses $L = 3$ levels, where each level has 1×1, 2×2 and 4×4 subdomains respectively, using a uniform rectangular domain decomposition. The domain decompositions for level 2 and level 3 are plotted in (d) and (e), and the multilevel FBPINN solution is shown in (f). Note the subdomain boundaries and window functions extend past the problem domain (in this case, $[0, 1]^d$). Example collocation points used to train the multilevel FBPINN are plotted in (a), (d) and (e).*

We can then define a global space decomposition,

$$\mathcal{V} = \frac{1}{L} \sum_{l=1}^{L} \sum_{j=1}^{J^{(l)}} \omega_j^{(l)} \mathcal{V}_j^{(l)},$$

and use this space decomposition to decompose any given function $u \in \mathcal{V}$ as follows,

$$u = \frac{1}{L} \sum_{l=1}^{L} \sum_{j=1}^{J^{(l)}} \omega_j^{(l)} v_j^{(l)} \quad \text{or} \quad u(\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{L} \sum_{l=1}^{L} \sum_{j=1}^{J^{(l)}} \omega_j^{(l)} v_j^{(l)}(\mathbf{x}, \boldsymbol{\theta}_j^{(l)}). \tag{2.2}$$

We refer to (2.2) as the multilevel FBPINN solution. Note, the original FBPINN solution described in subsection 2.1 can be obtained by simply setting $L = 1$; we refer to these as one-level FBPINNs going forward.

Analogously, we can train multilevel FBPINNs by using the same training scheme as PINNs and inserting (2.2) into the PINN loss function. When using the hard-constrained PINN loss function (1.4), this yields the corresponding multilevel FBPINN loss function

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} (\mathcal{N}[\mathcal{C} \frac{1}{L} \sum_{l=1}^{L} \sum_{j=1}^{J^{(l)}} \omega_j^{(l)} v_j^{(l)}](\mathbf{x}_i, \boldsymbol{\theta}_j^{(l)}) - f(\mathbf{x}_i))^2. \tag{2.3}$$

2.2.1. Example of a multilevel FBPINN. We now show a simple example of a multilevel FBPINN to aid understanding. In particular, we use a multilevel FBPINN to solve the Laplacian boundary value problem,

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega = [0,1]^d, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

First we consider the 1D case ($d = 1$), and set $f = 8$. Then the exact solution is given by $u(x) = 4x(1-x)$.

We create an $L = 3$ level FBPINN to solve this problem, with $J^{(1)} = 1$, $J^{(2)} = 2$, and $J^{(3)} = 4$. Each level uses a uniform domain decomposition given by

$$\Omega_j^{(l)} = \begin{cases} [0.5 - \delta/2, 0.5 + \delta/2] & l = 1, \\ \left[\frac{(j-1)-\delta/2}{J^{(l)}-1}, \frac{(j-1)+\delta/2}{J^{(l)}-1}\right] & l > 1, \end{cases} \tag{2.4}$$

where δ is defined as the *overlap ratio* and is fixed at a value of $\delta = 1.9$. Note that an overlap ratio of less than 1 means that the subdomains are no longer overlapping. The subdomain window functions form a partition of unity for each level and are given by

$$\omega_j^{(l)} = \frac{\hat{\omega}_j^{(l)}}{\sum_{j=1}^{J^{(l)}} \hat{\omega}_j^{(l)}} \quad \text{where} \quad \hat{\omega}_j^{(l)}(x) = \begin{cases} 1 & l = 1 \\ [1 + \cos(\pi(x - \mu_j^{(l)})/\sigma_j^{(l)})]^2 & l > 1, \end{cases}$$

where $\mu_j^{(l)} = (j-1)/(J^{(l)}-1)$ and $\sigma_j^{(l)} = (\delta/2)/(J^{(l)}-1)$ represent the center and half-width of each subdomain respectively. Note that FBPINNs are not restricted to these particular window functions or partition of unities and any other choice could be used instead. The window functions for each level are plotted in Figure 3 (a). A FCN (1.1) with 1 hidden layer and 16 hidden units is placed in each subdomain, and the x inputs to each subdomain network are normalized to the range [-1,1] over their individual subdomains.

The multilevel FBPINN is trained using the hard-constrained loss function (2.3) with a constraining operator given by $[\mathcal{C}u](x, \boldsymbol{\theta}) = \tanh(x/\sigma)\tanh((1-x)/\sigma)u(x, \boldsymbol{\theta})$ and $\sigma = 0.2$. The loss function is minimized using the Adam optimizer with a learning rate of 1×10^{-3} and $N = 80$ uniformly-spaced collocation points across the domain.

The resulting multilevel FBPINN solution is shown in Figure 3 (c), and the individual subdomain network solutions (with the constraining operator and window function applied) are shown in Figure 3 (b). In this case, we find the FBPINN closely matches the exact solution.

Next we consider the 2D case ($d = 2$), and set $f(x_1, x_2) = 32(x_1(1-x_1) + x_2(1-x_2))$. Then the exact solution is given by $u(x_1, x_2) = 16(x_1(1-x_1)x_2(1-x_2))$.

In this case we create a $L = 3$ level FBPINN to solve this problem, using a uniform rectangular domain decomposition for each level with $J^{(1)} = 1 \times 1 = 1$, $J^{(2)} = 2 \times 2 = 4$, and $J^{(3)} = 4 \times 4 = 16$, as shown in Figure 3 (d) and (e). The size of each subdomain along each dimension is defined similar as in (2.4) using, again, an overlap ratio of $\delta = 1.9$. The subdomain window functions are given by

$$\omega_j^{(l)} = \frac{\hat{\omega}_j^{(l)}}{\sum_{j=1}^{J^{(l)}} \hat{\omega}_j^{(l)}}, \quad \text{where} \quad \hat{\omega}_j^{(l)}(\mathbf{x}) = \begin{cases} 1 & l = 1 \\ \prod_i^d [1 + \cos(\pi(x_i - \mu_{ij}^{(l)})/\sigma_{ij}^{(l)})]^2 & l > 1, \end{cases} \tag{2.5}$$

where $\mu_{ij}^{(l)}$ and $\sigma_{ij}^{(l)}$ represent the center and half-width of each subdomain along each dimension, respectively. An FCN (1.1) with 1 hidden layer and 16 hidden units is placed in each subdomain, and the $\mathbf{x}$ inputs to each subdomain network are normalized to the range [-1,1] along each dimension over their individual subdomains.

Similar to above, the multilevel FBPINN is trained using the hard-constrained loss function (2.3), using a constraining operator given by

$$[\mathcal{C}u](\mathbf{x},\boldsymbol{\theta}) = \tanh(x_1/\sigma)\tanh((1-x_1)/\sigma)\tanh(x_2/\sigma)\tanh((1-x_2)/\sigma)u(\mathbf{x},\boldsymbol{\theta}),$$

with $\sigma = 0.2$. The loss function is minimized using the Adam optimizer with a learning rate of 1×10^{-3} and $N = 80\times80 = 6{,}400$ uniformly-spaced collocation points across the domain.

The resulting multilevel FBPINN solution is shown in Figure 3 (f). Similar to the 1D case, we find the multilevel FBPINN solution closely matches the exact solution.

2.2.2. Multilevel FBPINNs versus classical multilevel DDMs. Whilst multilevel FBPINNs are inspired by classical multilevel DDMs, a number of differences and similarities exist between these approaches. We believe it is insightful to briefly discuss these below.

