Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 33(10): 5965–5989.
DOI: 10.3934/era.2025265
Received: 30 June 2025
Revised: 28 September 2025
Accepted: 09 October 2025
Published: 15 October 2025

Research article

Domain decomposition based physics and equality constrained neural networks for the Helmholtz equation

Sungmin Won, Evan Olson and Xuemin Tu*

Department of Mathematics, University of Kansas, 1460 Jayhawk Blvd, Lawrence, KS 66045, USA

* **Correspondence:** Email: xuemin@ku.edu.

Abstract: Overlapping and non-overlapping domain decomposition methods are proposed for the physics and equality constrained neural networks (PECANN) to solve the Helmholtz equation. A coarse component is introduced using a small neural network defined on the whole domain with carefully designed loss functions. This coarse component supplies both function values and normal derivatives on the subdomain interfaces to the local solvers, ensuring the scalability of the algorithms; the number of outer iterations remains nearly constant as the number of subdomains increases. Numerical experiments conducted with a wide range of wavenumbers over both standard and enlarged domains demonstrate the efficiency and scalability of the proposed algorithms.

Keywords: PINN; PECANN; domain decomposition; overlapping; non-overlapping; coarse component; Helmholtz

1. Introduction

The Helmholtz equation has various applications in physics and other sciences, such as acoustics, elasticity, electromagnetic waves, and quantum mechanics. Developing efficient and accurate numerical algorithms for the Helmholtz equation has fundamental importance in scientific computing. One of the main difficulties with classical discretization methods is that, to maintain accuracy, a certain number of points per wavelength are required in the engineering applications to avoid the pollution effect [1, 2], which requires much higher mesh resolution for high wavenumbers.

Therefore, the resulting linear system is very large scale and iterative solvers have to be used. However, many iterative methods cannot solve the system efficiently since it is indefinite. Among different approaches for solving the Helmholtz equation, there are many domain decomposition methods developed including both overlapping and non-overlapping methods [2–10]. However, most of the algorithms still struggle with high wave numbers.

Using a neural network (NN) to approximate the solutions of partial differential equations (PDEs)

provides an alternative way to solve the Helmholtz equation due to its universal approximation capability [11]. A popular neural network-based approach for solving PDEs is the physics-informed neural network (PINN) [12], where the governing physics and boundary/initial conditions are incorporated into the training process directly. The PINNs have been applied to various PDEs and related inverse problems [13–21]. The physics and equality constrained artificial neural networks (PECANN) [22] reformulates the minimization problem proposed in the PINN into a constrained optimization problem and employs the augmented Lagrangian method (ALM) [23]. This approach does not need to manually choose the weights for the different terms in the PINN loss function, and therefore improves the efficiency for certain applications [22]. However, when it comes to large-scale applications of both the PINN and PECANN methods, globally training a single neural network to approximate PDE solutions incurs a high computational cost, which is a major drawback compared to traditional methods.

Domain decomposition methods (DDMs) [24, 25] divide the computational domain into smaller subdomains, and then reduce the global large problem into smaller subdomain problems. Since these subdomain problems are considered relatively easier to solve than the original problem, DDMs have been widely used for large-scale simulations based on classical discretization methods. DDMs for the PINN, PECANN, and finite basis physics-informed neural networks (FBPINNs), including both overlapping and non-overlapping domain partitions, have been proposed in [26–34]. Among them, the Helmholtz equation is considered in [30, 33, 34]. Overlapping multi-level FBPINNs are used in [33]. One-level non-overlapping DDMs are used for PECANN [34] and PINN [30]. In particular, PINN with plane wave activation-based neural networks are applied in [30], and the numerical experiments show that the convergence rate remains nearly constant as the wavenumber increases, given that suitable Robin parameters are chosen on the subdomain interfaces. However, the scalability of the algorithms is not addressed in [30, 33].

Following our previous work [35], where scalable two-level domain decomposition (DD) methods are proposed for PECANN with the applications to the positive definite elliptic problems, we develop both overlapping and non-overlapping DD-enhanced PECANN methods for the Helmholtz equation. In this study, the subdomain interface operators include both function values and normal derivatives for overlapping and non-overlapping methods. For the non-overlapping methods, incorporating the normal derivative information is essential to ensure the equivalence between the original problems and the corresponding domain decomposition formulations [25, 36]. Particularly, for the Helmholtz equation, the usual Dirichlet boundary conditions commonly used on subdomain interfaces for the Poisson equation do not guarantee the nonsingularity of the local subdomain problems unless the subdomain size is sufficiently small [36]. For the overlapping methods, including the normal derivative information helps stabilize the algorithms and improves their performance. Similar observations were reported in [8], where normal derivative terms were included in the transmission conditions to design optimized Schwarz methods for the Helmholtz equation using overlapping domain decomposition methods based on traditional discretizations.

For our two-level methods, the coarse component provides both function values and normal derivative information for the subdomain interface updates. This additional normal derivative information enhances and stabilizes the performance of both overlapping and non-overlapping two-level methods. Particularly, it turns out to be highly essential for the non-overlapping methods.

The main contributions in this work are the following:

1) We present both overlapping and non-overlapping methods for the Helmholtz equation within a unified framework.
2) For the one-level methods, PECANN is used as the subdomain local solvers. In addition to function values, normal derivative constraints are imposed for both overlapping and non-overlapping partitions.
3) For the two-level methods, we introduce a coarse component to improve scalability. In contrast to the elliptic cases studied in [35], the coarse model in our approach provides not only function values, but also normal derivatives at the subdomain interfaces, thereby enhancing the convergence of the local subdomain models.

The rest of the paper is organized as follows. We first setup the problem and review the PECANN methods in Section 2. A unified framework for overlapping and non-overlapping DD-enhanced PECANN is proposed in Section 3 including both one-level and two-level methods. In Section 4, numerical experiments for various wavenumbers are provided to demonstrate the efficiency of the proposed overlapping and non-overlapping methods. Finally, a conclusion is given in Section 5.

2. The problem setting and the PECANN methods

Let $\Omega \subset \mathbb{R}^d$ be a bounded Lipschitz domain with boundary $\partial\Omega$. We seek the solution $u(\mathbf{x})$ of the following Helmholtz equation:

$$\begin{aligned} \mathcal{H}(u) := -\Delta u - \kappa^2 u = f, \quad & \text{in } \Omega, \\ \mathcal{B}(u) = g, \quad & \text{on } \partial\Omega, \end{aligned} \tag{2.1}$$

where $\mathcal{H}$ is the Helmholtz operator, $\mathcal{B}$ is a boundary operator, and f and g are given functions. The constant κ is called the wave number. Using standard finite element or finite difference discretization methods, solving (2.1) can be transferred to solving an indefinite linear system, which is significantly expensive with many classical iterative methods, in particular when κ is large. Neural-network-based solutions can be an alternative approach to (2.1), taking advantage of its mesh-free structure.

Physics-informed neural networks (PINNs) were first proposed in [12] as neural network-based approximators for partial differential equations (PDEs). PINNs essentially find data-driven solutions to PDEs using neural networks, with loss functions incorporating the physical laws and boundary conditions. Let u_θ denote the approximated solutions by neural networks based on PINNs, where θ is the set of parameters, with weight matrices W_l and a bias vector b_l from the feedforward neural network

$$\begin{aligned} u_\theta(\mathbf{x}) &= (T_L \circ T_{L-1} \circ ... T_2 \circ T_1)(\mathbf{x}), \\ \text{with } T_l(\mathbf{x}) &= a_l(W_l\mathbf{x} + b_l). \end{aligned} \tag{2.2}$$

Here, L is the number of layers and a_l is an activation function that enables the network to learn non-linear, more complex outputs. In the training process, the neural network seeks the optimal parameters W_l and b_l through a chosen optimizer to minimize a loss function that incorporates physical laws and boundary conditions. This training process can be viewed as solving an unconstrained optimization problem. As these neural networks are able to approximate universal solutions over an entire domain without requiring any grids, they are often considered feasible for application to high-dimensional problems or various types of PDEs. However, it has been proven that the PINNs approach does

not always guarantee good performance, especially for more challenging problems, such as elliptic problems with rapid oscillations, due to the structure of their loss function [22]. Physics and equality constrained artificial neural networks (PECANNs) [22] suggest a new interpretation of the loss function used in the PINNs as a constrained optimization problem in the presence of boundary conditions as

$$\begin{aligned} &\min_{\theta\in\mathbb{R}^n} \mathcal{L}_{\mathcal{H}}(\theta), \\ &\text{with the constraint } \mathcal{L}_{\mathcal{B}}(\theta) = 0, \end{aligned} \tag{2.3}$$

where

$$\mathcal{L}_{\mathcal{H}}(\theta) = \frac{1}{N_{\mathcal{H}}} \sum_{k=1}^{N_{\mathcal{H}}} \left| \mathcal{H}\left(u_\theta(\mathbf{x}_h^{(k)})\right) - f(\mathbf{x}_h^{(k)}) \right|^2, \tag{2.4}$$

$$\mathcal{L}_{\mathcal{B}}(\theta) = \frac{1}{N_{\mathcal{B}}} \sum_{k=1}^{N_{\mathcal{B}}} \left| \mathcal{B}\left(u_\theta(\mathbf{x}_b^{(k)})\right) - g(\mathbf{x}_b^{(k)}) \right|^2. \tag{2.5}$$

Here, $\{\mathbf{x}_h^{(k)}\}_{k=1}^{N_{\mathcal{H}}}$ is the set of data points in Ω for approximating the physics loss $\mathcal{L}_{\mathcal{H}}$, and $\{\mathbf{x}_b^{(k)}\}_{k=1}^{N_{\mathcal{B}}}$ is the set of points on $\partial\Omega$ for approximating the boundary loss $\mathcal{L}_{\mathcal{B}}$. (2.3) can be reformulated into the following unconstrained optimization problem by employing augmented Lagrangian methods (ALM) as discussed in [23]:

$$\min_{\theta\in\mathbb{R}^n} \mathcal{L}(\theta;\lambda,\mu) = \mathcal{L}_{\mathcal{H}}(\theta) + \lambda_b \mathcal{L}_{\mathcal{B}}(\theta) + \frac{\mu_b}{2} \mathcal{L}_{\mathcal{B}}(\theta)^2, \tag{2.6}$$

where λ_b is the Lagrange multiplier and μ_b is a scalar penalty. After each optimization step, λ_b and μ_b are then updated as follows:

$$\lambda \leftarrow \lambda + \mu_b \mathcal{L}_{\mathcal{B}}(\theta), \quad \mu_b \leftarrow \min(2\mu_b, \mu_{\max}). \tag{2.7}$$

It has been shown that the PECANN methods improve accuracy for certain forward and inverse problems compared to the PINNs [22].

