DOMAIN DECOMPOSITION-BASED COUPLING OF PHYSICS-INFORMED NEURAL NETWORKS (PINNS) VIA THE SCHWARZ ALTERNATING METHOD

WILLIAM SNYDER*, IRINA TEZAUR†, AND CHRISTOPHER WENTLAND‡

Abstract. Physics-informed neural networks (PINNs) are appealing data-driven tools for solving nonlinear partial differential equations (PDEs). Unlike traditional Neural Networks (NNs), which train on solution data, a PINN incorporates a PDE's residual into its loss function and trains to minimize the said residual at a set of collocation points on the solution domain. This paper explores the use of the Schwarz alternating method as a means to couple PINNs with each other and with conventional numerical models (i.e., full order models, or FOMs, obtained via the finite element, finite difference or finite volume methods) following a decomposition of the physical domain. It is well-known that training a PINN can be difficult when the PDE has steep gradients, such as within the boundary layer of an advection-dominated problem. We investigate herein the use of domain decomposition and the Schwarz alterating method as a means to accelerate the PINN training phase. As a numerical example, we consider the one-dimensional (1D) steady state advection-diffusion equation in the advection-dominated (high Péclet) regime.

1. Introduction. In recent years, Physics-Informed Neural Networks (PINNs) [17] have emerged as a scientific machine learning (ML) technique advertised to solve partial differential equations (PDEs) without requiring any training data or an underlying mesh. This is accomplished by constructing and minimizing a loss function that includes the residual of the underlying PDE, sampled at a finite number of collocation points within the physical domain on which the PDE is posed. While the aforementioned properties of PINNs make them appealing within the modeling and simulation community, these models unfortunately suffer from several deficiencies. First, PINNs are notoriously difficult to train for problems having a slowly-decaying Kolmogorov n-width, as discussed in [13] and the references therein. This class of problems includes advection-dominanted flow problems in which the solution exhibits sharp boundary layers and/or shocks. Second, since PINNs are trained for a given geometry with a given set boundary conditions, they require retraining when applied to different geometries and /or different boundary data [21].

In this paper, we present and evaluate a promising technique that can mitigate both difficulties described above: the Schwarz alternating method [18] for domain decomposition-based coupling. The Schwarz alternating method is based on the simple idea that if the solution to a PDE is known in two or more regularly-shaped subdomains comprising a more complex domain, these local solutions can be used to iteratively build a solution on the more complex domain, with information propagating between subdomains through boundary conditions imposed on the subdomain boundaries. In several of our recent works [15, 16], the overlapping version of the Schwarz alternating method was pioneered as a mechanism for performing concurrent coupling of subdomains discretized in space by disparate meshes and in time by different time-integration schemes with different time-steps. Here, attention was restricted to the coupling of high-fidelity full order models (FOMs) in quasistatic and dynamic solid mechanics. Our recent work [1] extended the approach to the coupling of projection-based reduced order models (ROMs) with each other and with FOMs following either an overlapping or a non-overlapping domain decomposition of the underlying geometry. We additionally demonstrated in [8, 7] that it is also possible to transform the non-overlapping Schwarz alternating method into a novel contact enforcement algorithm which exhibits remarkable energy conservation properties.

The present work explores the use of the Schwarz alternating method for coupling PINNs

*Virginia Polytechnic Institute, swilli9@vt.edu
†Sandia National Laboratories, ikalash@sandia.gov
‡Sandia National Laboratories, crwentl@sandia.gov

with each other and with FOMs, following an overlapping domain decomposition of the underlying physical domain. For a detailed literature overview on existing methods which combine ML and domain decomposition methods, the reader is referred to [6]. While it is possible to couple pre-trained subdomain-local PINNs or NNs, as in [21], our initial study focuses primarily on evaluating the Schwarz alternating method as a means to facilitate PINN training offline. The proposed approach is similar to the deep domain decomposition method (D3M) and the deep domain decomposition (DeepDDM) methods, proposed recently in [10] and [11], respectively. Unlike the D3M method, which is based on a deep Ritz method (DRM) PINN formulation in which the loss function is derived from the weak variational form of the governing PDEs [3, 19], our loss function incorporates the PDE residual in strong form. Unlike the DeepDDM method, which imposes Schwarz and other boundary conditions weakly through a boundary loss contribution to the loss function being minimized, we consider both strong and weak formulations of the Dirichlet boundary conditions (DBCs) within our algorithm. For the former strong DBC enforcement, we borrow ideas from the Finite Basis PINN (FBPINN) literature [14, 2]. While the authors of [10, 11] focus their development and demonstrations on the Poisson equation, our work considers a much more difficult boundary value problem (BVP), namely the advection-diffusion equation in the advection-dominated (high Péclet) regime. Finally, to the best of our knowledge, ours is the first coupling formulation that enabling the creation of "hybrid" models through the coupling of PINNs with conventional numerical models constructed using the finite element, finite difference or finite volume method, among other discretization approaches.

The remainder of this paper is organized as follows. Section 2 defines the model problem considered herein, a one-dimensional (1D) advection-diffusion equation on the unit interval. Section 3 overviews the overlapping version of the Schwarz alternating method for domain decomposition-based coupling in the specific context of our advection-diffusion model problem. Section 4 provides an overview of PINNs and several PINN variants, namely PINNs in which DBCs are enforced strongly, rather than weakly, and PINNs in which a data loss term is incorporated into the loss function being minimized. In Section 5, we extend the general Schwarz formulation described in Section 3 to the specific case of PINN-PINN and PINN-FOM coupling. The proposed method is evaluated as a mechanism for facilitating PINN training via domain decomposition-based coupling on our model advection-diffusion problem in Section 6. Perspectives on employing the Schwarz alternating method to perform online coupling of pre-trained PINNs are provided in Section 7. Finally,we provide some conclusions and directions for future work in Section 8.

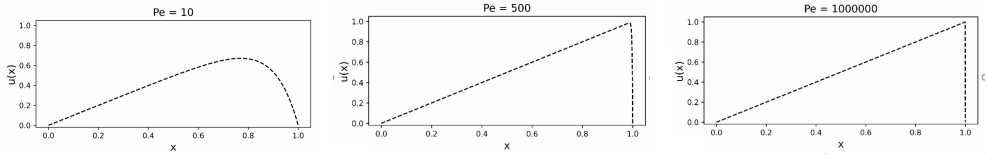
2. Model problem. As a numerical example to test our coupling method and PINNs, we choose the 1D steady-state advection-diffusion equation defined in an open bounded domain $\Omega = (0, 1)$ with boundary $\partial\Omega = \{0, 1\}$:

$$-\nu \frac{\partial^2 u}{\partial x^2} + \frac{\partial u}{\partial x} - 1 = 0 \quad \text{in } \Omega = (0, 1), \tag{2.1}$$

with boundary conditions

$$u(0) = u(1) = 0. \tag{2.2}$$

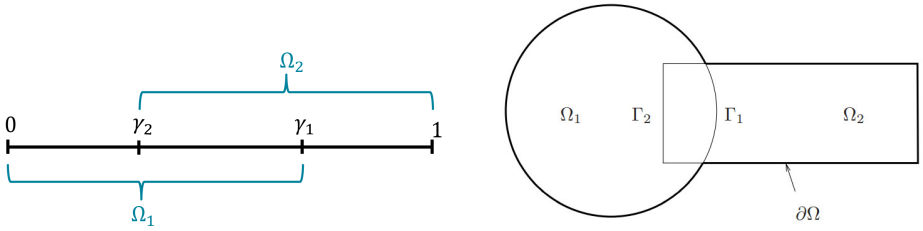
From this point forward, we will refer to the DBCs in (2.2) as the "system DBCs" or "system boundary conditions". In (2.1), ν is the diffusion coefficient, which defines the Péclet number of the problem, given by $Pe := \frac{1}{\nu}$. Our focus here is primarily on advection-dominated regimes for this problem, i.e., $10 \leq Pe \leq 10^6$. Figure 2.1 shows a plot of the solution to (2.1)–(2.2) as a function of the Péclet number. The reader can observe that the solution develops a sharp gradient near the right boundary of the domain, whose steepness is a function of Pe.



(a) $Pe = 10$ (b) $Pe = 500$ (c) $Pe = 1.0 \times 10^6$

Fig. 2.1: Solutions to our model advection-diffusion problem for different Péclet numbers Pe.