Most classical DDMs can be described in terms of the abstract Schwarz framework [44, 47]. Similar to FBPINNs, this framework is based on a decomposition of a global function space V into local spaces $\{V_j\}_{j=1}^J$ defined on overlapping subdomains Ω_j, where

$$V = \sum_{j=1}^{J} R_j^\top V_j. \tag{2.6}$$

Here, $R_j^\top : V_j \to V$ is an interpolation respectively prolongation operator from the local into the global space. These notions can be defined in a similar fashion at the continuous and discrete level. For the sake of simplicity, we suppose here a variational discretisation of the PDE to solve. The space decomposition (2.6) allows for decomposing any given discrete function $u \in V$ as

$$u = \sum_{j=1}^{J} R_j^\top v_j, \quad v_j \in V_j; \tag{2.7}$$

due to the overlap, this decomposition is generally not unique. Schwarz DDMs are then based on solving local overlapping problems corresponding to the local spaces $\{V_j\}_{j=1}^J$ and merging them via the prolongation operators $R_j^\top$.

Classical one-level Schwarz methods based on this framework are typically not scalable to large numbers of subdomains. In particular, since information is only transported via the overlap, their rate of convergence will deteriorate when increasing the number of subdomains [47]. In order to fix this, multilevel methods add coarser problems to the Schwarz framework to facilitate the global transfer of information; in particular, the coarsest level typically corresponds to a global problem.

We note that:

- In classical Schwarz methods, the global discretization space V is often fixed first, and then, the local spaces $\{V_j\}_{j=1}^J$ are constructed. In FBPINNs, we do the opposite; we define a local space of neural network functions on an overlapping domain decomposition $\{\Omega_j\}_{j=1}^J$ and construct the global discretization space from them.

- In classical Schwarz methods, the local functions $v_j \in V_j$ are generally not defined on the global domain Ω outside the overlapping subdomain Ω_j; the prolongation operators $R_j^\top$ extend the local functions to Ω such that $\text{supp}(R_j^\top v_j) \subset \Omega_j, \forall v_j \in V_j$. On the other hand, in FBPINNs, the local neural network functions v_j generally have global support, and the window functions ω_j are used to confine them to their subdomains. This difference steams from the fact that the local neural networks are not based on a spatial discretization but a function approximation; cf. subsection 1.3. Nonetheless, both the prolongation operators and the window functions ensure locality; cf. (2.1) and (2.7). Note that the prolongation operators in the restricted additive Schwarz (RAS) method [8] also include a partition of unity, such that they are very close to the window functions in FBPINNs.
- A key difference is how the boundary value problem is solved. Whereas in domain decomposition methods, local subdomain problems are explicitly defined and solved in a global iteration, in FBPINNs, the global loss function is minimized. Moreover, classical DDMs can exploit properties of the system to be solved. For instance, if the PDE is linear elliptic, convergence guarantees for classical DDMs can be derived; cf. [47, 15]. In FBPINNs, we always have to solve a non-convex optimization problem (1.3) or (1.4), which makes the derivation of convergence bounds difficult. Note that there are also nonlinear overlapping domain decomposition methods, for instance, additive Schwarz preconditioned inexact Newton (ASPIN) and additive Schwarz preconditioned exact Newton (ASPIN) methods [7].

3. Numerical results. In this section, we assess the performance of multilevel FBPINNs. In particular, we investigate the accuracy and computational cost of using multilevel FBPINNs to solve various differential equations, and compare them to PINNs and one-level FBPINNs.

First, in subsection 3.1, we introduce the problems studied. Then, in subsection 3.2 we introduce a notion of strong and weak scaling, inspired by classical DDMs, for assessing how the accuracy of FBPINNs and PINNs scales with computational effort and solution complexity. In subsection 3.3, we list the common implementation details used across all experiments. Finally, in subsection 3.4 we present our numerical results.

3.1. Problems studied. The following problems are used to assess the performance of multilevel FBPINNs;

3.1.1. Homogeneous Laplacian problem in two dimensions. First, we consider the 2D homogeneous Laplacian problem already presented above, namely

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega = [0,1]^2, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{3.1}$$

where

$$f(x_1, x_2) = 32(x_1(1 - x_1) + x_2(1 - x_2)).$$

In this case, the exact solution is given by

$$u(x_1, x_2) = 16(x_1(1 - x_1)x_2(1 - x_2)).$$

This problem is used to carry out simple ablation tests of the multilevel FBPINN. In particular, we assess how varying the number of levels and subdomains as well as the overlap ratio and size of the subdomain networks (architecture) affects the multilevel FBPINN performance.

3.1.2. Multi-scale Laplacian problem in two dimensions. Next, we consider a multi-scale variant of the Laplacian problem (3.1) above by using the source term

$$f(x_1, x_2) = \frac{2}{n} \sum_{i=1}^{n} \left(\omega_i \pi\right)^2 \sin(\omega_i \pi x_1) \, \sin(\omega_i \pi x_2). \tag{3.2}$$

Then, the exact solution is given by

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

In this case, multi-scale frequencies are contained in the solution, and the values of n and ω_i allow us to choose the number of components and the frequency of each component. We use this problem to assess how the performance of the multilevel FBPINN scales when more multi-scale components are added to the solution.

3.1.3. Helmholtz problem in two dimensions. Finally, we study the 2D Helmholtz problem

$$\begin{aligned} \Delta u - k^2 u &= f \quad \text{in } \Omega = [0,1]^2, \\ u &= 0 \quad \text{on } \partial\Omega, \\ f(\mathbf{x}) &= e^{-\frac{1}{2}(\|\mathbf{x}-0.5\|/\sigma)^2}, \end{aligned} \tag{3.3}$$

with a constant (scalar) wave number, k. Here, homogeneous Dirichlet boundary conditions and a Gaussian point source with a scalar width, σ, placed in the center of the domain are used. Note that, for this problem, the exact solution is not known, and instead, we compare our models to the solution obtained from FD modeling, as described in Appendix B.

In this case, the solution contains complex patterns of standing waves where the dominant frequency of the solution depends on the wave number, k. We use this problem to test the multilevel FBPINN on a more realistic problem. We first carry out some simple ablation tests by assessing how varying the number of levels, subdomains, overlap ratio and size of the subdomain networks affects the multilevel FBPINN performance. Then, we assess how the performance of the multilevel FBPINN scales when the value of k is increased.

3.2. Definition of strong and weak scaling. For both the multi-scale Laplacian and Helmholtz problems, we carry out strong and weak scaling tests. These assess how the accuracy of the multilevel FBPINN scales with computational effort and solution complexity and are inspired by the strong and weak scaling tests commonly used in classical DD. They are defined in the following way;

- *Strong scaling:* We fix the complexity of the problem and increase the model capacity. For optimal scaling, we expect the convergence rate and/or accuracy to improve at the same rate as the increase of model capacity.
- *Weak scaling:* We increase the complexity of the problem and the model capacity at the same rate. For optimal scaling, we expect the convergence rate and/or accuracy to stay approximately constant.