3. Domain decomposition (DD) enhanced PECANN methods

Domain decomposition methods (DDMs) [25] divide the computational domain into smaller subdomains, allowing each subdomain problem to be solved in parallel on distinct processors. There are two families of DDMs: overlapping and non-overlapping domain decomposition methods. In the overlapping methods, subdomains overlap beyond their boundaries, whereas in non-overlapping methods, subdomains meet only at the interface without having any overlaps. Figure 1 shows an example of such decompositions into 16 subdomains over the unit square.

Various overlapping and non-overlapping domain decomposition methods have been applied to the Helmholtz equation using classical discretization methods [2–10]. Different from the elliptic problems, to ensure the convergence of the Schwarz type methods, the subdomain interface conditions have to be designed carefully for the Helmholtz equation [2, 4, 6, 8, 36]. For the non-overlapping methods [3,5,7], a special coarse component, formed by the plane waves, is constructed to ensure the scalability of algorithms. Following [2], one-level deep domain decomposition methods are proposed in [30] with the PINN approach. In [34], the combinations of the subdomain interface function value, the

normal derivative, and the tangential derivative are used in the domain-decomposition-based PECANN methods for the Helmholtz equation.

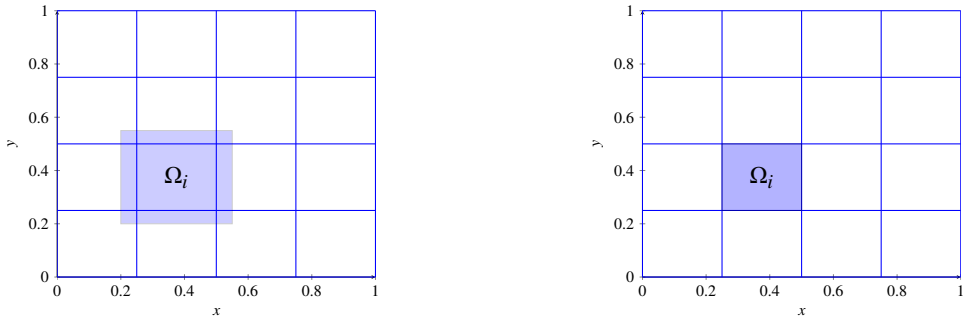

Figure 1. Subdomain partitions: Left: overlapping; Right: non-overlapping.

In this paper, we propose both overlapping and non-overlapping domain-decomposition-enhanced PECANN for the Helmholtz equation. In [34], three quantities are exchanged at the subdomain interfaces. However, our non-overlapping method uses only two: function values and normal derivatives between neighboring subdomains. More importantly, we propose two-level methods that introduce a coarse component into the algorithm. The two-level method greatly improves scalability for low κ and makes the approach applicable to large-scale computations where a large number of the subdomains are required. We first decompose our original computational domain Ω into subdomains as $\Omega = \bigcup_{i=1}^{N_s} \Omega_i$, where N_s is the number of subdomains and Ω_i represents the ith subdomain. We note that the subdomain partitions can be overlapped or non-overlapped, see Figure 1. The global problem (2.1) restricted to the subdomain Ω_i can be written as follows:

$$\begin{aligned} \mathcal{H}(u_i) &= f, \quad \text{in } \Omega_i, \\ \mathcal{B}(u_i) &= g, \quad \text{on } \partial\Omega_i \cap \partial\Omega, \\ \mathcal{D}_i(u_i) &= U_i, \quad \text{on } \Gamma_i = \partial\Omega_i \setminus \partial\Omega, \end{aligned} \tag{3.1}$$

where u_i is the solution to the local problem (3.1) over Ω_i and $\mathcal{D}_i$ is an interface operator. We call Γ_i the subdomain interface since it is not part of the original domain boundary. The subdomain solution u_i is represented by a local neural network and first trained in parallel to approximate the solutions to (3.1) on each subdomain. Next, these local solutions are combined using a set of partition of unity functions, denoted as $\chi_i(\mathbf{x})$, which equals 1 if $\mathbf{x} \in \Omega_i \cup \partial\Omega_i$ and 0 otherwise [25]. The approximated solution over the whole domain is presented as $\hat{u}$, and defined as

$$\hat{u}(\mathbf{x}) = \sum_{i=1}^{N_s} u_{\theta_i}(\mathbf{x})\chi_i(\mathbf{x}), \quad \forall \mathbf{x} \in \Omega \cup \partial\Omega. \tag{3.2}$$

U_i, the subdomain interface values, is updated for the next iteration as follows:

$$U_i \leftarrow \tau_r U_i + (1 - \tau_r)\mathcal{D}_i(\hat{u}), \tag{3.3}$$

where $\tau_r \in [0, 1)$ is a relaxation parameter. In the following two sections, we will give our detailed one-level and two-level DD-enhanced PECANN methods.

3.1. One-level PECANN methods

The one-level PECANN method begins with an initial guess U_i for u_{θ_i} at the subdomain interface Γ_i, where u_{θ_i} is the subdomain neural network model defined on the subdomain Ω_i. Given the initial guess U_i in (3.1), each u_{θ_i} seeks an optimal set of parameters that minimizes its preset loss function over several epochs in parallel, and then exchanges its solution with neighboring subdomains during the communication process. The updated solution from the communication is then used to update the initial interface value U_i for the next subdomain training. This subdomain training–communication process is repeated until the following L^2 relative error is achieved:

$$\varepsilon = \frac{\|u - \hat{u}\|_2}{\|u\|_2}, \tag{3.4}$$

where $\|f\|_2 = \sqrt{\sum_{\mathbf{x}} f^2(\mathbf{x})}$ with $\{\mathbf{x}\}$ is a set of uniform samples from a grid on $\Omega \cup \partial\Omega$.

In the following, we describe the loss function of u_{θ_i}. For any subdomain interface $\Gamma_{i,j} = \partial\Omega_i \cap \Omega_j$, we define the interface operator $\mathcal{D}_i$ and interface value U_i in (3.1) for Ω_i as

$$\mathcal{D}_i = \begin{pmatrix} u_i \\ \frac{\partial u_i}{\partial n} \end{pmatrix}, \tag{3.5}$$

$$U_i = \begin{pmatrix} u_{ij} \\ u_{n,ij} \end{pmatrix}, \tag{3.6}$$

where, for $0 \le c_i \le 1$,

$$\begin{aligned} u_{ij}(\mathbf{x}) &= \left(c_i u_{\theta_i} + (1 - c_i) u_{\theta_j}\right)(\mathbf{x}), \quad \forall \mathbf{x} \in \Gamma_{ij}, \\ u_{n,ij}(\mathbf{x}) &= \left(c_i \frac{\partial u_{\theta_i}}{\partial n_i} + (1 - c_i) \frac{\partial u_{\theta_j}}{\partial n_j}\right)(\mathbf{x}), \quad \forall \mathbf{x} \in \Gamma_{ij}. \end{aligned} \tag{3.7}$$

Here, $\dfrac{\partial u_{\theta i}}{\partial n_i}$ represents the outward normal derivative of u_i at the interface. This aims to introduce additional continuity conditions at the interface so that the approximations and their derivatives from two or more networks match where they meet. The global constrained optimization problem (2.3) is divided into local constrained optimization problems on each subdomain Ω_i as

$$\begin{gathered} \min_{\theta_i \in \mathbb{R}^n} \mathcal{L}_{\mathcal{H},i}(\theta_i), \\ \text{with the constraints } \mathcal{L}_{\mathcal{B},i}(\theta_i) = 0 \text{ and } \mathcal{L}_{\mathcal{D},i}(\theta_i) = 0, \end{gathered} \tag{3.8}$$

where $\mathcal{L}_{\mathcal{H},i}$, $\mathcal{L}_{\mathcal{B},i}$, and $\mathcal{L}_{\mathcal{D},i}$ represent the following physics, boundary, and interface losses defined over the corresponding subdomain Ω_i, respectively:

$$\mathcal{L}_{\mathcal{H},i}(\theta_i) = \frac{1}{N_{\mathcal{H},i}} \sum_{k=1}^{N_{\mathcal{H},i}} \left| \mathcal{H}\left(u_{\theta_i}(\mathbf{x}_{h,i}^{(k)})\right) - f(\mathbf{x}_{h,i}^{(k)}) \right|^2, \tag{3.9}$$

$$\mathcal{L}_{\mathcal{B},i}(\theta_i) = \frac{1}{N_{\mathcal{B},i}} \sum_{k=1}^{N_{\mathcal{B},i}} \left| \mathcal{B}\left(u_{\theta_i}(\mathbf{x}_{b,i}^{(k)})\right) - g(\mathbf{x}_{b,i}^{(k)}) \right|^2, \tag{3.10}$$

$$\mathcal{L}_{\mathcal{D},i}(\theta_i) = \frac{1}{N_{\mathcal{D},i}} \sum_{k=1}^{N_{\mathcal{D},i}} \left| \mathcal{D}\left(u_{\theta_i}(\mathbf{x}_{d,i}^{(k)})\right) - U_i(\mathbf{x}_{d,i}^{(k)}) \right|^2, \tag{3.11}$$

where $\{\mathbf{x}_{h,i}^{(k)}\}_{k=1}^{N_{\mathcal{H},i}} \subset \Omega_i$, $\{\mathbf{x}_{b,i}^{(k)}\}_{k=1}^{N_{\mathcal{B},i}} \subset \partial\Omega \cap \partial\Omega_i$, and $\{\mathbf{x}_{d,i}^{(k)}\}_{k=1}^{N_{\mathcal{D},i}} \subset \Gamma_i$.