3. The Schwarz alternating method for domain decomposition-based coupling. The Schwarz alternating method is the oldest known domain decomposition method, first proposed in 1870 by H. Schwarz [18]. The method is based on a simple idea, namely that the solution to a PDE on a complex domain Ω can be obtained via an iterative procedure on subdomains comprising Ω, with information propagating via transmission (or boundary) conditions imposed on subdomain boundaries. While the Schwarz alternating method can be formulated for both overlapping [15, 16, 12] and non-overlapping [1, 8, 7] domain decompositions (DDs), herein we restrict our attention to the former (overlapping) variant of the method.



(a) 1D (b) 2D

Fig. 3.1: Example overlapping domain decompositions for Schwarz alternating method.

Consider without loss of generality an overlapping DD of a geometry Ω into two subdomains, as shown in Figures 3.1(a) and (b) in one and two spatial dimensions, respectively. We present the basic overlapping Schwarz algorithm on our targeted model problem (2.1) with boundary conditions (2.2). Suppose the governing domain for our model problem is partitioned into two overlapping domains: $\Omega = \Omega_1 \cup \Omega_2$, with $\Omega_1 = [0, \gamma_1]$ and $\Omega_2 = [\gamma_2, 1]$, with $0 < \gamma_1 < \gamma_2 < 1$, so that $\Omega_1 \cap \Omega_2 \neq \emptyset$, as shown in Figure 3.1(a). (2.1)–(2.2) can solved using the Schwarz alternating method by performing the following iteration:

$$\begin{cases} -\nu \dfrac{\partial^2 u_1^{(n+1)}}{\partial x^2} + \dfrac{\partial u_1^{(n+1)}}{\partial x} = 1, \text{ in } \Omega_1, \\ \qquad\qquad u_1^{(n+1)} = 0, \text{ at } x = 0, \\ \qquad\qquad u_1^{(n+1)} = u_2^{(n)}, \text{ at } x = \gamma_1, \end{cases} \quad \begin{cases} -\nu \dfrac{\partial^2 u_2^{(n+1)}}{\partial x^2} + \dfrac{\partial u_2^{(n+1)}}{\partial x} = 1, \text{ in } \Omega_2, \\ \qquad\qquad u_2^{(n+1)} = 0, \text{ at } x = 1, \\ \qquad\qquad u_2^{(n+1)} = u_1^{(n+1)}, \text{ at } x = \gamma_2, \end{cases} \tag{3.1}$$

for Schwarz iterations $n = 0, 1, 2, \ldots$. In (3.1), u_i denotes the solution in Ω_i for $i = 1, 2$. The reader can observe that the transmission boundary conditions on the Schwarz boundaries $x = \gamma_1$ and $x = \gamma_2$ are of the Dirichlet type. The iteration (3.1) continues until convergence is

reached. Herein, convergence of the method is declared when $||u_i^{(n+1)} - u_i^{(n)}||_2 / ||u_i^{(n)}|| < \delta$, for $i = 1, 2$ and for some specified Schwarz tolerance δ. It can be shown [12, 15] that the Schwarz iteration procedure (3.1) converges to the solution of (2.1)–(2.2) provided the overlap is non-empty ($\Omega_1 \cap \Omega_2 \neq \emptyset$). It is straightforward to extend the Schwarz iteration procedure (3.1) to an arbitrary number of overlapping subdomains. The present work builds on recent extensions of the Schwarz alternating method to concurrent multi-scale coupling in quasistatic [15] and dynamic [16] solid mechanics, and to the coupling of projection-based reduced order models (ROMs) with each other and with high-fidelity full order models (FOMs) [1].

Remark 1. The Schwarz iteration procedure described above in (3.1) is often referred to as the multiplicative Schwarz algorithm [4]. In this algorithm, the Schwarz iteration in subdomain Ω_2 at time-step $n+1$ depends on the Schwarz solution in subdomain Ω_2 at time-step $n+1$. The Schwarz iteration can be modified to achieve what is commonly referred to as additive Schwarz [4] by applying boundary conditions from the n^{th} Schwarz iteration procedure in Ω_1 to the $(n+1)^{st}$ Ω_2 sub-problem. If this change is made, the Schwarz iteration sequence within can be parallelized over the number of subdomains, as done in [10, 11]. While we do not consider the additive variant of the Schwarz method in the present work, preliminary results have suggested that the method does not reduce solution accuracy and can achieve speed-ups if parallelized appropriately.

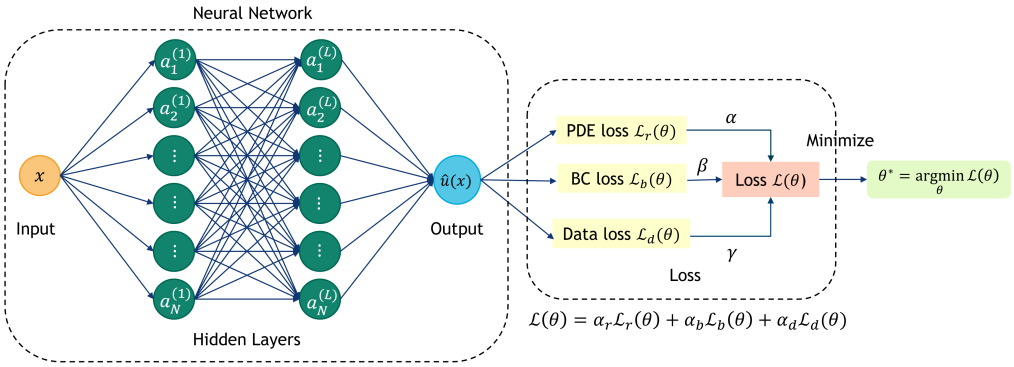


Fig. 4.1: Illustration of our PINN architecture for $L = 2$ layers each having $N = 6$ nodes per layer.

4. Physics-Informed Neural Networks (PINNs). Physics-Informed Neural Networks (PINNs), introduced in the seminal paper [17], are constructed via a least-squares collocation method applied to the governing PDEs, the solution of which is approximated by a neural network function, denoted by $NN(\cdot;\cdot)$. Let x denote the input and $\hat{u}$ denote the output of a typical feed-forward fully-connected NN[1] with L hidden layers and N neurons in each layer, as shown in Figure 4.1 for the specific case of $L = 2$ and $N = 6$. The output of the network is computed as: TODO: Will or Chris should check that the following is correct.

$$NN(\cdot;\cdot) = \sigma\left(\sum_{i=1}^{N} w_{1,i}^{(L)} a_i^{(L)} + b^{(L)}\right), \tag{4.1}$$

[1]From this point further, we will use the term NN and PINN interchangeably, since our most general PINN formulation (4.6) includes a data loss term. Please see Remark 2 for a comment on this.

where

$$a_n^{(j)} = \sigma\left(\sum_{i=1}^{N} w_{n,i}^{(j-1)} a_i^{(j-1)} + b^{(j-1)}\right) \tag{4.2}$$

for $j = 2, ..., L$ and $n = 1, .., N$, and

$$a_n^{(1)} = \sigma\left(w_{n,1}^{(0)} x + b^{(0)}\right) \tag{4.3}$$

for $n = 1, .., N$. In (4.1)–(4.3), $\sigma(\cdot)$ denotes a nonlinear function known as the activation function, $a_i^{(j)}$ is referred to as the activation of neuron i in layer j, and the constants $w_{i,j}^{(l)}$ and $b^{(l)}$ denote the weights and biases of the NN, respectively. Letting $\theta := \left(w_{1,1}^{(0)}, ..., w_{1,N}^{(L)}, b^{(0)}, ..., b^{(L)}\right)^T$, the NN is obtained by minimizing a pre-defined loss function $\mathcal{L}(\theta)$:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta). \tag{4.4}$$

A PINN is differentiated from a typical NN by the definition of the loss function $\mathcal{L}(\theta)$. Whereas a NN loss function measures the difference between the NN outputs and a corresponding set of solution data taken from the ground truth, a PINN does not require any any such solution data. Instead, a PINN uses the residual of the equation it is attempting to solve as the loss function (in this case, equation (2.1)). This is achieved by first taking as inputs a set of collocation points $x \in \Omega$. In its most basic form, the loss function defining the PINN minimization problem is given by the residual loss, that is, by

$$\mathcal{L}(\theta) = \mathcal{L}_r(\theta) := \frac{1}{n}\sum_{i=1}^{n}\left(-\nu\nabla_x^2\hat{u}(x_i,\theta) + \beta\nabla_x\hat{u}(x_i,\theta) - 1\right)^2, \tag{4.5}$$

where the $x_i \in \Omega$ for $i = 1, ..., n$ are the collocation points used to evaluate the residual loss and $NN(x;\theta) = \hat{u}(x;\theta) \approx u(x)$ represents the PINN approximation of the solution as a function of x and training parameters θ. Once the loss function (4.5) is defined along with the PINN architecture (Figure 4.1), a gradient-based optimization algorithm is used to solve (4.4), with derivatives determined via automatic differentiation.