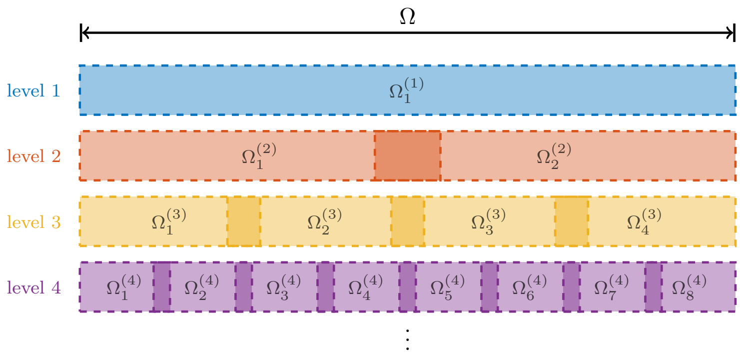

FIG. 4. *Hierarchy of levels used in the multilevel FBPINN. For all the multilevel FBPINNs tested we use an exponential level structure. This means that the number of subdomains in each level is given by $2^{d(l-1)}$, where l is the level number and d is the dimensionality of the domain. Our hypothesis is that this helps the multilevel FBPINN model solutions with frequency components that span multiple orders of magnitude.*

For all our tests, increasing the model capacity means increasing the number of levels, number of subdomains, size of the subdomain networks, and/or the number of collocation points used. The exact factors varied and their rates of increase are detailed in the relevant results sections below. Note all of the multilevel FBPINNs tested have been trained on a single GPU, and hence we only show strong and weak scaling tests with respect to model capacity and not hardware parallelization.

3.3. Common implementation details. Some of the implementation details of the multilevel FBPINNs, one-level FBPINNs, and PINNs tested are the same across all tests. These details are presented here; some are only changed for ablation studies, in which case they are described in the relevant results section below.

Level structure. Firstly, all multilevel FBPINNs use an exponentially increasing number of subdomains per level. In particular, we choose $J^{(l)} = 2^{d(l-1)}$ for $l = 1, ..., L$. This level structure is shown in Figure 4. This constraint is chosen so that the multilevel FBPINN is able to contain an exponentially large number of subdomains with a relatively small number of levels; our hypothesis is that this helps the multilevel FBPINN model solutions with frequency components that span multiple orders of magnitude.

Domain decomposition. All FBPINNs tested use a uniform rectangular domain decomposition for each level, with all multilevel FBPINNs having 2^{l-1} subdomains along each dimension. The size of each subdomain along each dimension is defined similar to (2.4), i.e., all 2D domain decompositions look similar to those shown in Figure 3 (d) and (e). Furthermore, all FBPINNs use the same subdomain window functions, given by (2.5).

Network architecture. All the FBPINNs tested use FCNs with identical architectures as their subdomain networks. The PINNs tested also use FCNs as their network architecture. For all the FBPINNs tested, the $\mathbf{x}$ inputs to each subdomain network are normalized to the range [-1,1] along each dimension over their individual subdomains. For the PINNs tested, the $\mathbf{x}$ inputs are normalized to the range [-1,1] along each dimension over the global domain.

Loss function and optimization. All FBPINNs and PINNs tested use the hard-constrained variants of their loss functions. All tests use the Adam optimizer with a

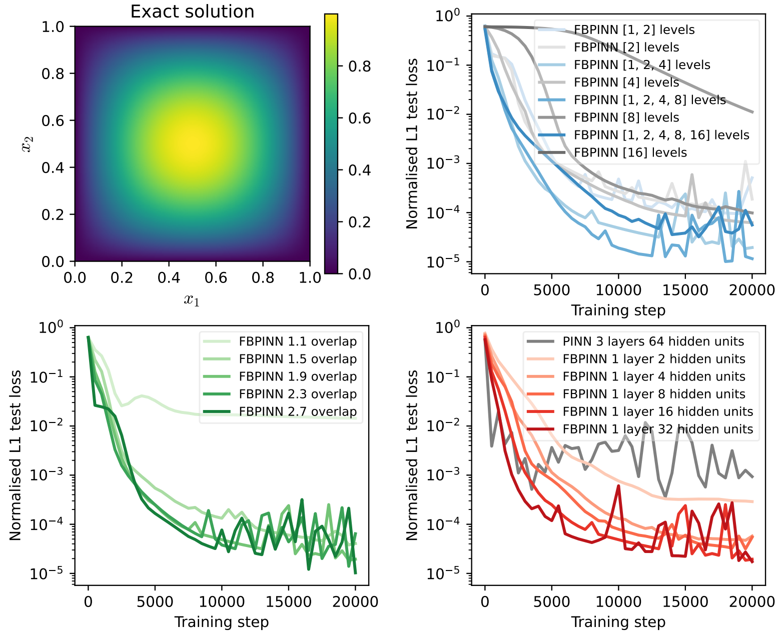

FIG. 5. *Ablation tests using the homogeneous Laplacian problem. The convergence curve of a baseline multilevel FBPINN is plotted when changing the number of levels (top right), overlap ratio (bottom left), and number of hidden units for each subdomain network (bottom right). The baseline model has $L = 3$ levels, an overlap ratio of $\delta = 1.9$, and 16 hidden units for each subdomain network. The exact solution is shown (top left). Convergence curves of two other benchmarks are shown; a PINN (bottom right), and one-level FBPINNs with varying numbers of subdomains (top right). The lists which label each model in the top right plot contain the number of subdomains along each dimension for each level in the model.*

learning rate of 1×10^{-3}. For fairness, the same constraining operator is used across all models tested on a given problem. Furthermore, exactly the same collocation points are used for training whenever multiple models are compared on a given problem. This is similarly the case for all testing points used after training. All models are evaluated using the normalized L1 test loss, given by $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M}\sum_i^M \|u(\mathbf{x}_i,\boldsymbol{\theta}) - u(\mathbf{x}_i)\|/\sigma$, where M is the number of test points and σ is the standard deviation of the set of true solutions $\{u(\mathbf{x}_i)\}_i^M$.

Software and hardware implementation. All FBPINNs and PINNs tested are implemented using a common training framework written in JAX [5]. Further details on our software implementation are given in Appendix A. All models are trained on a single NVIDIA RTX 3090 GPU.

3.4. Results. Here, we will discuss the results for the model problems described in subsection 3.1.

3.4.1. Homogeneous Laplacian problem in two dimensions. First, we carry out simple ablation tests of the multilevel FBPINN using the 2D homogeneous Laplacian problem described in subsection 3.4.1.

To carry out our ablation tests, we first train a baseline multilevel FBPINN to solve this problem, using $L = 3$ levels, an overlap ratio along each dimension of $\delta = 1.9$, and FCNs with 1 hidden layer and 16 hidden units for each subdomain network. The multilevel FBPINN is trained using the constraining operator $[\mathcal{C}u](\mathbf{x},\boldsymbol{\theta}) = \tanh(x_1/\sigma)\tanh((1-x_1)/\sigma)\tanh(x_2/\sigma)\tanh((1-x_2)/\sigma)u(\mathbf{x},\boldsymbol{\theta})$ with $\sigma = 0.2$. Here, $N = 80 \times 80 = 6{,}400$ uniformly-spaced collocation points and $M = 350 \times 350$ uniformly-spaced test points across the global domain are used to train and test the multilevel FBPINN, respectively.

Given this baseline model, we then vary different hyperparameters over a range of values and measure the change in performance. This is carried out for the number of levels ranging from $L = 2$ to 5, the overlap ratio ranging from 1.1 to 2.7, and the number of hidden units in the subdomain network ranging from 2 to 32. Our results are shown in Figure 5. We observe that the accuracy of the multilevel FBPINN does not depend significantly on the number of levels, likely because in this case the solution is very simple. However, its accuracy increases as the overlap ratio increases, likely because there is more communication between the subdomain networks, which is similar to what is expected in classical DDMs. Furthermore, its accuracy increases as the number of free parameters of the subdomain networks increases. This is expected as the capacity of the model increases. Thus, the multilevel FBPINN has similar characteristics to classical DDMs for this problem.