Different from the augmented Lagrange multiplier introduced in (2.6), for the Helmholtz equation, we introduce a vector Lagrange multiplier λ_b whose components are tailored to each constraint at a single boundary/interface data points $\mathbf{x}_b^{(k)}$ and $\mathbf{x}_d^{(k)}$, along with scalars μ_b and μ_d. Therefore, the subdomain constrained optimization problem (3.8) can be converted into the following unconstrained optimization problem:

$$\min_{\theta_i \in \mathbb{R}^n} \mathcal{L}_i(\theta_i; \lambda_{b,i}, \lambda_{d,i}, \mu_{b,i}, \mu_{d,i}) = \mathcal{L}_{\mathcal{H},i}(\theta_i) + \lambda_{b,i}\mathcal{L}_{\mathcal{B},i}(\theta_i) + \lambda_{d,i}\mathcal{L}_{\mathcal{D},i}(\theta_i) + \frac{\mu_{b,i}}{2}\mathcal{L}_{\mathcal{B},i}(\theta_i)^2 + \frac{\mu_{d,i}}{2}\mathcal{L}_{\mathcal{D},i}(\theta_i)^2, \tag{3.12}$$

where

$$\lambda_{b,i}\mathcal{L}_{\mathcal{B},i}(\theta_i) = \frac{1}{N_{\mathcal{B},i}} \sum_{k=1}^{N_{\mathcal{B},i}} \lambda_{b,i}^{(k)} \left| \mathcal{B}\left(u_{\theta_i}(\mathbf{x}_{b,i}^{(k)})\right) - g(\mathbf{x}_{b,i}^{(k)}) \right|^2, \tag{3.13}$$

$$\lambda_{d,i}\mathcal{L}_{\mathcal{D},i}(\theta_i) = \frac{1}{N_{\mathcal{D},i}} \sum_{k=1}^{N_{\mathcal{D},i}} \lambda_{d,i}^{(k)} \left| \mathcal{D}\left(u_{\theta_i}(\mathbf{x}_{d,i}^{(k)})\right) - U_i(\mathbf{x}_{d,i}^{(k)}) \right|^2, \tag{3.14}$$

$$\mu_b\mathcal{L}_{\mathcal{B},i}(\theta_i)^2 = \frac{1}{N_{\mathcal{B},i}} \sum_{k=1}^{N_{\mathcal{B},i}} \mu_{b,i} \left| \mathcal{B}\left(u_{\theta_i}(\mathbf{x}_{b,i}^{(k)})\right) - g(\mathbf{x}_{b,i}^{(k)}) \right|^4, \tag{3.15}$$

$$\mu_d\mathcal{L}_{\mathcal{D},i}(\theta_i)^2 = \frac{1}{N_{\mathcal{D},i}} \sum_{k=1}^{N_{\mathcal{B},i}} \mu_{d,i} \left| \mathcal{D}\left(u_{\theta_i}(\mathbf{x}_{d,i}^{(k)})\right) - U_i(\mathbf{x}_{d,i}^{(k)}) \right|^4. \tag{3.16}$$

At each iteration in the training process of the neural network u_{θ_i}, each $\lambda_{\cdot,i}^{(k)}$ and $\mu_{\cdot,i}$ are updated as follows: Let δ and $\mu_{\max}$ be predetermined constants, $\eta = 0$, and

$$\begin{aligned} &\text{if } \sqrt{\pi(\theta)} > \max(0.25\eta, \delta): \\ &\qquad \mu_{\cdot,i} \leftarrow \min(2\mu_{\cdot,i}, \mu_{\max}), \\ &\qquad \lambda_{\cdot,i}^{(k)} \leftarrow \lambda_{\cdot,i}^{(k)} + \mu_{\cdot,i} C_{\cdot,i}^{(k)}(\theta), \\ &\qquad \eta \leftarrow \sqrt{\pi(\theta)}, \end{aligned} \tag{3.17}$$

where $C_{\cdot,i}^{(k)}(\theta)$ represents the kth constraint in the subdomain Ω_i, given by either $|\mathcal{B}\left(u_{\theta_i}(\mathbf{x}_{b,i}^{(k)})\right) - g(\mathbf{x}_{b,i}^{(k)})|^2$ or $|\mathcal{D}\left(u_{\theta_i}(\mathbf{x}_{d,i}^{(k)})\right) - U_i(\mathbf{x}_{d,i}^{(k)})|^2$, and $\pi(\theta) = \sum_{k=1}^{N_{\cdot,i}} \left(C_{\cdot,i}^{(k)}(\theta)\right)^2 / N_{\cdot,i}$. For further discussion of the update rule, see [37].

In the one-level method, each local network u_{θ_i} seeks θ_i to minimize $\mathcal{L}_i$, and the approximated solutions are unified into $\hat{u}$, as defined in (3.2), and the subdomain interface values U_i are updated using (3.3). Algorithm 1 details the procedure of the one-level PECANN method.

Algorithm 1 One Level PECANN

Require: N_l=number of epochs for local networks, τ=tolerance, ε=initial relative error, $U_{i,0}$=initial interface guesses for $\forall i$, $\lambda_{\mathcal{B},i}, \lambda_{\mathcal{D},i}, \mu_{\mathcal{B},i}, \mu_{\mathcal{D},i}$ = initial set of ALM parameters for $\forall i$

iter = 0

while $\varepsilon > \tau$ and iter < max_iter **do**

1. Subdomain networks training:

for count = $1, \cdots, N_l$ **do**

For each i, update θ_i to minimize $\mathcal{L}_i$

Update $\lambda_{b,i}$, $\lambda_{d,i}$, $\mu_{b,i}$ and $\mu_{d,i}$ by the update rule (3.17)

end for

2. Communication:

Evaluate $\hat{u}$ in (3.2) and ε

Update interface values: Update U_i for $\forall \mathbf{x} \in \Gamma_i$ using (3.3)

iter = iter+1

end while

3.2. Two-level PECANN methods

The one-level methods can distribute the total workload over subdomain tasks, with the workload assigned to each subdomain model becoming lighter as the number of subdomains increases. However, the performance of the one-level methods deteriorates rapidly with a large number of subdomains. This issue arises because each subdomain exchanges information only with its adjacent subdomains.

One approach to alleviate such an issue is to introduce a hierarchy into the algorithms as in [25], resulting in two-level methods: one additional neural network representing the entire domain solution, and the other local neural networks consisting of subdomain solutions. We employ the additional network with fewer parameters for the entire domain, denoted as the coarse component u_c, and the subdomain networks for each subdomain, denoted as u_{θ_i} for $i = 1, \cdots, N_s$. Note that the coarse model is chosen to be lightweight, intended to achieve a workload comparable or lower than that of the one-level method. The physics and boundary loss functions for the coarse model are

$$\mathcal{L}_{\mathcal{H}}(\theta_c) = \frac{1}{N_{\mathcal{H}}^c} \sum_{k=1}^{N_{\mathcal{H}}^c} \left| \mathcal{H}\left(u_{\theta_c}(\mathbf{x}_{h,c}^{(k)})\right) - f(\mathbf{x}_{h,c}^{(k)}) \right|^2, \tag{3.18}$$

$$\mathcal{L}_{\mathcal{B}}(\theta_c) = \frac{1}{N_{\mathcal{B}}^c} \sum_{k=1}^{N_{\mathcal{B}}^c} \left| \mathcal{B}\left(u_{\theta_c}(\mathbf{x}_{b,c}^{(k)})\right) - g(\mathbf{x}_{b,c}^{(k)}) \right|^2, \tag{3.19}$$

where $N_{\mathcal{H}}^c$ and $N_{\mathcal{B}}^c$ are the number of sample points for the physics and boundary loss, respectively, and $\{\mathbf{x}_{h,c}^{(k)}\}_{k=1}^{N_{\mathcal{H}}^c} \subset \Omega$ and $\{\mathbf{x}_{b,c}^{(k)}\}_{k=1}^{N_{\mathcal{B}}^c} \subset \partial\Omega$. To create a dataset for the coarse model, we randomly select $N_{\mathcal{H}}^c/N_s$ points from each subdomain's set of points. That is, $\{\mathbf{x}_{h,c}^{(k)}\}_{k=1}^{N_{\mathcal{H}}^c} \subset \bigcup_{i=1}^{N_s} \{\mathbf{x}_{h,i}^{(k)}\}_{k=1}^{N_{\mathcal{H}}^i}$. This stratified sampling ensures that the coarse model influences the subdomain models evenly and maintains a consistent data size.

Similar to the coarse problems in [35, 38], we add the following local fitting loss to incorporate the local model information into the training of u_{θ_c}:

$$\mathcal{L}_C(\theta_c) = \frac{1}{N_C}\sum_{k=1}^{N_C} |u_{\theta_c}(\mathbf{x}_c^{(k)}) - \hat{u}(\mathbf{x}_k^c)|^2, \tag{3.20}$$

where $\hat{u}$ is obtained from the local subdomain models, defined in (3.23) , and $\{\mathbf{x}_c^{(k)}\}_{k=1}^{N_C}$ denotes the set of points that the coarse model aims to match $\hat{u}$. These points can be chosen to coincide with $\{\mathbf{x}_{h,c}^{(k)}\}_{k=1}^{N_{\mathcal{H}}^c}$.

Since the subdomain local solutions might not be accurate and the same for $\hat{u}$, we treat the fitting loss as a soft constraint and have the following constrained optimization problem:

$$\begin{aligned} &\min_{\theta_c\in\mathbb{R}^n} \mathcal{L}_{\mathcal{H}}(\theta_c) + w_C\mathcal{L}_C(\theta_c), \\ &\text{with the constraint } \mathcal{L}_{\mathcal{B}}(\theta_c) = 0, \end{aligned} \tag{3.21}$$

where w_C is a fixed weight used to emphasize or de-emphasize the fitting loss.