In the following subsections, we describe several PINN variations that allow for the incorporation of boundary conditions as well as available snapshot data. In this more general formulation, the loss being minimized takes the form

$$\mathcal{L}(\theta) = \alpha_r\mathcal{L}_r(\theta) + \alpha_b\mathcal{L}_b(\theta) + \alpha_d\mathcal{L}_d(\theta), \tag{4.6}$$

for some relaxation parameters $\alpha_r, \alpha_b, \alpha_d \in \mathbb{R}$, often normalized to sum to be between 0 and 1. In (4.6), $\mathcal{L}_b(\theta)$ and $\mathcal{L}_d(\theta)$ are the boundary and data losses, respectively. These terms are discussed in more detail below, in Sections 4.1 and 4.3.

Remark 2. While, in the original PINN paper by Raissi *et al.* [17], PINNs were presented as completely data-free, i.e., $\alpha_d = 0$ in (4.6), subsequent PINN formulations allowed the addition of the data-loss term $\mathcal{L}_d(\theta)$ for cases where some training data (from high-fidelity simulations or experiments) are available. For more information, the interested reader is referred to [9, 5] and the references therein.

4.1. Weak Dirichlet BCs (WDBCs). The traditional way to impose boundary conditions (BCs) in PINNs is through a weak formulation. Toward this effect, the loss function being minimized (4.6) takes the form

$$\mathcal{L}(\theta) = \alpha\mathcal{L}_r(\theta) + (1-\alpha)\mathcal{L}_b(\theta), \tag{4.7}$$

where $\alpha \in (0,1)$, and

$$\mathcal{L}_b(\theta) := \frac{1}{b}\sum_{i=1}^{b} \left(\hat{u}(x_i,\theta) - u(x_i)\right)^2. \tag{4.8}$$

In (4.8), the points $x_i \in \partial\Omega$ for $i = 1, ..., b$ are collocation points on the boundary of Ω at which Dirichlet boundary conditions are prescribed. In the 1D domain depicted in Figure 3.1(a), $x_i \in \{0,1\}$ for the domain Ω, whereas $x_1 \in \{0,\gamma_1\}$ and $x_2 \in \{\gamma_2,1\}$ for the subdomains Ω_1 and Ω_2, respectively.

4.2. Strong Dirichlet BCs (SDBCs). While the majority of PINNs in the literature employ a weak enforcement of the boundary conditions via a boundary loss term (4.8), as described in Section 4.1, this approach has some disadvantages. First, it is unclear *a priori* how to select the relaxation parameters α_r and α_b in (4.6); often this is done by trial and error, and the values require retuning when the problem setup is modified. Second, since the BCs in (4.6) are imposed weakly, the learned solution may be inconsistent with the underlying boundary value problem (BVP) (2.1)–(2.2). Moreover, some recent work has demonstrated that the PINN optimization problem (4.6) may be stiff to solve, leading to a lack of convergence as a result of the residual and boundary losses competing with each other in the loss function [22, 20].

In recent years, several approaches for imposing BCs strongly within a PINN or NN have been developed. In the FBPINN approach [14, 2], instead of defining a NN to directly approximate the solution as discussed above ($NN(x;\theta) \approx u(x)$), an ansatz of the form

$$\hat{u}(x;\theta) = g(x) + \psi(x)NN(x;\theta) \approx u(x), \tag{4.9}$$

where the functions $g(x)$ and $\psi(x)$ are derived such that $\hat{u}(x,\theta)$ in (4.9) satisfies the prescribed DBCs strongly. Since DBCs are imposed strongly in the NN, it is not necessary to include the boundary loss $\mathcal{L}_b(\theta)$ in the loss function; hence, the loss function reduces to the residual loss, i.e., (4.5). In the case of complex, multi-dimensional domains, the functional form of $g(x)$ may be rather complex, and will often depend on the distance of a point x from the boundary $\partial\Omega$, as discussed in [21]. An alternate approach that uses distance functions and geometry-aware trial functions within a Deep Ritz PINN (i.e., a PINN in which the residual loss is given in weak variational rather than strong form) is the work of Sukumar *et al.* [19].

Since information within the Schwarz alternating method propagates through the DBCs imposed at the subdomain boundaries and past convergence analyses [12, 15, 16] of the method have assumed a strong implementation of these BCs, we explore in this paper several mechanisms for imposing SDBCs within our PINNs for the BVP considered, namely (2.1)–(2.2).

4.2.1. Strong imposition of homogeneous DBCs on the system boundary $\partial\Omega = \{0,1\}$. Consider first the single-domain BVP (2.1)–(2.2) posed on $\Omega = (0,1)$ with boundary $\partial\Omega = \{0,1\}$. It is straightforward to see that the following approximation

$$u(x) \approx \hat{u}(x;\theta) = v(x)NN(x;\theta)\,, \tag{4.10}$$

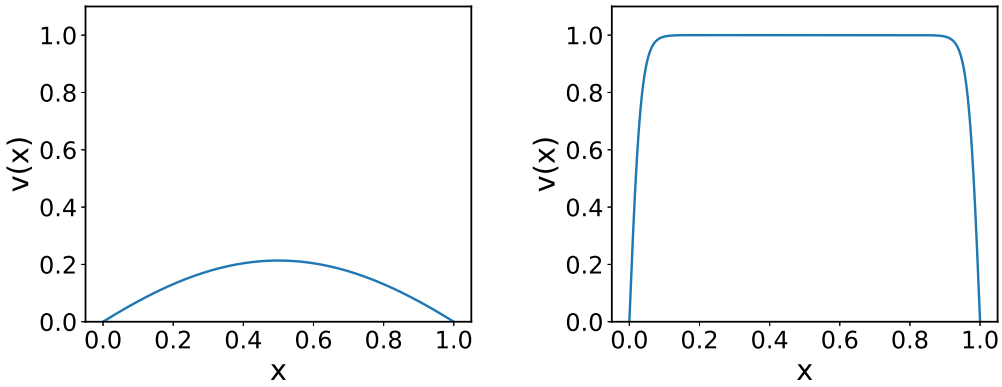
where $v(x)$ is a smooth function satisfying the homogeneous DBCs prescribed on $\partial\Omega$, i.e.,

$$v(0) = v(1) = 0, \tag{4.11}$$

will satisfy strongly the system SDBCs on $\partial\Omega$. In the numerical results presented herein (Section 6), a hyperbolic tangent scaling function of the form

$$v(x) = \tanh(k(1-x))\tanh(kx)\,. \tag{4.12}$$

was employed, for $k > 0$. The scaling function (4.12) is plotted for two values of k, namely $k = 1$ and $k = 30$, in Figure 4.2. The reader can observe that as k is increased, the scaling function $v(x)$ begins to resemble more an more a step function on $\Omega = (0, 1)$.



(a) $k = 1$ (b) $k = 30$

Fig. 4.2: Strong DBC scaling function (4.12) for two values of k: $k = 1$ (left) and $k = 30$ (right).

At first glance, it may seem as though employing a larger k in (4.12) should improve PINN convergence, as $v(x) \approx 1$ in Ω away from the boundary $\partial\Omega$, meaning that this scaling function minimally modifies the NN away from $\partial\Omega$. However, our numerical experiments reveal that selecting higher values of k, e.g., $k = 30$ (Figure 4.2(b)), severely hinders the NN's convergence to a solution. We believe that this happens because $v(x)$ exhibits sharp gradients when $k \gg 1$, requiring the NN to make rapid and substantial changes to the magnitudes of its outputs for a small subset of the spatial domain close to the boundaries, and further complicating the minimization of the loss function. For this reason, for the results presented herein (Section 6), a value of $k = 1$ is used to define $v(x)$ (4.12).

4.2.2. Strong imposition of inhomogeneous DBCs at all subdomain boundaries. Since the goal of this paper is to extend the Schwarz alternating method to the case of PINN-PINN and PINN-FOM couplings, it is important to be able to impose DBCs not only at the system boundaries but also at the interior subdomain (Schwarz) boundaries.