We carry out two other benchmark tests. First, we train a PINN with 3 hidden layers and 64 hidden units, and second, we train four one-level FBPINNs with $J^{(1)} = 2, 4, 8,$ and 16 subdomains along each dimension, respectively. All other relevant hyperparameters are kept the same as the baseline model. These results are also shown in Figure 5. In these tests, the PINN is able to solve the problem, although its final accuracy is lower than the baseline multilevel FBPINN and its convergence curve is more unstable. Furthermore, the accuracy of the one-level FBPINN reduces as more subdomains are added. This is analogous to the expected behavior of one-level classical DDMs, which is not scalable to large numbers of subdomains, and shows that coarse levels are required for scalability. It is therefore likely that the additional levels in FBPINNs serve the same purpose as in classical DDMs, i.e., they allow direct transfer of global information.

3.4.2. Multi-scale Laplacian problem in two dimensions. Next, we evaluate the strong and weak scalability of the multilevel FBPINN using the multi-scale Laplacian problem described in subsection 3.4.2.

Strong scaling test. First, we carry out a strong scaling test. Here, the problem complexity is fixed and we assess how the performance of the multilevel FBPINN changes as the capacity of the model is increased. In particular, we fix the problem complexity by choosing $n = 6$ with $\omega_i = 2^i$ for $i = 1, ..., n$ in (3.2). Thus, the solution contains 6 multi-scale components with exponentially increasing frequencies. This represents a much more challenging problem than the homogeneous problem studied above. The exact solution in this case is shown in Figure 6.

We increase the capacity of the multilevel FBPINN by increasing the number of levels, testing from $L = 2$ to 7. For each test, $(5 \times 2^{L-1}) \times (5 \times 2^{L-1})$ uniformly-spaced collocation points are used, i.e., the density of collocation points inside the subdomains in the highest level of each model is kept constant. The rest of the hyperparameters of the multilevel FBPINN are kept fixed across all tests. Namely, we use an overlap ratio of $\delta = 1.9$ and FCNs with 1 hidden layer and 16 hidden units for each subdomain network. All models are trained using the constraining operator $[\mathcal{C}u](\mathbf{x},\boldsymbol{\theta}) = \tanh(x_1/\sigma)\tanh((1-x_1)/\sigma)\tanh(x_2/\sigma)\tanh((1-x_2)/\sigma)u(\mathbf{x},\boldsymbol{\theta})$ with $\sigma = 1/\omega_n$. $M = 350 \times 350$ uniformly-spaced test points are used to test all models.

The results of this study are shown in Figure 6. We find that the accuracy

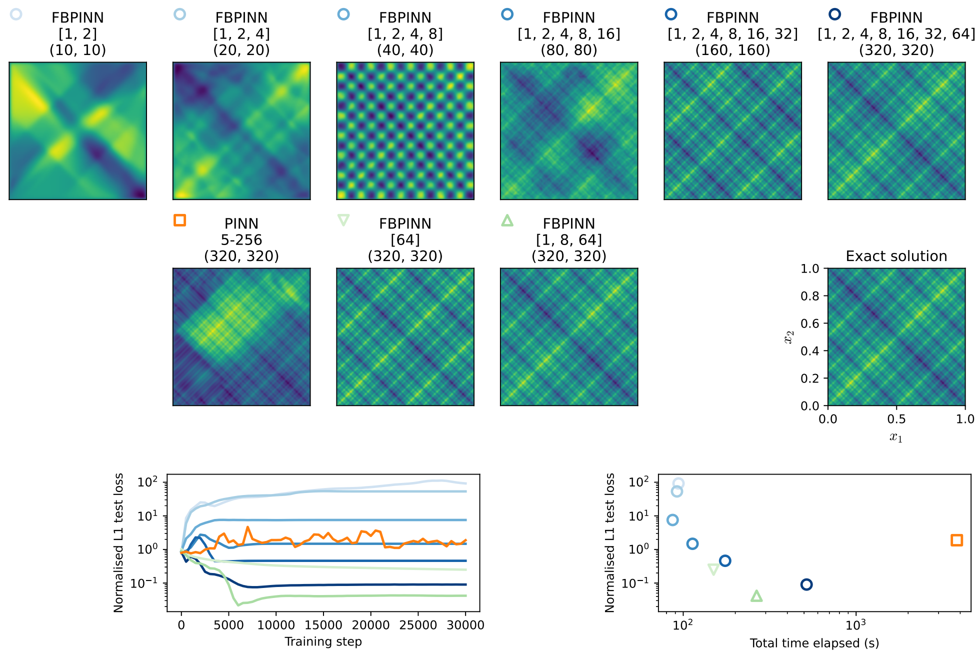

FIG. 6. *Strong scaling test using the multi-scale Laplacian problem. In this test the problem complexity is fixed and the solution estimated using multilevel FBPINNs with increasing numbers of levels and collocation points are plotted (top row). The title of each plot describes the level structure (first line) and the number of collocation points along each dimension (second line). The color-coded convergence curves and training times for each model are shown (bottom row). The exact solution is shown (middle row). Plots of the solutions and convergence curves of a PINN, one-level FBPINN and three-level FBPINN benchmark are also shown (middle and bottom row).*

of the multilevel FBPINN increases as the number of levels increases, where the $L = 2, 3, 4$ and 5 models are unable to accurately model the solution, whilst the $L = 6$ and 7 models are able to accurately model all of the frequency components. The test shows that the multilevel FBPINN is able to solve a high frequency, multi-scale problem, and exhibits strong scaling behavior somewhat similar to what is expected by classical DDMs.

Three other benchmark tests are carried out for this problem. First, we train a PINN with 5 hidden layers and 256 hidden units. Then, we train a one-level FBPINN with $J^{(1)} = 64$ subdomains along each dimension and a three-level FBPINN with $J^{(1)} = 1$, $J^{(2)} = 8$, and $J^{(3)} = 64$ subdomains along each dimension, respectively. All other relevant hyperparameters are kept the same as the baseline model above. These results are also shown in Figure 6. We find that the accuracy of the PINN is poor, and it is only able to model some of the cycles in the solution. Furthermore its convergence curve is very unstable, and its training time is an order of magnitude larger than the $L = 7$ level FBPINN tested. Its poor convergence is likely due to spectral bias and the increasing complexity of the PINN's optimization problem, as discussed in subsection 2.1 and [37]. This shows that the multilevel FBPINN strongly outperforms the PINN for this problem. The one-level FBPINN is able to model the solution, although its accuracy is less than the $L = 7$ level FBPINN. Finally, for this test the three-level FBPINN benchmark performs best, most accurately modeling the