Different from the approaches in [35], during the communication step, in additional to the function values, the coarse model also provides the normal derivatives to the subdomain local models. Therefore, the interface value U_i for the subdomain Ω_i is updated as

$$U_i = \tau_r U_i + (1-\tau_r)\begin{pmatrix} \hat{u}_i \\ \hat{u}_{n,i} \end{pmatrix}, \tag{3.22}$$

where

$$\begin{aligned} \hat{u}_i(\mathbf{x}) &= (1-\omega_c)u_{ij}(\mathbf{x}) + \omega_c u_c(\mathbf{x}), \quad \forall \mathbf{x}\in\Gamma_{ij}, \\ \hat{u}_{n,i}(\mathbf{x}) &= (1-\omega_c)u_{n,ij}(\mathbf{x}) + \omega_c\frac{\partial u_c}{\partial n_c}(\mathbf{x}), \quad \forall \mathbf{x}\in\Gamma_{ij}. \end{aligned} \tag{3.23}$$

The weight $\omega_c \in (0, 1]$ is initially set to be 1 and is multiplied by a decay factor $r \in [0, 1)$ at the end of each outer iteration, causing the influence of the coarse model to diminish as iterations progress. During the communication process, the coarse model serves as a global information source to correct the subdomain models. Conversely, the subdomain models also help the coarse model capture fine-grid features.

The two-level method proceeds as follows: The coarse model is initially trained for a few epochs to provide the initial values for the subdomain models. Using these initial interface estimates from the coarse model, the subdomain models then seek local solutions independently over several epochs. Once the parallel training stops, the subdomain models exchange information through a combination of relaxation between their direct neighbors and the coarse model. After the first outer iteration, the cycle of coarse model training, local model training, and communication is repeated until convergence. Algorithm 2 outlines the procedure in detail.

The choice of network depth and width for the coarse model is important in the stability and convergence of our two-level algorithms. Two additional key factors are the number of training sample points and epochs used for the coarse model in each iteration. A sufficiently large coarse model, with adequate depth, width, sample points, and training epochs, can meet the required tolerance on its own and provide accurate subdomain interface values, enabling the local subdomain models to converge

rapidly. However, training such a coarse model is significantly more expensive than training the local subdomain models. Conversely, if the coarse model is too small (with shallow depth and narrow width) or insufficiently trained (with too few sample points or epochs), its training cost will be relatively low, but it may yield inaccurate interface values. In this case, the coarse model fails to support, and may even hinder, the convergence of the local subdomain models. Therefore, a carefully designed coarse model that balances accuracy and efficiency is essential to the effectiveness of the two-level methods.

Algorithm 2 Two Level PECANN

Require: N_l, N_c=number of epochs for local and coarse networks, τ=tolerance, ε=initial relative error, r=decay rate, w_c=initial weight, $U_{i,0}$=initial interface guesses for $\forall i$, $\lambda_{\mathcal{B},i}, \lambda_{\mathcal{D},i}, \mu_{\mathcal{B},i}, \mu_{\mathcal{D},i}$ = initial set of ALM parameters for $\forall i$

Initial coarse network training:
Update θ_c to minimize $\mathcal{L}_c$ along with the ALM parameters over a few epochs

Initialize interface values: Set U_i using (3.22) with $\tau_r = 0$ and $\omega_c = 1.0$ for all i
iter = 0
while $\varepsilon > \tau$ and iter < max_iter **do**
1. Subdomain networks training:
for count = $1, \cdots, N_l$ **do**
For each i, update θ_i to minimize $\mathcal{L}_i$
Update $\lambda_{b,i}$, $\mu_{b,i}$, $\lambda_{d,i}$, and $\mu_{d,i}$ according to (3.17)
end for

2. Coarse network training:
for count = $1, \cdots, N_c$ **do**
Update θ_c to minimize $\mathcal{L}_c$
Update $\lambda_{b,c}$, and $\mu_{b,c}$ according to (3.17)
end for
$w_c \leftarrow w_c r$

3. Communication:
Evaluate $\hat{u}$ in (3.23) and compute ε
Update interface values: Update U_i according to (3.22), $\forall \mathbf{x} \in \Gamma_{ij}$ for all i
iter = iter+1
end while

In [3, 5, 7], the coarse models are constructed using plane waves defined on the subdomain edges. The analysis in [7] requires that a linear combination of the coarse basis functions provides a sufficiently accurate approximation of the functions defined in the coarse subdomain mesh, with accuracy on the order of H, the typical subdomain size. The plane waves perform better than the standard edge average functions. In contrast, our coarse model is constructed using neural networks, which offer a more flexible representation of the coarse space, while still requiring the approximation

to achieve a certain level of accuracy. This accuracy depends on the careful choice of the coarse model's depth, width, sample points, and training epochs. Special choices of the activation functions in the coarse model might improve the performance as in [30] and our numerical experiments in the next section.

4. Numerical experiments

Consider the following Helmholtz equation:

$$\begin{aligned} -\Delta u(x,y) - \kappa^2 u(x,y) &= f, \quad (x,y) \in \Omega, \\ u(x,y) &= 0, \quad (x,y) \in \partial\Omega, \end{aligned} \tag{4.1}$$

where $\Omega = [-1,1] \times [-1,1]$. For each κ, we choose $f = -(\kappa^2 - 17\pi^2)\sin(\pi x)\sin(4\pi y)$ such that (4.1) has the true solution

$$u(x,y) = \sin(\pi x)\sin(4\pi y). \tag{4.2}$$

We first explored the Helmholtz equation over the entire domain using a single neural network and selected an optimizer and learning rate. The optimizer chosen in this way is L-BFGS with a learning rate of 0.0015 since the Adam optimizer was too slow and failed to converge within the prescribed threshold.

In the one-domain exploration, we observed that the number of epochs required to achieve the tolerance varies significantly across different datasets. This makes it impractical to use a fixed number of coarse epochs as the pace of coarse model training quite differed between runs. Therefore, in the two-level methods, we adjust the pace of the coarse model to ensure that the coarse model supports the local models at comparable paces across seeds; we add 5 epochs to the current coarse epochs if the coarse error is greater than the local error, and subtract 5 epochs if the coarse error is smaller. If the local error is sufficiently close to the predetermined cutoff, the coarse network is trained using the minimum number of coarse epochs, which is set to 5. For all two-level methods, the cutoff is set to 0.007, and the number of coarse epochs for the initial training and the first outer iteration is set to 50.

In our numerical results, we use the term "outer iteration" to denote the iterations of our domain decomposition algorithms required to reach the prescribed tolerance, and "epoch" to denote the steps used to train the neural network. The tolerance for the outer iteration is set to 0.005, so the outer iteration proceeds until either the iteration count exceeds the maximum number of iterations 100, or the relative error in (3.4) becomes less than the tolerance. All numerical results were obtained by testing three distinct datasets selected using different random seeds.

The hyper-parameters for the subdomain local neural network setup, the numbers of sample points, and the training epochs are given in Table 1. In addition, we also provide the floating point operation counts (FLOPs) for subdomain local neural network evaluations, which are obtained using "FlopCountAnalysis" from "fvcore". "M" is a shorthand for 10^6. The hyper-parameters, sample points, and FLOPs for the coarse neural network are provided in Table 2. The predetermined parameters δ and $\mu_{\max}$ in (3.17) are set to 0 and 10^4, respectively. Each $\mu_{b,i}$ and $\mu_{d,i}$ is initialized to 1, while all the other parameters are initialized to zero for both overlapping and non-overlapping cases. The weights of all neural networks are initialized using Xavier normal initialization, and the biases are initialized to zero.

The input data is normalized using a uniform distribution $\mathcal{U}[-1, 1]$, which has a mean of 0.0 and a standard deviation of approximately 0.58.

We first set the wavenumber $\kappa = 1$ and test both the one-level and two-level methods. We also test our algorithms with higher wavenumbers $\kappa = 100$ and $\kappa = 1000$.

For $k = 1$, as shown in Tables 3 and 5, the one-level methods may perform poorly for the Helmholtz equation with a low wavenumber since the discrete problem is nearly elliptic and the low-frequency error should be smoothed on the coarse grid. In this case, the coarse grid correction plays an essential role, as in classical elliptic problems.

Typically, for large wavenumbers, the resulting linear system for the Helmholtz equation with classical discretizations is very ill-conditioned, making it challenging to design scalable and efficient overlapping or non-overlapping domain decomposition methods [7, 36]. However, as κ grows, the Helmholtz equation (2.1) becomes dominated by the $-\kappa^2 u$ term. Let u be the exact solution of (2.1) and recall that u_θ is the approximated solution by the neural network. The physics loss function can be approximated as

$$\mathcal{H}(u_\theta) - f = -\Delta u_\theta - \kappa^2 u_\theta - \left(-\Delta u - \kappa^2 u\right) = \Delta(u - u_\theta) + \kappa^2(u - u_\theta) \approx \kappa^2(u - u_\theta), \tag{4.3}$$

under the assumption that $\Delta(u - u_\theta)$ is bounded. Thus, for κ sufficiently large, the physics loss function (2.4) can be approximated as follows:

$$\mathcal{L}_{\mathcal{H}}(\theta) \approx \frac{\kappa^2}{N_{\mathcal{H}}} \sum_{k=1}^{N_{\mathcal{H}}} |u_\theta(\mathbf{x}_{\mathcal{H}}^{(k)}) - u(\mathbf{x}_{\mathcal{H}}^{(k)})|^2. \tag{4.4}$$

When κ is large, the term $\mathcal{L}_{\mathcal{H}}(\theta)$ gets heavily weighted and becomes the most influential one in the total loss function, while $\|\Delta(u - u_\theta)\|$ acts as a small perturbation. This encourages the neural network to focus more directly on the true solution to minimize the residuals, thereby accelerating the convergence rate. A typical example of $\|\Delta(u - \hat{u})\|/(\kappa^2\|u - \hat{u}\|)$ versus κ^2 is shown in Figure 2 to confirm this.

Table 1. Subdomain local network hyper-parameters for the overlapping/non-overlapping methods.

# Sub.	2×2	4×4	6×6	8×8	10×10	Large domain
Layers	4	4	4	4	4	4
Neurons	35	18	12	9	7	18
Inter. Points	1600	400	178	100	64	500/160
Bound. Points	800/2400	200/600	89/270	50/150	32/96	260/600
Local Param.	3921	1099	517	307	197	1099
Total Param.	15,684	17,584	18,612	19,648	19,700	17,584
Epochs	90/180	80/120	60/100	45/80	40/60	100/120
Flops	9.072M/15.120M	0.616M/1.026M	0.125M/0.21M	0.0405M/0.0675M	0.0161K/0.0269M	0.780M/0.780M

Table 2. Coarse network hyper-parameters for the overlapping/non-overlapping methods.