In order to keep the discussion as general as possible for our 1D model problem (2.1), consider a generic overlapping decomposition of Ω into n_D overlapping subdomains, so that $\Omega = \cup_{i=1}^{n_D}\Omega_i$, where $\Omega_i = (\gamma_{2i-2}, \gamma_{2i-1})$, for $i = 1, ..., n_D$, where $\gamma_0 = 0$ and $\gamma_{2n_D-1} = 1$. We begin by defining the following subdomain-local BVP for our targeted 1D advection-diffusion equation

$$-\nu\frac{\partial^2 u_i}{\partial x^2} + \frac{\partial u_i}{\partial x} - 1 = 0, \qquad \text{in } \Omega_i := (\gamma_{2i-2}, \gamma_{2i-1}), \tag{4.13}$$

with boundary conditions

$$u_i(\gamma_{2i-2}) = g_{2i-2}, \qquad u_i(\gamma_{2i-1}) = g_{2i-1}, \tag{4.14}$$

for $g_i \in \mathbb{R}$, with $i = 1, ..., n_D$. Let $NN_i(x;\theta)$ denote a NN trained to represent the solution in Ω_i. The reader can observe that the following function is guaranteed to satisfy strongly the DBCs (4.14) on $\partial\Omega_i$:

$$\hat{u}_i(x;\theta) = v_i(x)NN_i(x;\theta) + \phi_i(x)g_{2i-2} + \psi_i(x)g_{2i-1}, \tag{4.15}$$

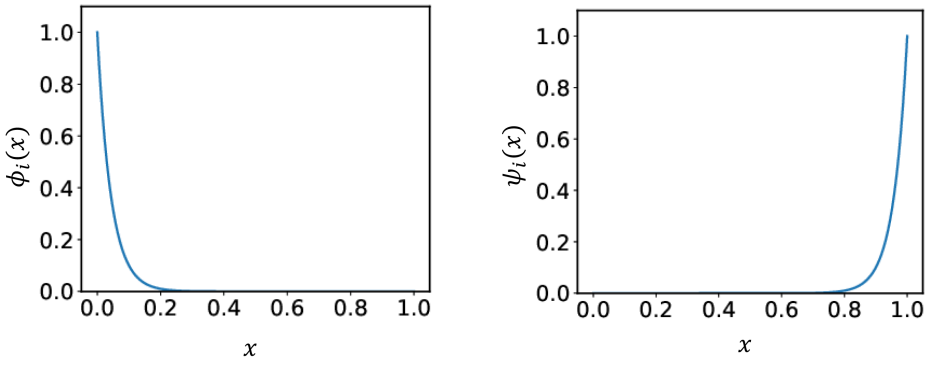
where $v(x)$ is a function such that $v(\gamma_{2i-2}) = v(\gamma_{2i-1}) = 0$, $\phi_i(x)$ is a function such that $\phi_i(\gamma_{2i-2}) = 1$ and $\phi_i(\gamma_{2i-1}) = 0$, and $\psi_i(x)$ is a function such that $\psi_i(\gamma_{2i-2}) = 0$ and $\psi_i(\gamma_{2i-1}) = 1$. In the present work, motivated by the discussion in Section 4.2.1, we employ the following expressions for these functions:

$$v_i(x) = \tanh(x - \gamma_{2i-2}) \tanh(\gamma_{2i-1} - x), \tag{4.16}$$

$$\phi_i(x) = 10^{-10(x-\gamma_{2i-2})}, \tag{4.17}$$

$$\psi_i(x) = 10^{10(x-\gamma_{2i-1})}. \tag{4.18}$$

The function $v_i(x)$ was plotted earlier for the interval $\Omega_i = (0, 1)$ in Figure 4.2(a). The function $\phi_i(x)$ and $\psi_i(x)$, also for the interval $\Omega_i = (0, 1)$, are shown in Figure 4.3. We emphasize that, like the function $v_i(x)$, the inhomogeneous boundary condition functions $\phi_i(x)$ and $\psi_i(x)$ are not unique. While exploring alternate choices for these functions goes beyond the scope of the present paper, we remark that our preliminary numerical experiments suggested that a smoothly varying $\phi_i(x)$ and $\psi_i(x)$ function is in general more effective than a Dirac function, as it minimizes the presence of sharp gradients, which can hinder the convergence of a NN.



(a) $\phi_i(x)$ (b) $\psi_i(x)$

Fig. 4.3: Inhomogeneous boundary condition enforcement functions (4.17) $\phi_i(x)$ and (4.18) $\psi_i(x)$ for the interval $\Omega_i = (0, 1)$.

4.3. Incorporation of data into a PINN. While the appeal of PINNs is at least partly that solution data are not required for training, we remark that it is possible to incorporate available data into the NN by adding a so-called data loss term $\mathcal{L}_d(\theta)$, as in (4.6). This data loss term takes the form of

$$\mathcal{L}_d(\theta) := \frac{1}{d}\sum_{i=0}^{d}\left(\hat{u}(x_i;\theta) - u(x_i)\right)^2, \tag{4.19}$$

is the data loss, where d is the number of snapshots used to evaluate the data loss. IKT comment: Equation (4.19) doesn't make sense... looks like collocation is being done when I think we'd have solution snapshots for different parameter values. TODO: discuss with Chris and Will.

5. The Schwarz alternating method for PINN-PINN and PINN-FOM coupling. Our goal in this paper is to extend the Schwarz alternating method, described earlier in Section 3, to PINN-PINN and PINN-FOM couplings, following an overlapping domain decomposition of the underlying spatial domain Ω. Two coupling scenarios are possible:

- *Coupling Scenario 1.* Use DD and the Schwarz alternating method to facilitate training of PINNs (offline).
- *Coupling Scenario 2.* Use DD and the Schwarz alternating method to couple pre-trained subdomain-local PINNs (online).

As discussed in Section 1, Coupling Scenario 1 has been considered in several recent works, most notably [10, 11, 6], whereas Coupling Scenario 2 was explored in [21]. Herein, attention is mostly focused on Coupling Scenario 1. Specifically, we explore the hypothesis that, by creating and coupling (using the Schwarz alternating method) several subdomain-local PINNs with each other and/or with FOMs, it may be possible to accelerate PINN training. For the targeted advection-diffusion problem (2.1)–(2.2), we are particularly interested in the advection-dominated (high Péclet) regime, in which it is well-known that PINNs are particularly difficult to train [13]. We provide some perspectives on Coupling Scenario 2 later, in Section 7.

In order to develop a PINN-PINN coupling algorithm using the Schwarz alternating method, we combine ideas presented earlier in Sections 3 and 4. As in Section 4.2.2, suppose the subdomain $\Omega = (0,1)$ has been partitioned into n_D overlapping subdomains Ω_i, so that $\Omega = \cup_{i=1}^{n_D}\Omega_i$, where $\Omega_i = (\gamma_{2i-2}, \gamma_{2i-1}) \in \mathbb{R}$ for $i = 1, ..., n_D$, with $\gamma_0 = 0$ and $\gamma_{n_D} = 1$. Let $\mathcal{L}_{r,i}$ and $\mathcal{L}_{d,i}$ denote the residual and data losses, defined earlier in (4.7) and (4.19), respectively, but restricted to subdmomain Ω_i. Let $NN_i(x;\theta)$ denote a NN within domain Ω_i, and let $\hat{u}_i(x;\theta) \approx u_i(x)$ be the approximation of the solution $u(x)$ in Ω_i. We provide some pseudo-code for our online PINN training in Algorithm 1. The reader can observe that both the boundary loss term $\mathcal{L}_{b,i}(\theta)$ and the approximate PINN solution $\hat{u}_i(x)$ depend on the type of DBC enforcement selected, namely the variable $DBCtype$, which can take on three values ($WDBC$, corresponding to weakly-imposed DBCs on all boundaries; $SDBC_{sys}$, corresponding to strongly-imposed system DBCs and weakly imposed Schwarz DBCs; $SDBC_{all}$, corresponding to strongly-imposed DBCs on all system and subdomain boundaries).

TODO: Someone should go through Algorithm 1 carefully and check everything... Irina has not done this year. Would be good to have second pair of eyes on it because the different cases are sort of a mess...

It is straightforward to modify Algorithm 1 to the case where a FOM is employed in one or more subdomains Ω_i. The details are omitted here for the sake of brevity.

Algorithm 1: Alternating overlapping Schwarz PINN-PINN coupling algorithm

Inputs: overlapping DD of Ω into n_D overlapping subdomains $\Omega_i = (\gamma_{2i-2}, \gamma_{2i-1})$; relaxation parameters $\alpha_r, \alpha_d \in [0,1]$; $DBCtype \in \{WDBC, SDBC_{sys}, SDBC_{all}\}$.