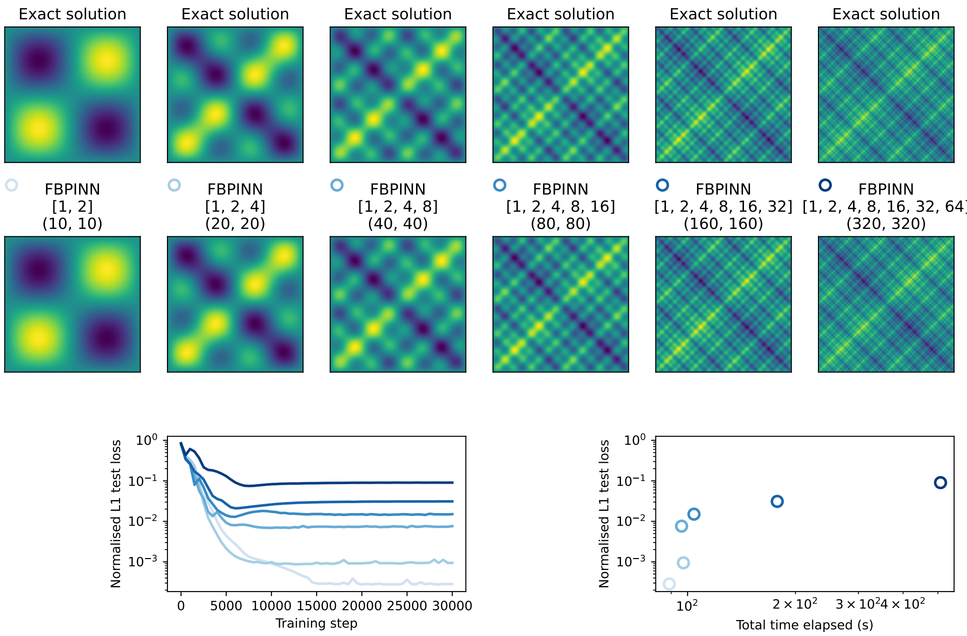

Fig. 7. *Weak scaling test using the multi-scale Laplacian problem. In this test the problem complexity is increased (in this case, the number of frequency components in the solution) (top row) and the solution estimated using multilevel FBPINNs with increasing numbers of levels and collocation points are plotted (middle row). The title of each plot describes the level structure (first line) and the number of collocation points along each dimension (second line). The color-coded convergence curves and training times for each model are shown (bottom row).*

solution. This suggests that a stronger coarsening ratio between the levels may be beneficial; this is likely to depend on the problem.

Weak scaling test. Next, we carry out a weak scaling test. Here, both the problem complexity and model capacity are scaled at the same rate, and we assess how the performance of the multilevel FBPINN changes. We increase the model capacity in exactly the same way as the strong scaling test above, i.e., the number of levels is increased from $L = 2$ to 7, where each test has $(5 \times 2^{L-1}) \times (5 \times 2^{L-1})$ uniformly-spaced collocation points. However, now the problem complexity is also scaled, such that for each test $n = L - 1$ and $\omega_i = 2^i$ for $i = 1, ..., n$. Note that the number of subdomains per level and the frequency range of the solution both grow exponentially, and the multilevel FBPINN is in alignment with the problem structure. All other hyperparameters are fixed to the same values as the strong scaling test above.

The results of this test are shown in Figure 7. We find that the multilevel FBPINNs are able to model all of the problems tested accurately, that is, modeling all of their frequency components. However, the normalized L1 accuracy of the multilevel FBPINNs does reduce slightly as the problem complex increases. Thus in this case the multilevel FBPINN exhibits near – but not perfect – weak scaling.

3.4.3. Helmholtz problem in two dimensions. Finally, we test the multilevel FBPINN using the more complex Helmholtz problem described in subsection 3.4.3. Again, we carry out ablation tests first and then carry out a weak scaling study assessing how the performance of the multilevel FBPINN changes as the wave

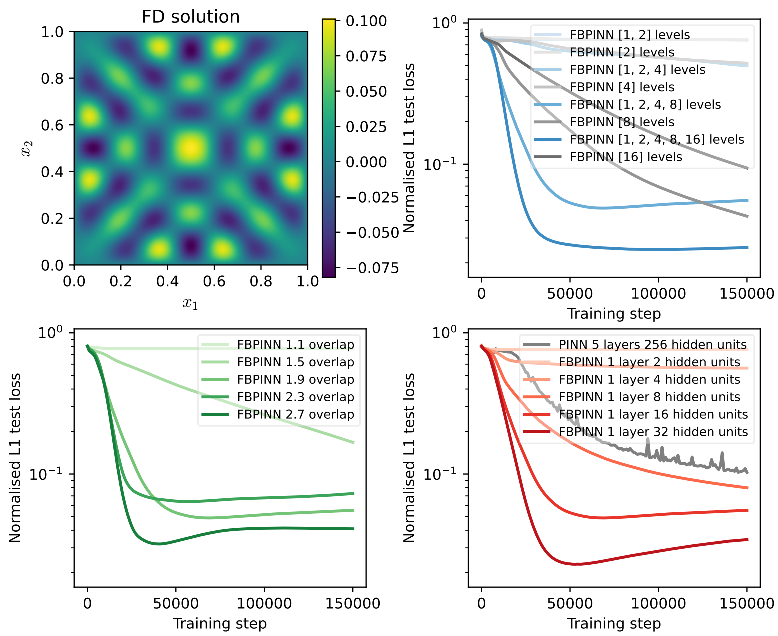

Fig. 8. *Ablation tests using the Helmholtz problem. The convergence curve of a baseline multilevel FBPINN is plotted when changing the number of levels (top right), overlap ratio (bottom left), and number of hidden units for each subdomain network (bottom right). The baseline model has $L = 4$ levels, an overlap ratio of $\delta = 1.9$, and 16 hidden units for each subdomain network. The solution obtained from FD modeling is shown (top left). Convergence curves of two other benchmarks are shown; a PINN (bottom right), and one-level FBPINNs with varying numbers of subdomains (top right). The lists which label each model in the top right plot contain the number of subdomains along each dimension for each level in the model.*

number, k, increases.

Ablation tests. For our ablation tests, we fix the problem parameters to be $k = 2^4\pi/1.6$ and $\sigma = 0.8/2^4$ in (3.3). Then, similar to subsection 3.4.1, we train a baseline multilevel FBPINN to solve this problem, using $L = 4$ levels, an overlap ratio along each dimension of $\delta = 1.9$, and FCNs with 1 hidden layer and 16 hidden units for each subdomain network. The multilevel FBPINN is trained using the constraining operator $[\mathcal{C}u](\mathbf{x}, \boldsymbol{\theta}) = \tanh(x_1/\sigma)\tanh((1-x_1)/\sigma)\tanh(x_2/\sigma)\tanh((1-x_2)/\sigma)u(\mathbf{x}, \boldsymbol{\theta})$ with $\sigma = 1/k$. We use $N = 160 \times 160 = 25{,}600$ uniformly-spaced collocation points and $M = 320 \times 320$ uniformly-spaced test points to train and test the multilevel FBPINN, respectively.

Given this baseline model, we then vary different hyperparameters over a range of values and measure the change in performance. This is carried out for the number of levels ranging from $L = 2$ to 5, the overlap ratio ranging from 1.1 to 2.7, and the number of hidden units in the subdomain network ranging from 2 to 32. Our results are shown in Figure 8. We obtain similar results to the ablation tests carried out in subsection 3.4.1 for the homogeneous Laplace problem. Namely, that the accuracy of the multilevel FBPINN improves as the overlap ratio or the number of free parameters of the subdomain networks increases. Furthermore, its accuracy improves as the number of levels increases, likely because the solution contains relatively high frequencies and multiple subdomains are needed.