# Sub.	2×2	4×4	6×6	8×8	10×10	Large domain
Layers	4	4	4	4	4	4
Neurons	20	20	20	20	20	20
Inter. Points	520	560	612	640	700	4,000
Bound. Points	260	288	324	384	400	2,200
Total Param.	1341	1341	1341	1341	1341	1341
Flops	0.983M	1.068M	1.179M	1.290M	1.386M	7.812M

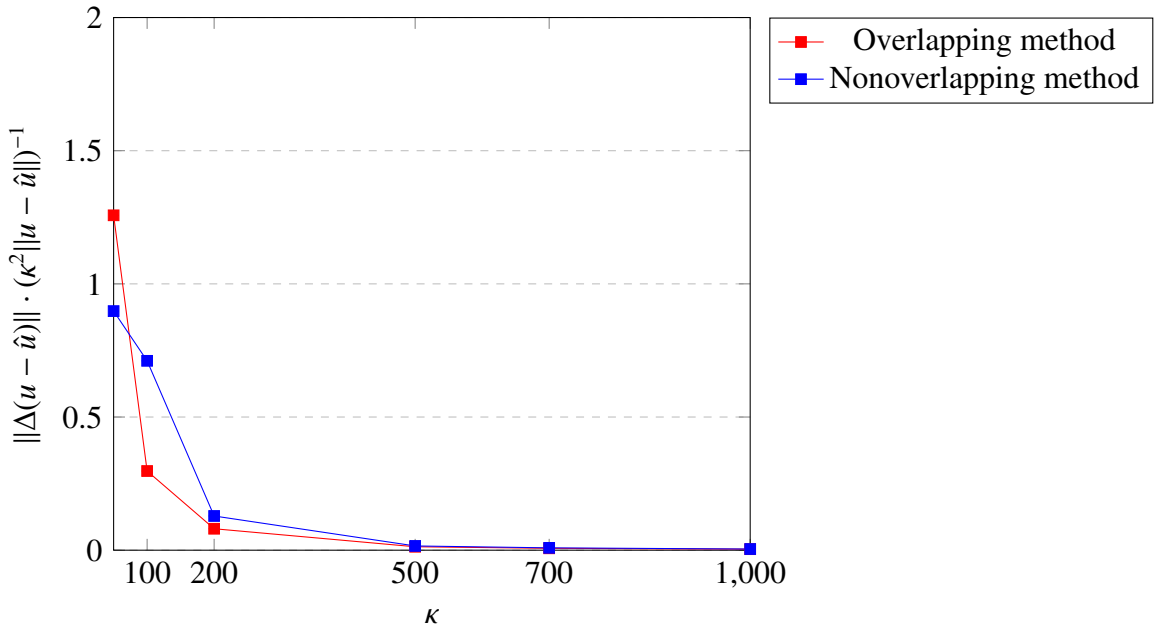

Figure 2. Behavior of $||\Delta(u - \hat{u})||/(\kappa^2||u - \hat{u}||)$ versus κ for the one-level methods after 2 outer iterations (16 subdomains).

Numerical experiments in Sections 4.1 and 4.2 also confirm that large values of κ can lead to more efficient training. This observation is also consistent with the results in [30], where deep domain decomposition methods using the PINNs required less time than the classical finite difference methods to reach comparable accuracy at $\kappa = 100$.

4.1. Overlapping methods

For the one-level methods, the initial guess, U_i^0, for the subdomain interface is chosen to be zero. We consider two types of the interface operator $\mathcal{D}_i$, defined in (3.5). One choice is $\mathcal{D}_i(u) = u_i$, returning the function values on the interface only. We call this type the "Dirichlet" interface operator. The second choice is the same as in (3.5), which returns both the function values and the normal derivatives on the interface. We call this type the "Mixed" interface operator. Due to the overlapping domain partition, the subdomain interface usually lies within the interior of the neighboring subdomains. Therefore, we

take $c_i = 0$ in (3.7), meaning that we just use the values from the neighboring subdomain interiors, which usually give a better approximation of the true solution. The relaxation parameter τ_r and the decay rate r are set to 0.0 and 0.9, respectively.

For $\kappa = 1$, we test two activation functions: the hyperbolic tangent (tanh) and the sine function for the one-level overlapping methods. We note that the sine function is closely related to the plane wave functions used in [30]. The results are reported in Table 3. As can be seen, the methods with the sine activation function perform better than those with tanh. The results using tanh for 100 subdomains are marked as >100 because most runs did not converge within the maximum number of outer iterations, making it impossible to compute an average. The "Mixed" interface operator also yields better results than the "Dirichlet" operator. However, for all one-level methods, the number of outer iterations increases as the number of subdomains grows, especially when the number of subdomains becomes large. Thus, the algorithms are not scalable.

The results for $\kappa = 100$ and $\kappa = 1000$ are reported in Table 4. For large κ, the performance of both choices of the subdomain interface operator $\mathcal{D}_i$ are comparable and we only present the results with the "Dirichlet" operator. To reach the tolerance, only a very small number of outer iterations is required, and this is independent of the number of subdomains. This aligns with our expectation.

For the two-level methods, we only use the sine activation function as it yields better results than tanh in the one-level methods. An adaptive training strategy is used for the coarse model. The coarse model provides the local models with interface function values for the local subdomain solvers with the "Dirichlet" operator and both the function values and the normal derivatives for the local subdomain solvers with the "Mixed" operator instead. The results are reported in Table 3 for $\kappa = 1$ and Table 4 for $\kappa = 100$ and 1000. For $\kappa = 1$, the results show that the additional normal derivative information improves the performance of the two-level methods. Even as the number of the subdomains increases, the number of outer iterations required to reach the tolerance stays nearly constant for both choices of the subdomain interface operator. Thus, the two-level algorithms are scalable. But, the variations for the "Mixed" subdomain local models are smaller than those with the "Dirichlet" local models. The results for $\kappa = 1$ are shown in Figure 3.

For $\kappa = 100$ and 1000, as expected, the two-level methods perform similarly. The number of outer iterations required to reach the tolerance is quite small. The results for $\kappa = 100$ and 1000 are shown in Figure 4. In these cases, the two-level methods are not necessary since the local subdomain solvers alone already provide scalable methods.

Table 3. The numbers of outer iterations for overlapping methods with $\kappa = 1$.

Method	One level		One level		Two level	
Activation function	tanh		sine		sine	
Interface Operator	Dirichlet	Mixed	Dirichlet	Mixed	Dirichlet	Mixed
2×2	10.3 ± 1.5	10.3 ± 1.5	10.5 ± 1.5	10.7 ± 1.5	7.7 ± 2.9	6.0 ± 1.0
4×4	20.0 ± 1.0	22.0 ± 2.6	19.7 ± 1.2	22.3 ± 1.5	26.3 ± 6.5	14.3 ± 3.5
6×6	20.7 ± 1.5	20.3 ± 3.5	17.7 ± 2.1	19.3 ± 1.5	22.7 ± 7.6	22.0 ± 2.6
8×8	27.3 ± 7.3	25.0 ± 3.6	21.0 ± 5.3	17.6 ± 2.1	27.7 ± 6.4	21.0 ± 7.9
10×10	> 100	> 100	32.0 ± 2.0	29.0 ± 3.9	26.3 ± 13.1	23.3 ± 0.6

Table 4. The numbers of outer iterations for overlapping methods with high κ. The "Dirichlet" interface operator and the sine activation function are used.

Method	One level		Two level	
κ	100	1000	100	1000
2×2	4.3 ± 2.3	3.3 ± 0.6	3.7 ± 0.6	3.0 ± 1.0
4×4	2.7 ± 0.6	2.3 ± 0.6	3.0 ± 0.0	2.7 ± 0.6
6×6	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.0
8×8	4.0 ± 0.0	4.3 ± 0.6	4.0 ± 0.0	4.7 ± 0.6
10×10	4.3 ± 0.6	5.0 ± 0.0	4.7 ± 0.6	5.3 ± 2.9

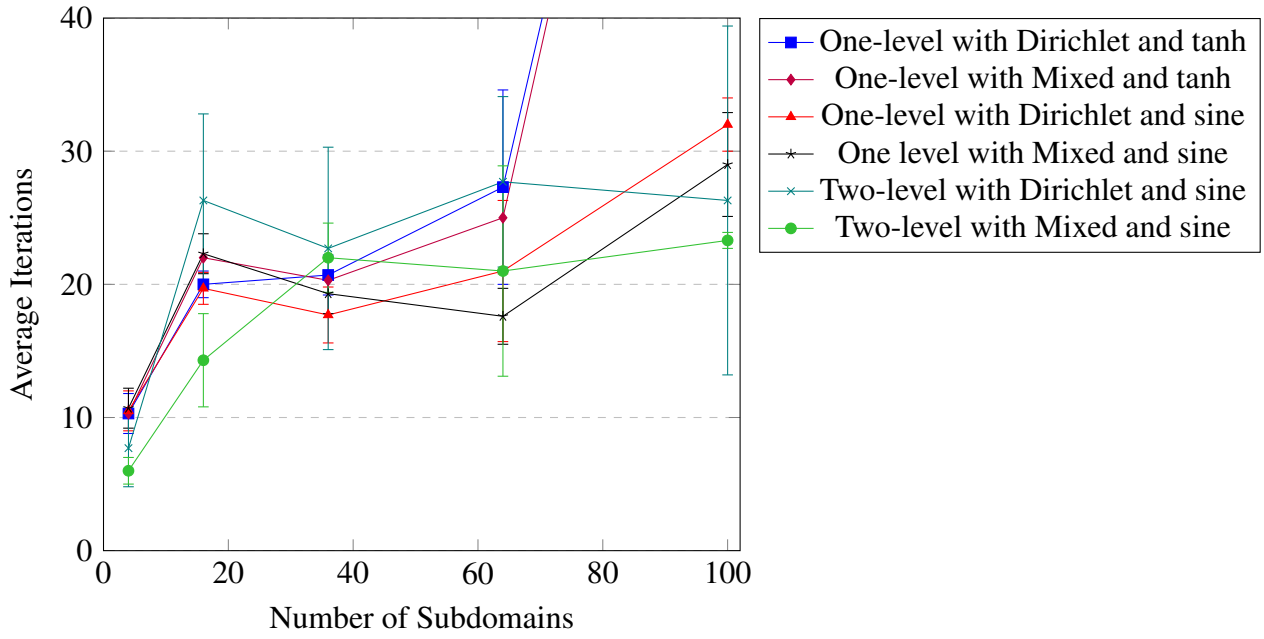

Figure 3. Overlapping methods with $\kappa = 1$.