Initialize $\hat{u}_i(\gamma_{2i-1}, \theta) = 0$ for $i = 1, ..., n_D$;

Set $NN_{-i}(\cdot;\cdot) = NN_{n_D+1}(\cdot;\cdot) = 0$;

while *Schwarz method unconverged* **do**

  for $i = 1, ..., n_D$ **do**

    Train PINN $NN_i(x;\theta)$ in Ω_i with loss

$$\mathcal{L}_i(\theta) := \alpha_r \mathcal{L}_{r,i}(\theta) + (1-\alpha_r)\mathcal{L}_{b,i}(\theta) + \alpha_d \mathcal{L}_{d,i}(\theta),$$

    where

$$\mathcal{L}_{b,i}(\theta) = \begin{cases} \mathcal{L}_{b,i}^{sys}(\theta) + \mathcal{L}_{b,i}^{sch}(\theta), & \text{if } DBCtype = WDBC, \\ \mathcal{L}_{b,i}^{sch}(\theta), & \text{if } DBCtype = SDBC_{sys}, \\ 0, & \text{if } DBCtype = SDBC_{all}, \end{cases}$$

    with

$$\mathcal{L}_{b,i}^{sys}(\theta) = \begin{cases} NN_i(0;\theta)^2, & \text{if } i = 1, \\ NN_i(1;\theta)^2, & \text{if } i = n_D, \\ 0, & \text{otherwise}, \end{cases}$$

    and

$$\mathcal{L}_{b,i}^{sch}(\theta) = \begin{cases} \left(NN_i(\gamma_{2i-1};\theta) - \hat{u}_{i+1}(\gamma_{2i-1},\theta)\right)^2, & \text{if } i = 1, \\ \left(NN_i(\gamma_{2i-2};\theta) - \hat{u}_{i-1}(\gamma_{2i-2},\theta)\right)^2, & \text{if } i = n_D, \\ \left(NN_i(\gamma_{2i-2};\theta) - \hat{u}_{i-1}(\gamma_{2i-2},\theta)\right)^2 + & \\ \left(NN_i(\gamma_{2i-1};\theta) - \hat{u}_{i+1}(\gamma_{2i-1},\theta)\right)^2, & \text{otherwise}. \end{cases}$$

    Set

$$\hat{u}_i(x;\theta) = \begin{cases} NN_i(x;\theta), & \text{if } DBCtype = WDBC, \\ v_i(x) NN_i(x;\theta), & \text{if } DBCtype = SDBC_{sys}, \\ v_i(x) NN_i(x;\theta) + \phi_i(x) NN_{i-1}(\gamma_{2i-2};\theta) + & \\ \psi_i(x) NN_{i+1}(\gamma_{2i-1};\theta), & \text{if } DBCtype = SDBC_{all}, \end{cases}$$

    where $v_i(x)$, $\phi_i(x)$ and $\psi_i(x)$ are given by (4.2.2), (4.17) and (4.18), respectively;

    Interpolate $\hat{u}_i(x,\theta)$ onto $x = \gamma_{2i}$.

  end

end

6. Numerical Results. In this section, we perform some numerical experiments aimed at understanding to what extent DD combined with the Schwarz alternating method can assist with the training of PINNs for the targed advection-diffusion BVP (2.1)–(2.2).

In order to perform the studies described herein, we created a modular Python code which invoked the Tensorflow library for PINN training/creation. All of the NNs evaluated herein used the same hyperparameters. Each network had a 1D input layer to receive the spatial data, x, two hidden layers each with 20 nodes per layer, and a 1D output layer for the approximations of $u(x)$, similar to the NN architecture shown in Figure 4.1. Each hidden layer employed the Swish activation function, that is,

$$\sigma(z) := \frac{z}{1 + e^{-\mu z}}, \tag{6.1}$$

in (4.1)–(4.3), for $\mu =$?? TODO: what was μ? Ask Will. Irina thinks it's the default value of 1 from looking at TensorFlow documentation, but someone should double check When

calculating the loss, the values of the relaxation parameters in (4.6) were $\alpha_r = 0.25$,

$$\alpha_b = \begin{cases} 1-\alpha_r, & \text{if using WDBCs,} \\ 0, & \text{if using SDBCs,} \end{cases} \qquad \alpha_d = \begin{cases} 1-\alpha_r, & \text{if using data loss,} \\ 0, & \text{otherwise,} \end{cases} \tag{6.2}$$

in all cases. IKT comment: Wouldn't it make sense to just set $\alpha_r = 1$ if not doing WDBCs or data loss? What is done in the code? The input data were not normalized, as they were already on the interval $[0, 1]$. We chose to use the Adam optimizer for all of our NNs with a constant learning rate of 0.001. The number of training epochs for each NN per Schwarz iteration was 1024. For all our experiments, we used $n = 1024$ quasi-random uniform collocation points as inputs to evaluate the residual loss (4.5). IKT comment: I think the 1024 is only true for the full domain not the subdomains. Why were the points quasi-uniform and not just uniform? For experiments in which PINN-FOM couplings were evaluated (Section 6.4), the FOM was generated by discretizing the governing PDE (2.1) using a second-order accurate backward finite different approach with a mesh resolution of $h = \frac{1}{1024}$. This finite difference solution was also considered the ground truth against which all PINN approximations were compared in our relative error calculations. For experiments in which the data loss term (4.19) was included in the NN minimization problem, all FOM representations of this system were created using a second-order accurate backward finite difference approach with 1024 evenly spaced points on the domain $x \in (0, 1)$. IKT comment: I still don't understand how the snapshot data were generated for this steady problem. These details should be added here. For the models in which the data loss term (4.19) was included in the minimization problem, snapshot data was generated on each subdomain Ω_i by interpolating our FOM solution to the collocation points in Ω_i. Our convergence criteria required both the Schwarz relative error to drop below a tolerance of 0.001, and the L_2 relative error in the NN approximation with respect to the reference FOM solution to drop below a tolerance of 0.005. TODO: Double check with Will that these were relative errors. We allowed each model a maximum of 100 Schwarz iterations before cutting off training and declaring the model non-converged for a given problem case.

The study summarized herein is aimed at understanding the relative impact of the following PINN- and coupling-related parameters on both the accuracy and the efficiency of the PINN training process:

- the number of subdomains, n_D;
- the size of the overlap region;
- the type of DBC enforcement in the PINN (WDBC vs. SDBC); and
- the impact of including the data loss term $\mathcal{L}_d(\theta)$ (4.19) in the loss function being minimized.

While Section 4.2.2 presented an approach for implementing SDBCs on both the outer system boundaries and the Schwarz boundaries, unless otherwise noted, in the results herein, we are imposing strongly only the system DBCs (2.2) (see Section 4.2.1), meaning the Schwarz BCs are imposed weakly via a boundary loss term.

6.1. PINN-PINN coupling: parameter sweep study. We begin by investigating the impact of the size of the overlap region and the number of Schwarz subdomains n_D on both the accuracy and the efficiency of the resulting PINN-PINN coupling. To do this, we first perform a parameter sweep study in which we vary n_D between 2 and 5, and the percentage overlap of the subdomains between 5-50% in increments of 5%. TODO: I would like to include more detail on how the overlap size was varied, for reproducibility. We also investigate the impact of using SDBCs vs. WDBCs on the two system boundaries, and the impact of including the data loss (4.19) in the loss function being minimized. We consider a relatively low Péclet number, namely $Pe = 10$, for this initial study.

During our parameter sweep study, we recorded both the CPU time required to satisfy the convergence criteria (Figure 6.1) and the final L_2 relative error with respect to the FOM solution averaged across all subdomain models (Figure 6.2). The reader can observe by inspecting Figures 6.1 and 6.2 that neither the CPU time nor the L_2 relative error exhibit any noticeable trend as the percentage overlap parameter is varied. The former result was initially surprising, as it has been demonstrated in a number of references, including [12, 15, 16, 11], that the number of Schwarz iterations (which typically correlates with the CPU time) is inversely proportional to the size of the overlap region. The result is less surprising when taking into account the fact that PINNs on larger subdomains tend to be more difficult to train for the targeted BVP, especially for the subdomains containing the boundary layer. IKT comment: Did Will look at # Schwarz iterations or just CPU time? Technically the result in the literature is that # Schwarz iterations should go down as the overlap region size is increased, but not necessarily the CPU time. A counterexample is the case where you make the overlap so large, that you are effectively doing near full domain solves at each iterations. It can also be seen in Figure 6.2 that there is no observable correlation between the number of subdomains, n_D, and the L_2 relative error upon PINN convergence. Most likely, this is a result of the solution L_2 relative error being part of the convergence criteria defined within our algorithm, which was set to a value of 0.005. As a consequence, the relative errors at convergence for most of the models evaluated in Figure 6.2 are close to 0.005.