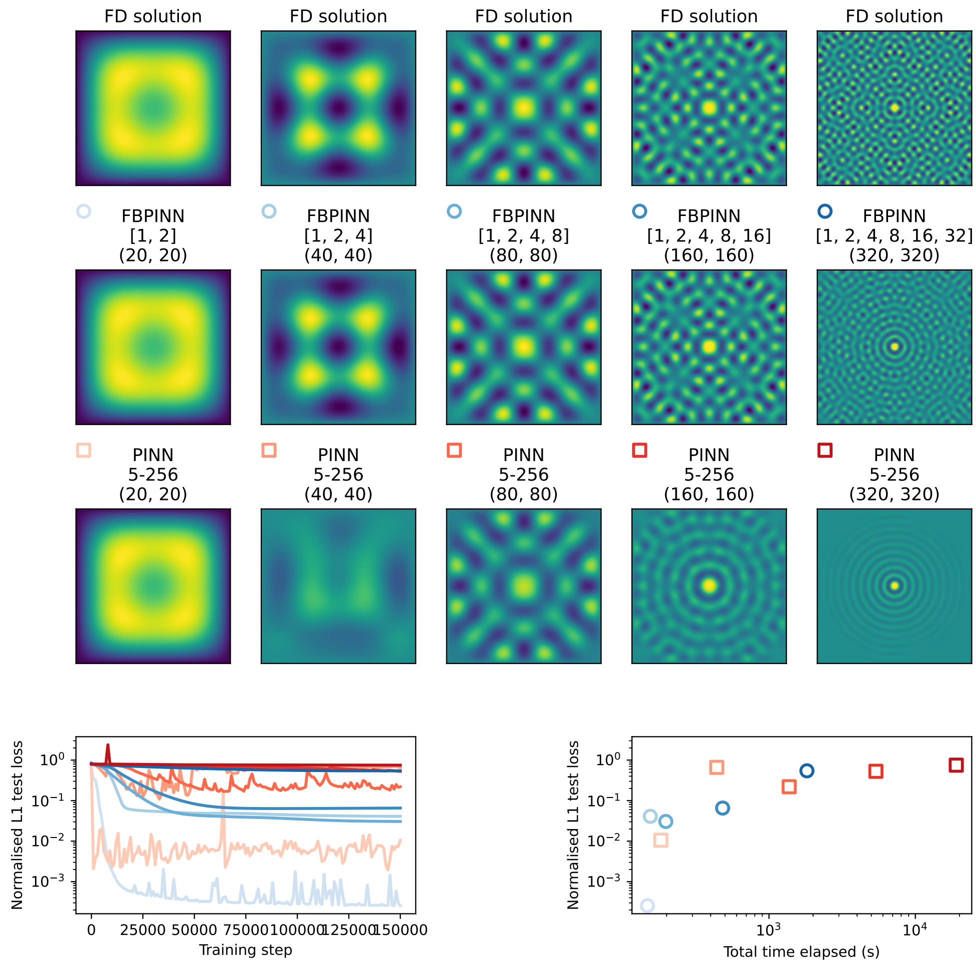

Fig. 9. *Weak scaling test using the Helmholtz problem. In this test the problem complexity is increased (in this case, the wave number) (top row) and the solution estimated using multilevel FBPINNs with increasing numbers of levels and collocation points are plotted (second row). The title of each plot describes the level structure (first line) and the number of collocation points along each dimension (second line). The color-coded convergence curves and training times for each model are shown (bottom row). A PINN benchmark using a fixed network size and increasing numbers of collocation points is also shown (third and bottom row).*

We carry out two other benchmark tests. First, we train a PINN with 5 hidden layers and 256 hidden units, and second, we train four one-level FBPINNs with $J^{(1)} = 2, 4, 8,$ and 16 subdomains along each dimension, respectively. All other relevant hyperparameters are kept the same as the baseline model. These results are also shown in Figure 8. Here, the PINN converges poorly, which again highlights the shortcomings of PINNs when solving more complex problems. Furthermore, the convergence of all the one-level FBPINNs is much slower than the multilevel FBPINN, and their final accuracy is worse. This again suggests that multiple levels are required for scalability.

Weak scaling test. We carry out a weak scaling study, where both the problem complexity and model capacity are scaled at the same rate. In a similar fashion to the weak scaling test in subsection 3.4.2, the capacity of the multilevel FBPINN is increased by increasing the number of levels, testing from $L = 2$ to 6. For each test, $(10 \times 2^{L-1}) \times (10 \times 2^{L-1})$ uniformly-spaced collocation points are used. The problem complexity for each test is increased by setting $k = 2^L\pi/1.6$ and $\sigma = 0.8/2^L$ in (3.3). All other hyperparameters are fixed to the same values as the baseline model used in the ablation tests above.

The results of this test are shown in Figure 9. We find that the multilevel FBPINN is able to accurately model all the problems tested, except for the highest wave number test. In this case, the multilevel FBPINN successfully models the dominant frequency and overall concentricity of the solution but fails to model its more complex motifs. In this case, we believe that the FBPINN is struggling to satisfy both the point source and Dirichlet boundary conditions. Without the Dirichlet boundary condition, the solution to (3.3) is that of a simple point source. For all tests, we notice that in the first few training steps this is the solution learned by the multilevel FBPINN, which is then updated to the correct solution after further training. Thus, it appears the presence of the Dirichlet boundary condition leads to an optimization problem which remains challenging. Further work is required to understand this behavior; one may be able to address this problem by using scheduling strategies to incrementally train the multilevel FBPINN, as proposed in [37].

Finally, we carry out the same weak scaling test but using a PINN instead of a multilevel FBPINN. For each test, the PINN's architecture is kept fixed at 5 layers and 256 hidden units whilst the number of collocation points and problem complexity is increased in the same way as the previous test. All other relevant hyperparameters are kept the same. The result of this study is shown in Figure 9. In this case, we find that the PINN is unable to accurately model any of the solutions, and its training time is an order of magnitude larger than the multilevel FBPINN. Thus, the multilevel FBPINN still strongly outperforms the PINN for this problem.

4. Discussion. Across all the problems studied, we find that the multilevel FBPINNs consistently outperform the one-level FBPINNs and PINNs tested. The multilevel FBPINNs are more accurate than the one-level FBPINNs when a large number of subdomains are used, suggesting that coarse levels are required for scalability by improving the global communication. Furthermore, the multilevel FBPINNs significantly outperform the PINNs across all problems tested.

We have only started to investigate multilevel FBPINNs in this work and there are a range of ways they could be extended. One interesting direction would be to investigate more complex domain decompositions and level hierarchies; in this work, we restrict ourselves to uniform rectangular decompositions with an exponentially increasing number of subdomains with respect to the levels. It is likely that irregular domain decompositions would be useful for complex problem geometries, and domain decompositions which are tailored to the structure of the solution are likely to help where the solution has a large amount of variation. Taking this further, it may be possible to learn the domain decomposition itself, for example, by learning the parameters of the window functions. This would remove the need to know about the solution structure beforehand, and be similar to, e.g., adaptive meshes in traditional methods.

Furthermore, we only consider one type of window function and partition of unity in this work, and it would be useful to assess the impact of different partitioning schemes. We only use small and identical FCNs as subdomain networks, and it would be interesting to understand the performance of other network architectures. For example, it may be useful to use different size networks for different subdomains if part of the solution is more complex and requires a higher capacity model than elsewhere. For the Helmholtz problem, we tested sinusoidal activation functions similar to [43, 28], although we did not notice a significant improvement.

Another valuable direction would be to study the theoretical convergence properties of multilevel FBPINNs. A major limitation of PINNs compared to classical DDMs is that their convergence properties are still poorly understood. In particular, whilst the multilevel FBPINN exhibits good scaling properties for the Laplacian problems studied, it remains unclear why the optimization of the high wave number Helmholtz problem is challenging; note that the convergence of classical DDMs for high wave number Helmholtz problems is also not fully understood.