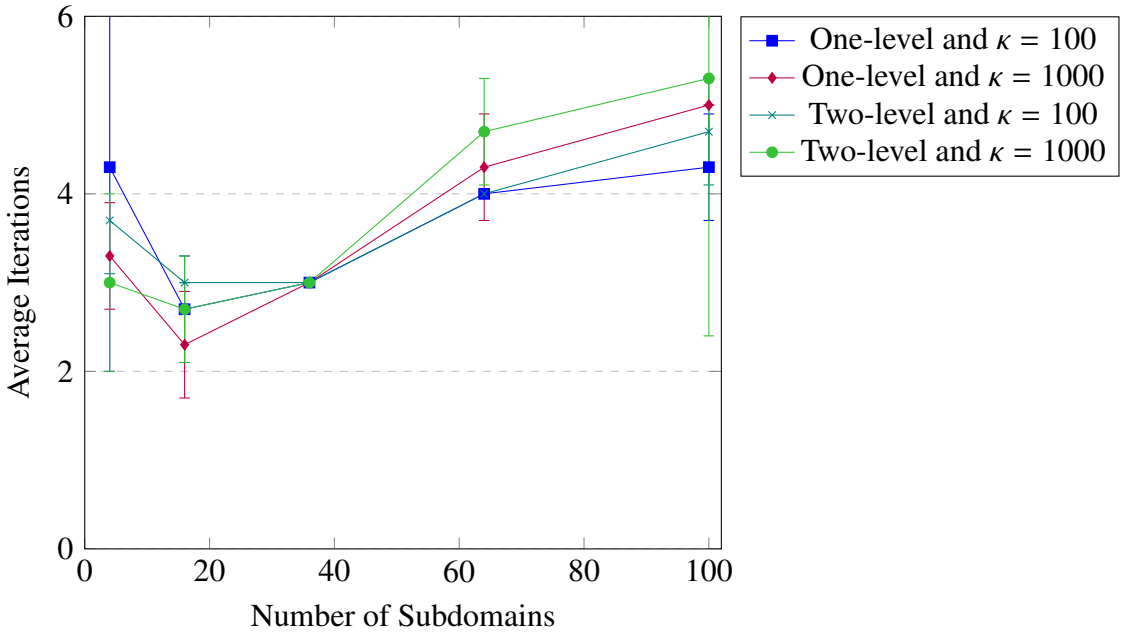

Figure 4. Overlapping methods with $\kappa = 100, 1000$. The "Dirichlet" interface operator and the sine activation function are used.

4.2. Non-overlapping methods

In our non-overlapping methods, a sine function is adopted as the activation function, and the relaxation parameter τ_r and the decay factor r are set to 0.0 and 0.9, respectively. The weight c_i in (3.7) is set to 0.5. It usually depends on the PDE coefficients, which are constant in (2.1). Due to the non-overlapping partition, only the "Mixed" subdomain interface operator is used for our subdomain local models.

As in the overlapping methods, the initial guess U_i^0 is set to zero for the one-level methods. The numerical results for the one-level methods are presented in Table 5 for $\kappa = 1$ and Table 6 for $\kappa = 100, 1000$. Table 5 shows that none of the one-level algorithms reach the tolerance within the maximum number of iterations when $\kappa = 1$, except in the case with 4 subdomains. This is because our initial interface guess $U_i^0 = 0$ coincides with the exact solution values. On the contrary, when $\kappa = 100, 1000$, the one-level algorithm results in fast convergence within only a few iterations.

For the two-level methods, the coarse network is first trained over 50 epochs without the fitting loss in (3.21) ($w_C = 0$), after which the outcomes provide the initial guess U_i^0 for the subdomains. This initial guess is intended to prevent the local networks from starting their training with poor initialization. After the initial training, w_C is increased from 0 to 0.1 over the coarse training. In contrast, over a large domain, this w_C is increased to 1.0 from 0.0 after the initial training. These values were selected experimentally, but they exhibit a consistent trend. When the coarse component's error tends to drop faster than the local models', a lower weight offers better performance. While when the coarse error consistently exceeds the local errors, a higher weight is more effective, enabling the local models to contribute more detailed corrections. Similar to the overlapping decomposition, an adaptive epoch strategy is used to train the coarse model, and the maximum number of outer iterations is set to 100. For the two-level non-overlapping method, subdomain models always use the "Mixed"

interface operator as in the one-level method. In contrast, the coarse model uses either the "Dirichlet^C" or "Mixed^C" operator to indicate the information communication between the subdomain local and coarse models. In the two-level methods with "Dirichlet^C", the coarse model only provides the function values to the local models. As a result, each subdomain exchanges normal derivative information only with its immediate neighbors without any assistance from the coarse model. This configuration was also adopted in our previous work [35]. On the other hand, in the two-level PECANN with "Mixed^C", each subdomain receives both function values and normal derivative information from its direct neighbors as well as from the coarse model.

The results for the two-level methods with $\kappa = 1$ are presented in Table 5. One remarkable finding from these numerical experiments is that the algorithm fails to converge when the subdomain interface information excludes normal derivative estimates from the coarse model. This behavior is illustrated by the two-level "Dirichlet^C" method in Table 5, where the algorithm does not converge within the maximum iteration counts, even for a small number of subdomains. On the other hand, the "Mixed^C" operator results in an almost constant number of outer iterations as the number of subdomains increases. Therefore, the two-level method with the "Mixed^C" coarse component is scalable.

In general, when $\kappa = 100$ and 1000, regardless of the number of levels and the communication between the coarse and local models, all the algorithms converged quickly within a few iterations as in Table 6.

The results for $\kappa = 1$ and $\kappa = 100, 1000$ are also presented in Figures 5 and 6, respectively.

Table 5. The number of outer iterations for non-overlapping methods with $\kappa = 1$. Dirichlet^C: Only the function values are communicated between the coarse and local models. Mixed^C: The function values and normal derivatives are communicated between the coarse and local models.

Method	One level	Two level	
Coarse	–	Dirichlet^C	Mixed^C
2×2	4.0 ± 0.0	> 100	5.4 ± 2.6
4×4	> 100	> 100	8.6 ± 2.7
6×6	> 100	> 100	9.8 ± 6.7
8×8	> 100	> 100	20.8 ± 7.5
10×10	> 100	> 100	14.6 ± 4.7

Table 6. Non-overlapping methods with high κ.

Method	One level		Two level	
κ	100	1,000	100	1,000
2×2	1.7 ± 0.6	1.3 ± 0.6	1.7 ± 0.6	1.7 ± 0.6
4×4	2.0 ± 0.0	2.0 ± 0.0	1.0 ± 0.0	1.7 ± 0.6
6×6	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
8×8	2.3 ± 0.6	3.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
10×10	2.0 ± 0.0	2.0 ± 0.0	2.3 ± 0.6	3.0 ± 0.0

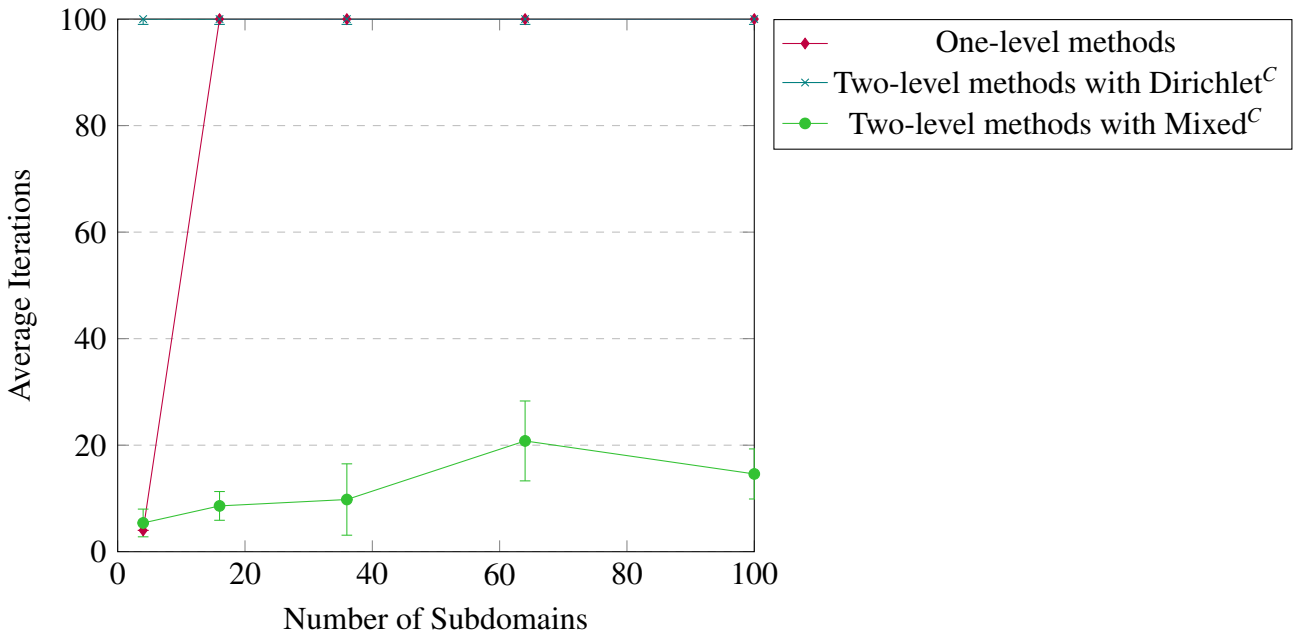

Figure 5. Non-overlapping methods with $\kappa = 1$. Dirichlet^C: Only the function values are communicated between the coarse and local models. Mixed^C: The function values and normal derivatives are communicated between the coarse and local models.