On the other hand, it is clear from Figure 6.1 that there is a noticeable increase in the CPU time needed to achieve convergence with increasing n_D. This is particularly apparent in trials of the PINN with strongly enforced system BCs and no data loss (top right panel in Figure 6.1). The positive relationship between the CPU time and n_D shown here makes sense: the larger the value of n_D, the more PINNs must be trained, which increases the total number of training epochs per Schwarz iteration. It is curious that imposing the DBCs at the system boundaries strongly tends to increase the total CPU time, especially for higher values of n_D (top right panel of Figure 6.1); this result was unexpected, and is explored in more detail below. It is in general unclear from Figures 6.1 and 6.2 whether the addition of the data loss into the objective function being minimized helps or hurts convergence and accuracy (see the bottom two panels in these figures).

Having determined that the size of the overlap region has little effect on the accuracy or efficiency of a given PINN-PINN coupling, we perform a second parameter sweep study in which we fix the overlap percentage at 10%, and vary n_D and the DBC enforcement strategy, and explore the impact of incorporating a data loss term into the loss function being minimized. Here, we consider two Péclet numbers: $Pe = 10$ and $Pe = 100$. The main results are summarized in the Pareto plots shown in Figure 6.3. As expected, the models which incorporated snapshots through a data loss, denoted by the left- and upward-pointing triangles, frequently performed better than their true PINN counterparts in terms of both CPU time and L_2 norm error for the $Pe = 10$ problem case. Employing SDBCs had a noticeable positive effect at the higher Péclet number, $Pe = 100$, as none of the couplings employing WDBCs converged. Curiously, the trend is different for the lower Péclet number, $Pe = 10$: as seen earlier in Figure 6.1, the PINNs with WDBCs, denoted by circles and downward-pointing triangles, converged faster than their SDBC analogs for this case. In general, it is clear from Figure 6.3 that convergence challenges are encountered as the Péclet number is increased: for the larger $Pe = 100$ value, only half of the couplings evaluated converge within the specified maximum number of Schwarz iterations, set to be 100. The unconverged cases are indicated by a marker that is not filled.

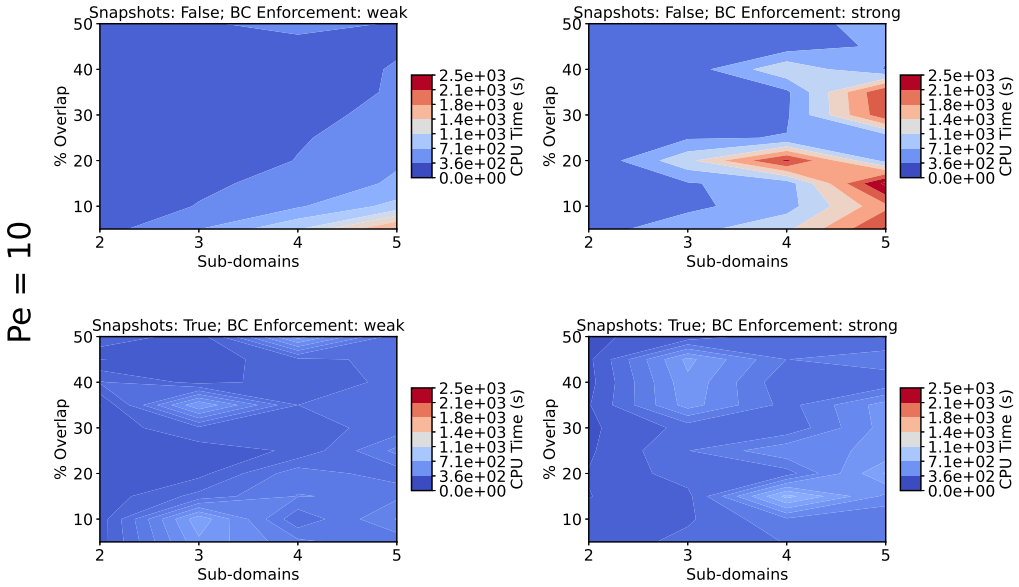


Fig. 6.1: Color maps indicating the CPU time required to achieve Schwarz convergence according to our convergence criteria for a variety of PINN-PINN coupling arrangements, which differ in both n_D and the percentage overlap. Each frame contains such a study for one of four combinations NN techniques denoted by the frame titles. All cases model the advection-diffusion (2.1)–(2.2) with $Pe = 10$.

6.2. PINN-PINN coupling: the impact of WDBCs vs. SDBCs for higher Pe numbers. Following the systematic parameter sweep studies described in the previous subsection, we performed a manual study aimed at determining the largest Péclet number for which an alternating Schwarz-based PINN-PINN coupling can be made to converge, and exploring in more detail the impact of strong vs. weak system DBC implementation on model convergence and accuracy. In this investigation, snapshot data were not included in the loss function being minimized, i.e., $\alpha_d = 0$ in (4.6). In our trials, the highest Péclet number for which convergence was achieved in ≤ 100 Schwarz iterations by PINNs using either form of DBC enforcement was $Pe = 250$ (Figures 6.4(d) and 6.5(d)). The PINN models which achieved convergence for this Pe had $n_D = 4$, 10% subdomain overlap, and a relaxation term of $\alpha_r = 0.2$.

It is interesting to observe that, while both DBC enforcement strategies give rise to couplings that converged in 100 Schwarz iterations, the manner in which the solutions progress is very different. When imposing WDBCs, the subdomain-local PINNs appeared stuck for the first 60-70 Schwarz iterations, with its approximations not resembling the FOM solution at all (Figure 6.4). The coupled model then quickly converged to the FOM solution in the remaining 30 iterations. In contrast, when imposing SDBCs at the system boundaries, the subdomain-local PINNs converged very quickly to a solution that resembled the FOM solution but had an incorrect magnitude, and required an additional 60-70 iterations to correct themselves (Figure 6.5). A similar phenomenon was observed when attempting to converge PINN-PINN couplings with strongly-enforced system DBCs for higher Péclet

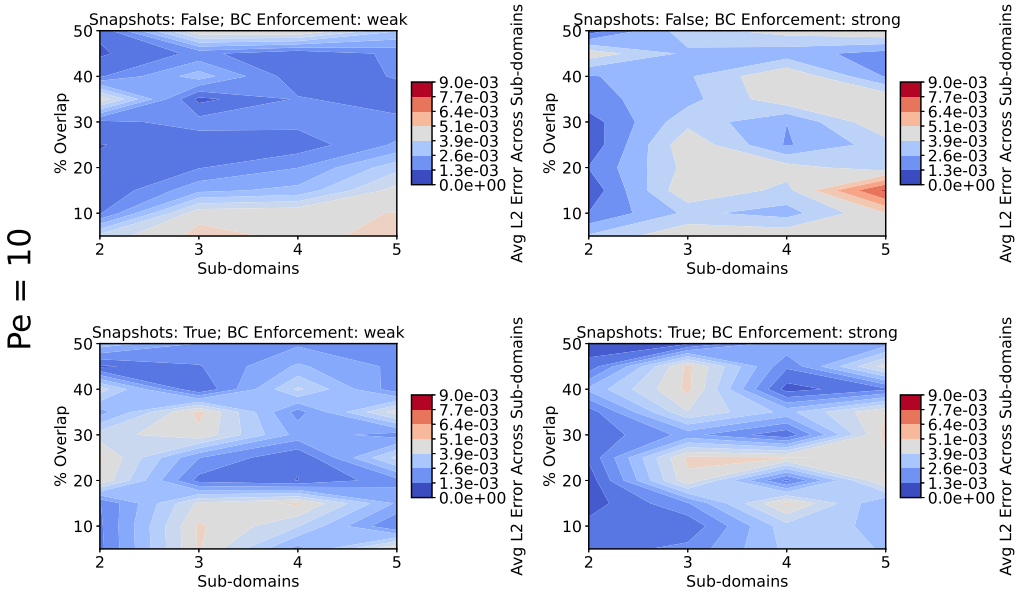
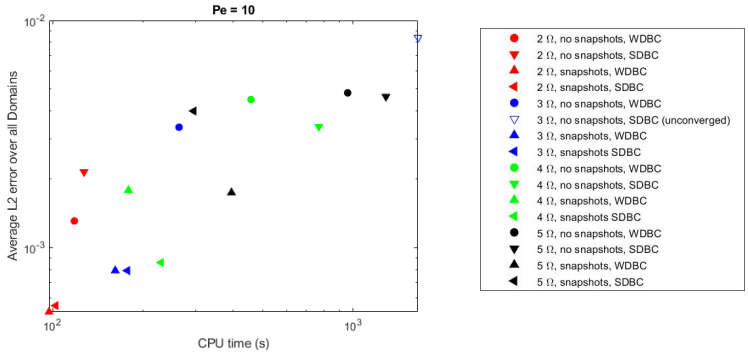