A limitation of the multilevel FBPINNs tested is that, despite them being over an order of magnitude more efficient than the PINNs tested, their training times are still likely to be slower than many traditional methods, such as numerical solvers for finite difference or finite element systems. Fundamentally, this is because (FB)PINNs yield a non-convex optimization problem, which is relatively expensive compared to the linear solves which traditional methods typically rely on. FBPINNs could be extended in various ways to reduce their training cost; one direction, as suggested in [37], is to provide more inputs to the subdomain networks, such as BCs and PDE coefficients, and train across a range of these inputs so that the multilevel FBPINN learns a fast surrogate model which does not need to be retrained for each new solution. Another option is to implement multi-GPU training; in [37] a parallel FBPINN training algorithm with minimal communication between subdomains is suggested, which may allow highly scalable training. We note that classical numerical solvers can also be efficiently parallelized, for instance by using domain decomposition methods. Finally, it remains important to test the performance of FBPINNs on 3D problems; only 2D problems were studied here, and adding more dimensions is likely to significantly increase the number of collocation points and subdomains required. These limitations will be addressed in future work.

Code availability. All the code for reproducing the original FBPINN paper [37] is available here: https://github.com/benmoseley/FBPINNs. All the code for training multilevel FBPINNs and reproducing this work will be released on publication.

Appendix A. Software implementation. All FBPINNs and PINNs are implemented using a common training framework written using the JAX automatic differentiation library [5]. When training FBPINNs, computing the FBPINN solution (either (2.1) or (2.2)) naively can be very expensive. This is because evaluating the solution at each collocation point involves summing over all subdomain networks and all levels. However, the cost of this summation can be significantly reduced by exploiting that, because the output of all subdomain networks is zero outside of the corresponding subdomains, only subdomains which contain each collocation point contribute to the summation. Practically, this can be carried out by pre-computing a mapping describing which subdomains contain each collocation point before training and only evaluating the corresponding subdomain networks during training. Another important efficiency gain in our software implementation is that the outputs of each subdomain network are computed in parallel on the GPU by using JAX's `vmap` functionality. This is important as the FBPINNs tested use small subdomain networks that if evaluated sequentially would not fully utilize the GPU's parallelism.

Appendix B. Finite difference solver for Helmholtz equation. We use a finite difference (FD) solver to compute a reference solution for the Helmholtz problem studied in subsection 3.4.3. For all the problem variants studied, we discretize the Laplacian operator in (3.3) using a 5-point stencil, and we discretize the solution using a 320×320 uniformly-spaced mesh over the problem domain. This turns (3.3) into a set of linear equations, which are solved using the `scipy.sparse.linalg` [48] sparse direct solver, that is, using UMFPACK [13].

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, Mar. 2016, https://doi.org/10.48550/arXiv.1603.04467.

[2] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, *Solving inverse problems using data-driven models*, Acta Numerica, 28 (2019), pp. 1–174, https://doi.org/10.1017/s0962492919000059.

[3] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, K. Willcox, and S. Lee, *Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence*, tech. report, USDOE Office of Science (SC) (United States), feb 2019, https://doi.org/10.2172/1478744, http://www.osti.gov/servlets/purl/1478744/.

[4] R. Basri, D. Jacobs, Y. Kasten, and S. Kritchman, *The convergence rate of neural networks for learned functions of different frequencies*, in Advances in Neural Information Processing Systems, vol. 32, Neural information processing systems foundation, jun 2019, http://arxiv.org/abs/1906.00425, https://arxiv.org/abs/1906.00425.

[5] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, *JAX: composable transformations of Python+NumPy programs*, 2018, http://github.com/google/jax.

[6] S. Cai, Z. Wang, F. Fuest, Y. J. Jeon, C. Gray, and G. E. Karniadakis, *Flow over an espresso cup: Inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks*, Journal of Fluid Mechanics, 915 (2021), p. 102, https://doi.org/10.1017/jfm.2021.135, https://doi.org/10.1017/jfm.2021.135, https://arxiv.org/abs/2103.02807.

[7] X.-C. Cai and D. E. Keyes, *Nonlinearly preconditioned inexact Newton algorithms*, SIAM Journal on Scientific Computing, 24 (2002), pp. 183–200, https://doi.org/10.1137/S106482750037620X.

[8] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797, https://doi.org/10.1137/S106482759732678X.

[9] Y. Cao, Z. Fang, Y. Wu, D.-X. Zhou, and Q. Gu, *Towards Understanding the Spectral Bias of Deep Learning*, IJCAI, (2021), http://arxiv.org/abs/1912.01198, https://arxiv.org/abs/1912.01198.

[10] Y. Chen, L. Lu, G. E. Karniadakis, and L. Dal Negro, *Physics-informed neural networks for inverse problems in nano-optics and metamaterials*, Optics Express, 28 (2020), p. 11618, https://doi.org/10.1364/oe.384875, https://arxiv.org/abs/1912.01085.

[11] Z. Chen, Y. Liu, and H. Sun, *Physics-informed learning of governing equations from scarce data*, Nature Communications, 12 (2021), pp. 1–13, https://doi.org/10.1038/s41467-021-26434-1, https://www.nature.com/articles/s41467-021-26434-1, https://arxiv.org/abs/2005.03448.

[12] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, *Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next*, Journal of Scientific Computing, 92 (2022), pp. 1–62, https://doi.org/10.1007/s10915-022-01939-z, https://link.springer.com/article/10.1007/s10915-022-01939-z, https://arxiv.org/abs/2201.05624.

[13] T. A. Davis, *Umfpack-an unsymmetric-pattern multifrontal method with a column pre-ordering*

strategy, ACM Trans. Math. Software, 30 (2004), pp. 196–199.

[14] V. Dolean, A. Heinlein, S. Mishra, and B. Moseley, *Finite basis physics-informed neural networks as a schwarz domain decomposition method*, 2022, https://doi.org/10.48550/ARXIV.2211.05560.

[15] V. Dolean, P. Jolivet, and F. Nataf, *An introduction to domain decomposition methods*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2015, http://dx.doi.org/10.1137/1.9781611974065.ch1. Algorithms, theory, and parallel implementation.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[17] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber, *Machine learning in adaptive domain decomposition methods - predicting the geometric location of constraints*, SIAM Journal on Scientific Computing, 41 (2019), pp. A3887–A3912.

[18] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber, *Combining machine learning and domain decomposition methods for the solution of partial differential equations – a review*, GAMM-Mitteilungen, 44 (2021), p. e202100001.

[19] Z. Hu, A. D. Jagtap, G. E. Karniadakis, and K. Kawaguchi, *Augmented physics-informed neural networks (apinns): A gating network-based soft domain decomposition methodology*, 2022, https://arxiv.org/abs/2211.08939.

[20] 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 (2020), pp. 2002–2041, https://doi.org/10.4208/CICP.OA-2020-0164.

[21] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, *NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations*, Journal of Computational Physics, 426 (2021), p. 109951, https://doi.org/10.1016/j.jcp.2020.109951, https://arxiv.org/abs/2003.06496.

[22] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, *Physics-informed machine learning*, Nature Reviews Physics, (2021), pp. 1–19, https://doi.org/10.1038/s42254-021-00314-5, www.nature.com/natrevphys.

[23] H. J. KELLEY, *Gradient Theory of Optimal Flight Paths*, ARS Journal, 30 (1960), pp. 947–954, https://doi.org/10.2514/8.5282.

[24] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, Jan. 2017, https://doi.org/10.48550/arXiv.1412.6980. arXiv:1412.6980 [cs].

[25] 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 (1998), pp. 987–1000, https://doi.org/10.1109/72.712178, https://arxiv.org/abs/9705023.