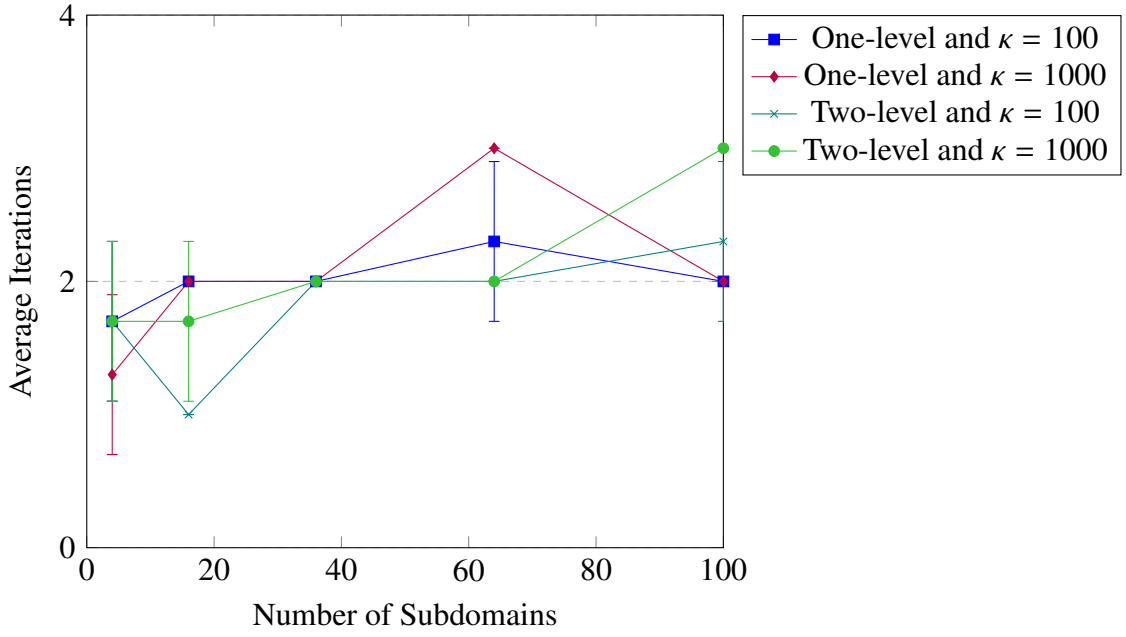

Figure 6. Non-overlapping methods with $\kappa = 100$ and 1000.

4.3. Computational cost analysis and a large domain example

In this section, we evaluate the efficiency of the proposed algorithms in terms of the number of floating-point operations (FLOPs) required for convergence. Additionally, we demonstrate their effectiveness by applying them to the Helmholtz equation on a large computational domain.

We first measure the FLOPs for the evaluations of both subdomain local and coarse neural networks using "FlopCountAnalysis" from "fvcore". The results are reported in Tables 1 and 2, respectively.

For the one-level methods, the number of epochs used to train the subdomain local models is listed in Table 1. In each epoch, the LBFGS optimizer can perform up to 25 function evaluations, which are used in most cases. For simplicity, the total FLOPs for the subdomain local models are approximated as the product of the network FLOPs, the fixed number of epochs, and the number of outer iterations. These estimates serve as a reasonable indicator of the overall computational cost associated with the subdomain local models. The two-level methods, in addition to the FLOPs from the subdomain local models, are estimated in the same way as for the one-level methods.

We also account for the FLOPs required by the coarse models. The total number of epochs used to train the coarse models in different algorithms are listed in Table 7 for $\kappa = 1$. The total FLOPs for the coarse models are measured as the product of the network FLOPs and the total number of training epochs. For $\kappa = 100$ and 1000, the one-level methods perform well and the coarse component is not required.

The FLOP results for different methods with $\kappa = 1$ are reported in Table 8. For the one-level overlapping methods, the FLOPs for training the subdomain local models decrease rapidly as the number of subdomains increases due to the sharp reduction in the number of neurons and training samples used in each local model (Table 1). Although the number of outer iterations increases, the overall FLOPs still drop quickly, as shown in Table 8. For all methods, the computational cost with 64 subdomains is less than 1% of that with 4 subdomains. However, when the number of subdomains is 100, the methods using the activation function tanh fail to reach the tolerance within 100 outer iterations. Similarly, the one-level non-overlapping methods fail to converge within 100 outer iterations for all tested subdomain numbers. For the two-level overlapping methods, when the number of subdomains is small, the FLOPs are dominated by the subdomain local components, and the FLOPs of the coarse component are negligible. The two-level methods are highly efficient. In particular, the two-level methods with the "Mixed" interface operators require only about half as many FLOPs as the one-level methods for the 4 subdomain case. When the number of subdomains becomes large, however, the local FLOPs decrease rapidly, and the coarse component FLOPs dominate. In this regime, a more efficient multi-level coarse component is needed to further improve overall efficiency. For the two-level non-overlapping methods, the FLOPs for the coarse and local models remain comparable even with 36 subdomains. This is because more sample points are used in the subdomain local models (Table 1), and the one-level methods have difficulty converging. As a result, the two-level methods are more favorable. However, when the number of subdomains exceeds 36, the FLOPs become dominated by the coarse component, and more efficient coarse solvers are required.

For large $\kappa = 100, 1000$, the one-level methods are the most efficient since the coarse model is not needed to accelerate convergence. We only present the FLOPs for the one-level methods with $\kappa = 100$ in Table 9. The FLOPs for training the subdomain local models decrease rapidly as the number of subdomains increases, making the algorithms extremely efficient.

At the end of this section, for $\kappa = 100$, we test our methods on a large domain $[-10, 10] \times [-1, 1]$ with the exact solution defined in (4.2). Our overlapping methods use the "Dirichlet" interface operator and two-level non-overlapping methods use "Mixed^C" coarse choice. The sine activation function is used for all methods. We partition the domain into 10×4 subdomains. The hyper-parameters for subdomain local neural networks and the coarse networks are presented in Tables 1 and 2, respectively. The results are reported in Table 10. Similar to previous experiments, all tested algorithms perform well; however, the coarse model (a single neural network on the whole domain) alone on this large domain fails to reach the tolerance within 5,000 epochs.

Table 7. Total number of coarse epochs in two-level methods for $\kappa = 1$.

Method	Overlapping methods		Non-overlapping methods
Interface operator	Dirichlet	Mixed	Mixed
2×2	340.0 ± 113.0	261.7 ± 62.5	305.0 ± 124.4
4×4	905.0 ± 55.7	526.7 ± 167.5	426.0 ± 154.5
6×6	578.3 ± 134.3	588.3 ± 135.0	374.0 ± 175.4
8×8	440.0 ± 105.4	356.7 ± 75.2	403.0 ± 38.0
10×10	621.7 ± 285.7	390.0 ± 82.3	415.0 ± 58.5

Table 8. FLOPs (M) to reach tolerance with the one-level and two-level methods with $\kappa = 1$. "D": The "Dirichlet" interface operator. "M": The "Mixed" interface operator.

Overlapping methods						
# Sub.		2×2	4×4	6×6	8×8	10×10
One-level	D+tanh	8409.7 ± 1224.7	985.6 ± 49.3	155.2 ± 11.2	49.8 ± 13.1	-
One-level	M+tanh	8409.7 ± 1224.7	1084.2 ± 128.1	152.2 ± 26.2	45.6 ± 6.6	-
One-level	D+sine	8573.0 ± 1224.7	970.8 ± 59.1	132.8 ± 15.8	40.5 ± 9.7	20.6 ± 1.3
One-level	M+sine	8736.3 ± 1224.7	1098.9 ± 73.9	144.7 ± 11.2	32.1 ± 3.8	18.7 ± 2.5
Coares	D +sine	334.2 ± 111.1	966.5 ± 59.5	681.8 ± 158.3	567.6 ± 136.0	861.7 ± 396.0
Local	D+sine	6286.9 ± 2367.8	1296.1 ± 320.3	170.3 ± 57.0	50.5 ± 11.7	17.0 ± 8.5
Coarse	M +sine	257.3 ± 61.4	562.5 ± 178.9	693.6 ± 159.2	460.1 ± 97.0	540.5 ± 114.1
Local	M+sine	4898.9 ± 816.5	704.7 ± 172.5	165.0 ± 19.5	38.3 ± 14.4	15.0 ± 0.4
Non-overlapping methods						
One-level		–	–	–	–	–
Coarse	M+sine	299.8 ± 122.3	455.0 ± 165.0	440.9 ± 206.8	519.9 ± 49.0	575.2 ± 81.1
Local	M+sine	14697.0 ± 7076.2	1059 ± 332.4	206 ± 140.7	112 ± 40.5	24 ± 7.6

Table 9. FLOPs (M) to reach tolerance with the one-level methods with $\kappa = 100$. "D": The "Dirichlet" interface operator. "M": The "Mixed" interface operator.

Overlapping methods						
# Sub.		2×2	4×4	6×6	8×8	10×10
One-level	D+sine	3510.9 ± 1877.9	133.1 ± 29.6	22.5 ± 0.0	7.3 ± 0.0	2.8 ± 0.4
Non-overlapping methods						
One-level	M+sine	4626.7 ± 1633.0	246.2 ± 0.0	42.0 ± 0.0	12.4 ± 0.0	3.2 ± 0.0

Table 10. The results over a large domain with $\kappa = 100$.