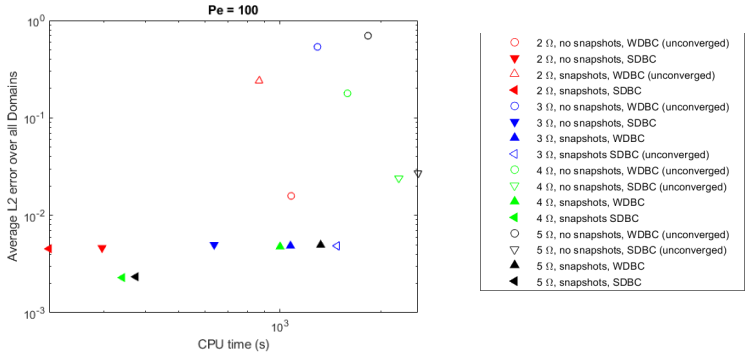
Fig. 6.2: Color maps indicating the L_2 relative error at the end of the Schwarz iteration for a variety of PINN-PINN coupling arrangements, which differ in both n_D and the percentage overlap. Each frame contains such a study for one of four combinations NN techniques denoted by the frame titles. All cases model the advection-diffusion (2.1)–(2.2) with $Pe = 10$.

numbers, namely $Pe \in [300, 500]$. These cases required more than 100 Schwarz iterations to converge. While the two DBC enforcements lead to comparable results when considering CPU times and Schwarz iteration counts alone, it can be argued that the convergence trajectory taken within the SDBC variant of the method is preferable. It is possible that a sort of hybrid DBC enforcement strategy, in which one applies SDBCs in the first N Schwarz iterations, followed by SDBCs in the subsequent M Schwarz iterations, may give the most desirably convergence result, especially for higher Péclet numbers.

6.3. PINN-PINN coupling vs. single-domain PINN. One major question that has not yet been addressed is whether our proposed multi-domain coupling strategy for PINNs is actually beneficial for training such models in the advection-dominated regime when compared to a single-domain PINN having an equivalent architecture. To answer this question, we constructed a single-domain PINN for both forms of DBC enforcement and compared its CPU time with the PINN-PINN couplings described above. We found that a single domain PINN with weakly enforced BCs could achieve full convergence for $Pe = 250$ in 75,776 epochs, the equivalent of 74 Schwarz iterations of our coupled models. It was also observed that the single domain PINN could make partial progress toward solution convergence for a higher Péclet number, $Pe = 300$, in 102,400 epochs, equivalent of 100 Schwarz iterations. From these results, we conclude that a single-domain PINN with a WDBC enforcement took approximately the same amount of CPU time to train as the proposed four subdomain PINN-PINN coupling with strongly-imposed system DBCs discussed in Section 6.2. While this may seem like a negative result for the proposed coupling strategy as a means to



(a) $Pe = 10$



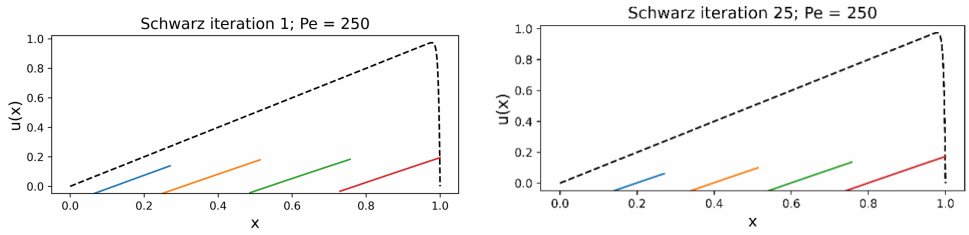
(b) $Pe = 100$

Fig. 6.3: Pareto plots depicting L_2 relative error vs. CPU time for PINN-PINN couplings having different values of n_D and different DBC enforcement types, with and without a data loss term. Results are shown for both converged and non-converged cases modeling the targeted advection-diffusion equation (2.1)–(2.2) for $Pe = 10$ (a) and $Pe = 100$ (b).

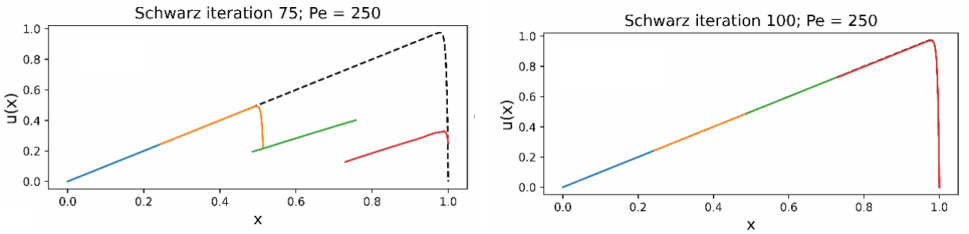
accelerate PINN training, we remark that it is likely possible to improve our PINN-PINN coupling results by: (1) imposing SDBCs not only on the system boundaries but also the Schwarz boundaries, as described in Section 4.2, (2) introducing additional parallelism into the Schwarz iteration by employing the additive variant of the Schwarz alternating method (Remark 1), and/or (3) combining WDBC and SDBC enforcements, as suggested at the end of Section 6.2.

TODO: Consider adding all SDBC results, if have / can get them.

6.4. PINN-FOM coupling. A coupling strategy enabled by the Schwarz alternating method but not yet investigated is the coupling of PINNs with high-fidelity FOMs. Our numerical results reveal that it is possible to obtain a convergent coupled model for arbitrarily-high Péclet numbers in as few as 10 Schwarz iterations by coupling a PINN to a sufficiently-accurate finite difference model used to represent the sharp gradient within the boundary layer that forms. Consider a DD of $\Omega = \Omega_1 \cup \Omega_2$ into $\Omega_1 = (0, ??)$ and $\Omega_2 = (??, 1)$. Figure 6.6 shows the result of a PINN-FOM coupling in which a PINN is specified in Ω_1, and subsequently coupled to a finite difference model with mesh resolution $h = \frac{1}{1024}$ in Ω_2. The reader can observe that it is possible to achieve convergence in just 10 Schwarz iterations for a Péclet number of $Pe = 1 \times 10^6$.

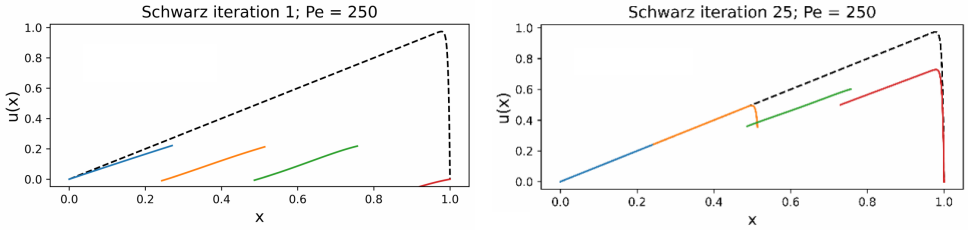


(a) Schwarz iteration #1 (b) Schwarz iteration #25

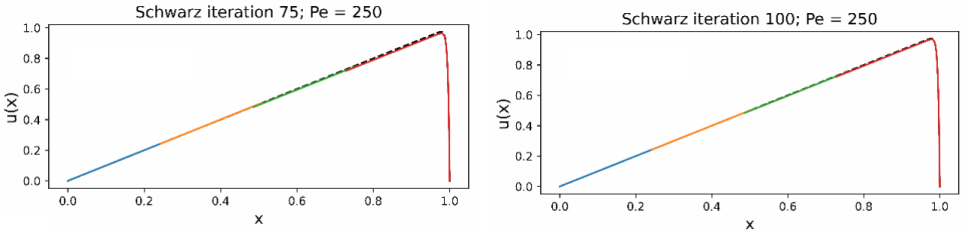


(c) Schwarz iteration #75 (d) Schwarz iteration #100

Fig. 6.4: Four subdomain alternating Schwarz-based solutions to the (2.1)–(2.2) BVP with weakly enforced DBCs for several Schwarz iterations.