[26] C. Leake and D. Mortari, *Deep Theory of Functional Connections: A New Method for Estimating the Solutions of Partial Differential Equations*, Machine Learning and Knowledge Extraction, 2 (2020), pp. 37–55, https://doi.org/10.3390/make2010004, https://europepmc.org/articles/PMC7259480https://europepmc.org/article/pmc/pmc7259480.

[27] K. Li, K. Tang, T. Wu, and Q. Liao, *D3m: A deep domain decomposition method for partial differential equations*, IEEE Access, 8 (2019), pp. 5283–5294.

[28] W. Li, Z. Wang, T. Cui, Y. X. null, and X. Xiang, *Deep Domain Decomposition Methods: Helmholtz Equation*, Advances in Applied Mathematics and Mechanics, 15 (2023), pp. 118–138, https://doi.org/10.4208/aamm.OA-2021-0305.

[29] W. Li, X. Xiang, and Y. Xu, *Deep domain decomposition method: Elliptic problems*, in Mathematical and Scientific Machine Learning, PMLR, 2020, pp. 269–286.

[30] P.-L. Lions, *On the Schwarz alternating method. I*, in First International Symposium on Domain Decomposition Methods for Partial Differential Equations (Paris, 1987), SIAM, Philadelphia, PA, 1988, pp. 1–42.

[31] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Mathematical Programming, 45 (1989), pp. 503–528, https://doi.org/10.1007/BF01589116.

[32] V. Mercier, S. Gratton, and P. Boudier, *A coarse space acceleration of deep-ddm*, 2021, https://doi.org/10.48550/ARXIV.2112.03732.

[33] S. Mishra and R. Molinaro, *Physics informed neural networks for simulating radiative transfer*, J. Quant. Spectroscopy and Radiative Transfer, 270 (2021).

[34] S. Mishra and R. Molinaro, *Estimates on the generalization error of physics-informed neural networks for approximating PDEs*, IMA Journal of Numerical Analysis, 00 (2022), pp. 1–43, https://doi.org/10.1093/imanum/drab093, https://academic.oup.com/imajna/advance-article/doi/10.1093/imanum/drab093/6503953, https://arxiv.org/abs/2006.16144.

[35] B. Moseley, *Physics-informed machine learning: from concepts to real-world applications*,

PhD thesis, University of Oxford, 2022, https://doi.org/10.13039/501100000266.

[36] B. Moseley, A. Markham, and T. Nissen-Meyer, *Solving the wave equation with physics-informed deep learning*, arXiv, (2020), http://arxiv.org/abs/2006.11894, https://arxiv.org/abs/2006.11894.

[37] B. Moseley, A. Markham, and T. Nissen-Meyer, *Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations*, arXiv, (2021), https://arxiv.org/abs/2107.07871v1http://arxiv.org/abs/2107.07871, https://arxiv.org/abs/2107.07871.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, in Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019, https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html (accessed 2023-04-05).

[39] N. Rahaman, A. Baratin, D. Arpit, F. Draxlcr, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville, *On the spectral bias of neural networks*, in 36th International Conference on Machine Learning, ICML 2019, vol. 2019-June, International Machine Learning Society (IMLS), jun 2019, pp. 9230–9239, http://arxiv.org/abs/1806.08734, https://arxiv.org/abs/1806.08734.

[40] 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 (2019), pp. 686–707, https://doi.org/10.1016/j.jcp.2018.10.045, https://doi.org/10.1016/j.jcp.2018.10.045.

[41] M. Raissi, A. Yazdani, and G. E. Karniadakis, *Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations*, Science, 367 (2020), pp. 1026–1030, https://doi.org/10.1126/science.aaw4741.

[42] Y. Shin, J. Darbon, and G. E. Karniadakis, *On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs*, Communications in Computational Physics, 28 (2020), pp. 2042–2074, https://doi.org/10.4208/cicp.oa-2020-0193, http://arxiv.org/abs/2004.01806, https://arxiv.org/abs/2004.01806.

[43] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, *Implicit Neural Representations with Periodic Activation Functions*, NeurIPS 2020, (2020), http://arxiv.org/abs/2006.09661, https://arxiv.org/abs/2006.09661.

[44] B. F. Smith, P. E. Bjø rstad, and W. D. Gropp, *Domain decomposition*, Cambridge University Press, Cambridge, 1996.

[45] P. Stiller, F. Bethke, M. Böhme, R. Pausch, S. Torge, A. Debus, J. Vorberger, M. Bussmann, and N. Hoffmann, *Large-Scale Neural Solvers for Partial Differential Equations*, in Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI, J. Nichols, B. Verastegui, A. B. Maccabe, O. Hernandez, S. Parete-Koon, and T. Ahearn, eds., Cham, 2020, Springer International Publishing, pp. 20–34.

[46] L. Sun, H. Gao, S. Pan, and J. X. Wang, *Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data*, Computer Methods in Applied Mechanics and Engineering, 361 (2020), https://doi.org/10.1016/j.cma.2019.112732, http://arxiv.org/abs/1906.02382http://dx.doi.org/10.1016/j.cma.2019.112732, https://arxiv.org/abs/1906.02382.

[47] A. Toselli and O. Widlund, *Domain decomposition methods—algorithms and theory*, vol. 34 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2005, https://doi.org/10.1007/b137868.

[48] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods, 17 (2020), pp. 261–272, https://doi.org/10.1038/s41592-019-0686-2.

[49] S. Wang, H. Wang, and P. Perdikaris, *Learning the solution operator of parametric partial differential equations with physics-informed DeepONets*, Science Advances, 7 (2021), pp. 8605–8634, https://doi.org/10.1126/sciadv.abi8605, https://www.science.org/doi/full/10.1126/sciadv.abi8605, https://arxiv.org/abs/2103.10974.

[50] S. Wang, H. Wang, and P. Perdikaris, *On the eigenvector bias of Fourier feature net-*

works: From regression to solving multi-scale PDEs with physics-informed neural networks, Computer Methods in Applied Mechanics and Engineering, 384 (2021), p. 113938, https://doi.org/10.1016/j.cma.2021.113938, https://arxiv.org/abs/2012.10047.

[51] S. Wang, X. Yu, and P. Perdikaris, *When and why PINNs fail to train: A neural tangent kernel perspective*, Journal of Computational Physics, 449 (2022), p. 110768, https://doi.org/10.1016/j.jcp.2021.110768, https://arxiv.org/abs/2007.14527.

[52] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, *Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems*, ACM Computing Surveys, 55 (2022), https://doi.org/10.1145/3514228, https://dl.acm.org/doi/10.1145/3514228, https://arxiv.org/abs/2003.04919.

[53] Z. Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma, *Frequency principle: Fourier analysis sheds light on deep neural networks*, Communications in Computational Physics, 28 (2020), pp. 1746–1767, https://doi.org/10.4208/CICP.OA-2020-0085, https://arxiv.org/abs/1901.06523.

[54] L. Yang, X. Meng, and G. E. Karniadakis, *B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data*, Journal of Computational Physics, 425 (2021), p. 109913, https://doi.org/10.1016/j.jcp.2020.109913.

[55] Y. Zhu, N. Zabaras, P. S. Koutsourelakis, and P. Perdikaris, *Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data*, Journal of Computational Physics, 394 (2019), pp. 56–81, https://doi.org/10.1016/j.jcp.2019.05.024, http://arxiv.org/abs/1901.06314http://dx.doi.org/10.1016/j.jcp.2019.05.024, https://arxiv.org/abs/1901.06314.