Method	Overlapping methods		Non-overlapping methods	
	one-level	two-level	one-level	two-level
# of outer iteration	16.3 ± 3.1	16.3 ± 2.5	37.0 ± 5.0	12.7 ± 0.6
Total coares epochs	–	1843.3 ± 328.7	–	1120.0 ± 95.4

5. Conclusions

In this paper, we propose both overlapping and non-overlapping domain-decomposition-enhanced PECANN methods for the Helmholtz equation, formulating one-level and two-level approaches. In the one-level method, we investigate whether exchanging only function values or additionally including normal derivative information at the subdomain interfaces leads to better performance in the overlapping methods. Our numerical experiments further show that the sine activation function outperforms the tanh function in this setting. In the two-level method, for the overlapping case, the two interface options were compared at the subdomain level, while the coarse model consistently provides either function values or both function values and normal derivative information according to the needs from the subdomain interface operators. For the non-overlapping case, the local models exchange both the function values and normal derivatives, whereas the coarse model provides either only function values or both. In particular, the normal derivative information helps stabilize the overlapping algorithms and improve their performance for a small wavenumber. For the non-overlapping method, adding normal derivative information from the coarse model is necessary as the algorithm fails to achieve scalability when the coarse model provides only function values. Finally, provided that the number of outer iterations does not grow too rapidly, the one-level overlapping methods remain highly efficient, even for larger computational domains.

In contrast to approaches that train a single global neural network to approximate PDE solutions, our methods substantially reduce the overall computational cost by training subdomain-local neural networks in parallel on different processors. This strategy makes the algorithms particularly effective for large-scale computations, such as those on very large domains, where a single neural network often encounters convergence difficulties. Furthermore, unlike one-level domain decomposition methods, our two-level framework ensures that the number of outer iterations remains nearly independent of the number of subdomains, a property that is essential for achieving scalability in large-scale parallel

computations. These coarse components are particularly crucial for the non-overlapping methods.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the HPC facilities operated by the Center for Research Computing at the University of Kansas.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. I. M. Babuška, S. A. Sauter, Is the pollution effect of the FEM avoidable for the Helmholtz equation considering high wave numbers? *SIAM J. Numer. Anal.*, **34** (1997), 2392–2423. https://doi.org/10.1137/S0036142994269186
2. W. Chen, Y. Liu, X. Xu, A robust domain decomposition method for the Helmholtz equation with high wave number, *ESAIM. Math. Model. Numer. Anal.*, **50** (2016), 921–944. https://doi.org/10.1051/m2an/2015058
3. C. Farhat, A. Macedo, M. Lesoinne, A two-level domain decomposition method for the iterative solution of high-frequency exterior Helmholtz problems, *Numer. Math.*, **85** (2000), 283–308. https://doi.org/10.1007/PL00005389
4. M. J. Gander, F. Magoulès, F. Nataf, Optimized Schwarz methods without overlap for the Helmholtz equation, *SIAM J. Sci. Comput.*, **24** (2002), 38–60. https://doi.org/10.1137/S1064827501387012
5. C. Farhat, J. Li, An iterative domain decomposition method for the solution of a class of indefinite problems in computational structural dynamics, *Appl. Numer. Math.*, **54** (2005), 150–166. https://doi.org/10.1016/j.apnum.2004.09.021
6. M. J. Gander, L. Halpern, F. Magoulès, An optimized Schwarz method with two-sided Robin transmission conditions for the Helmholtz equation, *Int. J. Numer. Methods Fluids*, **55** (2007), 163–175. https://doi.org/10.1002/fld.1433
7. J. Li, X. Tu, Convergence analysis of a balancing domain decomposition method for solving a class of indefinite linear systems, *Numer. Linear Algebra Appl.*, **16** (2009), 745–773. https://doi.org/10.1002/nla.639
8. M. J. Gander, H. Zhang, Optimized Schwarz methods with overlap for the Helmholtz equation, *SIAM J. Sci. Comput.*, **38** (2016), A3195–A3219. https://doi.org/10.1137/15M1021659

9. P. Lu, X. Xu, A robust multilevel preconditioner based on a domain decomposition method for the Helmholtz equation, *J. Sci. Comput.*, **81** (2019), 291–311. https://doi.org/10.1007/s10915-019-01015-z
10. Y. Liu, X. Xu, An optimized Schwarz method with relaxation for the Helmholtz equation: the negative impact of overlap, *ESAIM. Math. Model. Numer. Anal.*, **53** (2019), 249–268. https://doi.org/10.1051/m2an/2018061
11. K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, *Neural Networks*, **2** (1989), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8
12. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, **378** (2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045
13. S. Cai, Z. Mao, Z. Wang, M. Yin, G. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: a review, *Acta Mech. Sin.*, **37** (2021), 1727–1738. https://doi.org/10.1007/s10409-021-01148-1
14. S. Cuomo, V. Schiano D. Cola, F. Giampaolo, G. Rozza, M. Raissi, et al., Scientific machine learning through physics-informed neural networks: where we are and what's next, *J. Sci. Comput.*, **92** (2022), 88. https://doi.org/10.1007/s10915-022-01939-z
15. J. Su, G. Tao, R. Li, S. Zhang, Physics-driven deep learning methods and numerically intractable "bad" Jaulent–Miodek equation, *Chaos*, **35** (2025), 063101. https://doi.org/10.1063/5.0264041
16. P. Lan, J. Su, S. Zhang, Surrogate modeling for unsaturated infiltration via the physics and equality-constrained artificial neural networks, *J. Rock Mech. Geotech. Eng.*, **16** (2024), 2282–2295. https://doi.org/10.1016/j.jrmge.2023.09.014
17. S. Zhang, P. Lan, J. Su, Wave-packet behaviors of the defocusing nonlinear Schrodinger equation based on the modified physics-informed neural networks, *Chaos*, **31** (2021), 113107. https://doi.org/10.1063/5.0067260
18. J. Su, S. Zhang, P. Lan, X. Chen, Explorations of certain nonlinear waves of the Boussinesq and Camassa–Holm equations using physics-informed neural networks, *Proc. R. Soc. A*, **480** (2024), 20230580. https://doi.org/10.1098/rspa.2023.0580
19. S. Zhang, J. Deng, X. Li, Z. Zhao, J. Wu, W. Li, et al., Solving the one dimensional vertical suspended sediment mixing equation with arbitrary eddy diffusivity profiles using temporal normalized physics-informed neural networks, *Phys. Fluids*, **36** (2024), 017132. https://doi.org/10.1063/5.0179223
20. X. Li, J. Wu, X. Tai, J. Xu, Y. Wang, Solving a class of multi-scale elliptic PDEs by Fourier-based mixed physics informed neural networks, *J. Comput. Phys.*, **508** (2024), 113012. https://doi.org/10.1016/j.jcp.2024.113012
21. X. Li, J. Wu, Y. Huang, Z. Ding, X. Tai, L. Liu,et al., Fourier-feature induced physics informed randomized neural network method to solve the biharmonic equation, *J. Comput. Appl. Math.*, **468** (2025), 116635. https://doi.org/10.1016/j.cam.2025.116635

22. S. Basir, I. Senocak, Physics and equality constrained artificial neural networks: application to forward and inverse problems with multi-fidelity data fusion, *J. Comput. Phys.*, **463** (2022), 111301. https://doi.org/10.1016/j.jcp.2022.111301

23. M. J. D. Powell, A method for nonlinear constraints in minimization problems, *Optimization*, 1969.

24. A. Quarteroni, A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford University Press, New York, 1999. https://doi.org/10.1093/oso/9780198501787.001.0001

25. A. Toselli, O. Widlund, *Domain Decomposition Methods - Algorithms and Theory*, Springer Verlag, Berlin-Heidelberg-New York, 2005.

26. A. D. Jagtap, E. Kharazmi, G. E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems, *Comput. Methods Appl. Mech. Eng.*, **365** (2020), 113028. https://doi.org/10.1016/j.cma.2020.113028

27. A. D. Jagtap, G. E. Karniadakis, Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, *Commun. Comput. Phys.*, **28** (2020), 2002–2041. https://doi.org/10.4208/cicp.OA-2020-0164

28. K. Shukla, A. D. Jagtap, G. Karniadakis, Parallel physics-informed neural networks via domain decomposition, *J. Comput. Phys.*, **447** (2021), 110683. https://doi.org/10.1016/j.jcp.2021.110683

29. W. Li, X. Xiang, Y. Xu, Deep domain decomposition method: Elliptic problems, *Proc. Mach. Learn. Res.*, **107** (2020), 269–286. Available from: https://proceedings.mlr.press/v107/li20a.html.

30. W. Li, Z. Wang, T. Cui, Y. Xu, X. Xiang, Deep domain decomposition methods: Helmholtz equation, *Adv. Appl. Math. Mech.*, **15** (2023), 118–138. https://doi.org/10.4208/aamm.OA-2021-0305

31. H. J. Yang, H. H. Kim, Iterative algorithms for partitioned neural network approximation to partial differential equations, *Comput. Math. Appl.*, **170** (2024), 237–259. https://doi.org/10.1016/j.camwa.2024.07.007

32. D. Jang, K. Kim, H. H. Kim, Partitioned neural network approximation for partial differential equations enhanced with Lagrange multipliers and localized loss functions, *Comput. Methods Appl. Mech. Eng.*, **429** (2024), 117168. https://doi.org/10.1016/j.cma.2024.117168

33. V. Dolean, A. Heinlein, S. Mishra, B. Moseley, Multilevel domain decomposition-based architectures for physics-informed neural networks, *Comput. Methods Appl. Mech. Eng.*, **429** (2024), 117116. https://doi.org/10.1016/j.cma.2024.117116

34. Q. Hu, S. Basir, I. Senocak, Non-overlapping Schwarz-type domain decomposition method for physics and equality constrained artificial neural networks, *Comput. Methods Appl. Mech. Eng.*, **436** (2025), 117706. https://doi.org/10.1016/j.cma.2024.117706

35. E. Olson, S. Won, X. Tu, Scalable two-level domain decomposition methods for the physics and equality constrained neural networks, *Comput. Methods Appl. Mech. Eng.*, In press.

36. M. J. Gander, H. Zhang, A class of iterative solvers for the Helmholtz equation: factorizations, sweeping preconditioners, source transfer, single layer potentials, polarized traces, and optimized Schwarz methods, *SIAM Rev.*, **61** (2019), 3–76. https://doi.org/10.1137/16M109781X

37. S. Basir, I. Senocak, An adaptive augmented Lagrangian method for training physics and equality constrained artificial neural networks, preprint, arXiv:2306.04904.

38. V. Mercier, S. Gratton, P. Boudier, A coarse space acceleration of deep-DDM, preprint, arXiv:2112.03732.

AIMS Press