(a) Schwarz iteration #1 (b) Schwarz iteration #25



(c) Schwarz iteration #75 (d) Schwarz iteration #100

Fig. 6.5: Four subdomain alternating Schwarz-based solutions to the (2.1)–(2.2) BVP with strongly imposed DBCs on the system boundary $\partial\Omega$ and weakly imposed DBCs on the Schwarz boundaries γ_i for several Schwarz iterations.

The more interesting possibility that the PINN-FOM coupling strategy presents is the idea of pre-training several subdomain PINNs for different sections of a problem using FOM couplings on either side of the PINN. Thereafter, these pre-trained PINNs could be coupled online and may be capable of modeling more difficult problem cases than what is achievable

with PINN-PINN coupled training, as discussed briefly in Section 6. We note, however, that this remains speculative as we have not yet tested such a strategy.

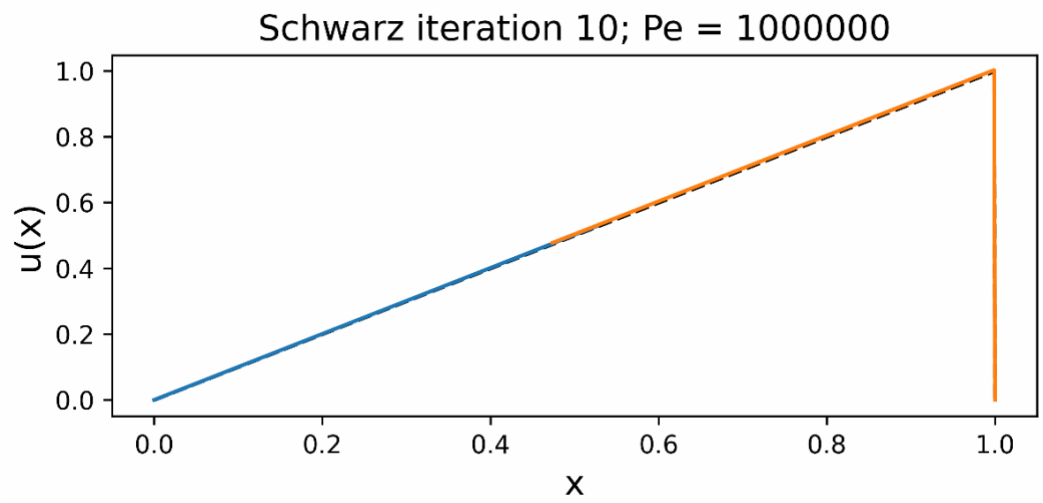


Fig. 6.6: Converged PINN-FOM coupled model with two subdomains and 10% subdomain overlap for the advection-diffusion equation with $Pe = 10^6$. This model used WDBCs. The black dashed line indicates the FOM solution, the blue line indicates the PINN subdomain model solution, and the orange line indicates the FOM subdomain model solution.

7. Online coupling of pre-trained PINNs via the Schwarz alternating method. As mentioned in Section 5, it is possible to utilize the Schwarz alternating method to couple a set of subdomain-local PINNs that were trained independently of one another (Coupling Scenario 2). An approach of this type can be used to create predictive NNs for various unseen domains and boundary conditions through the "tiling" of the pre-trained subdomain-local models [21], and was explored briefly herein. In this preliminary exploratory work, subdomain-local PINNs were trained to receive multiple inputs that would provide information regarding the problem at hand. In addition to the spatial data from collocation points x, the PINN also took as an input the boundary condition data (to enable generalization to a variety of boundary conditions) as well as the Péclet number (to enable prediction across Péclet number). The output remained the same, still approximating the solution state, $u(x)$. We emphasize that, unlike Coupling Scenario 1, the subdomain-local PINNs in Coupling Scenario 2 are trained *independently* of one another. This training does not require the use of the Schwarz alternating method. Once subdomain-local PINNs are trained, a simple Schwarz iteration can be introduced for the online coupling of these models following an overlapping tiling of the models to cover the full domain Ω.

Initial attempts at implementing Coupling Scenario 2 have suggested that it is difficult to create subdomain-local models which generalize to unseen Péclet numbers. The models frequently over-fit the data, approximate $u(x)$ for only the Péclet numbers on which they were trained. It may be possible to mitigate this challenge through alternative training strategies which aim to reduce the potential for over-fitting. We may also consider alterations to the model inputs to make generalization to many Péclet number cases easier.

8. Conclusions. TODO: Irina will write once have more or less complete version of paper.

Future work:

- Rigorous study of strong SDBCs on all boundaries.
- Non-overlapping Schwarz.
- Multi-D/time-dependent problems.
- Additive Schwarz to improve efficiency.
- Pre-training subdomain-local PINNs and coupling via Schwarz.
- Hybrid WDBC + SDBCs.

REFERENCES

[1] J. Barnett, I. Tezaur, and A. Mota, *The schwarz alternating method for the seamless coupling of nonlinear reduced order models and full order models*, 2022.

[2] V. Dolean, A. Heinlein, S. Mishra, and B. Moseley, *Finite basis physics-informed neural networks as a schwarz domain decomposition method*, 2023.

[3] W. E and B. Yu, *The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems*, 2017.

[4] M. J. Gander, *Schwarz methods over the course of time.*, ETNA. Electronic Transactions on Numerical Analysis [electronic only], 31 (2008), pp. 228–255.

[5] Z. Hao, S. Liu, Y. Zhang, C. Ying, Y. Feng, H. Su, and J. Zhu, *Physics-informed machine learning: A survey on problems, methods and applications*, 2023.

[6] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber, *Combining Machine Learning and Domain Decomposition Methods – A Review*. Universitat zu Koln Technical Report Series, Center for Data and Simulation Science. Technical Report ID: CDS-2020-9, 2020.

[7] J. Hoy, I. Tezaur, and A. Mota, *The Schwarz alternating method for multiscale contact mechanics*. in Computer Science Research Institute Summer Proceedings 2021. E. Galvan and D. Smith, eds., Technical Report SAND2021-9886C, 2021.

[8] D. Koliesnikova, A. Mota, I. Tezaur, and J. Hoy, *The schwarz alternating method for contact mechanics*. in preparation, 2023.

[9] Z. K. Lawal, H. Yassin, D. T. C. Lai, and A. Che Idris, *Physics-informed neural network (pinn) evolution and beyond: A systematic literature review and bibliometric analysis*, Big Data and Cognitive Computing, 6 (2022), p. 140.

[10] K. Li, K. Tang, T. Wu, and Q. Liao, *D3m: A deep domain decomposition method for partial differential equations*, IEEE Access, 8 (2020), pp. 5283–5294.

[11] W. Li, X. Xiang, and Y. Xu, *Deep Domain Decomposition Method: Elliptic Problems*, Proceedings of Machine Learning Research, 107 (2020), pp. 269–286.

[12] P. Lions, *On the Schwarz alternating method I.* in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, 1988.

[13] R. Mojgani, M. Balajewicz, and P. Hassanzadeh, *Kolmogorov n–width and Lagrangian physics-informed neural networks: A causality-conforming manifold for convection-dominated PDEs*, Computer Methods in Applied Mechanics and Engineering, 404 (2023), p. 115810.

[14] B. Moseley, A. Markahm, and T. Nissen-Meyer, *Finite basis physics-informed neural networks (FBPINNs): a scalable domain decompositin approach for solving differential equations*, Advances in Computational Mathematics, 69 (2023).

[15] A. Mota, I. Tezaur, and C. Alleman, *The Schwarz alternating method in solid mechanics*, Computer Methods in Applied Mechanics and Engineering, 319 (2017), pp. 19–51.

[16] A. Mota, I. Tezaur, and G. Phlipot, *The Schwarz alternating method for dynamic solid mechanics*, Int. J. Numer. Meth. Engng, (2022), pp. 1–36.

[17] M. Raissi, P. Perdikaris, and G. 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.

[18] H. A. Schwarz, *Ueber einen Grenzübergang durch alternirendes Verfahren*, Zürcher u. Furrer, 1870.

[19] N. Sukumar and A. Srivastava, *Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks*, Computer Methods in Applied Mechanics and Engineering, 389 (2022), p. 114333.

[20] 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), p. 112732.

[21] H. Wang, R. Planas, A. Chandramowlishwaran, and R. Bostanabad, *Mosaic flows: A transferable deep learning framework for solving PDEs on unseen domains*, Computer Methods in Applied Mechanics and Engineering, 389 (2022), p. 114424.

[22] S. Wang, Y. Teng, and P. Perdikaris, *Understanding and mitigating gradient flow pathologies in*

physics-informed neural networks, SIAM Journal on Scientific Computing, 43 (2021), pp. A3055–A3081.