Advances in Applied Mathematics and Mechanics
Adv. Appl. Math. Mech., Vol. **15**, No. 1, pp. 118-138

DOI: 10.4208/aamm.OA-2021-0305
February 2023

Deep Domain Decomposition Methods: Helmholtz Equation

Wuyang Li[1,3], Ziming Wang[2,4], Tao Cui[2,4], Yingxiang Xu[1] and Xueshuang Xiang[3,*]

[1] *School of Mathematics and Statistics, Jilin National Applied Mathematics Center at NENU, Northeast Normal University, Changchun, Jilin 130024, China*
[2] *NCMIS, LSEC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China*
[3] *Qian Xuesen Laboratory of Space Technology, China Academy of Space Technology, Beijing 100094, China*
[4] *School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China*

Received 27 September 2021; Accepted (in revised version) 18 March 2022

Abstract. This paper proposes a deep-learning-based Robin-Robin domain decomposition method (DeepDDM) for Helmholtz equations. We first present the plane wave activation-based neural network (PWNN), which is more efficient for solving Helmholtz equations with constant coefficients and wavenumber k than finite difference methods (FDM). On this basis, we use PWNN to discretize the subproblems divided by domain decomposition methods (DDM), which is the main idea of DeepDDM. This paper will investigate the number of iterations of using DeepDDM for continuous and discontinuous Helmholtz equations. The results demonstrate that: DeepDDM exhibits behaviors consistent with conventional robust FDM-based domain decomposition method (FDM-DDM) under the same Robin parameters, i.e., the number of iterations by DeepDDM is almost the same as that of FDM-DDM. By choosing suitable Robin parameters on different subdomains, the convergence rate is almost constant with the rise of wavenumber in both continuous and discontinuous cases. The performance of DeepDDM on Helmholtz equations may provide new insights for improving the PDE solver by deep learning.

AMS subject classifications: 35Q68, 65N55, 78A48

Key words: Helmholtz equation, deep learning, domain decomposition method, plane wave method.

*Corresponding author.
Emails: liwy648@nenu.edu.cn (W. Li), wangziming@lsec.cc.ac.cn (Z. Wang), tcui@lsec.cc.ac.cn (T. Cui), yxxu@nenu.edu.cn (Y. Xu), xiangxueshuang@qxslab.cn (X. Xiang)

1 Introduction

The Helmholtz equation, the main research object in this paper, is a kind of important partial differential equation (PDE) in scientific computing and industrial applications. It can be regarded as a time-independent form of the wave equation, which has important applications in acoustics and describes the propagation of waves. Besides acoustic waves, it is also used to describe electromagnetics because it can be reduced from Maxwell's equations [26]. Moreover, in the fields of elasticity, quantum mechanics and geophysics, we also encounter different forms of Helmholtz equations.

We assume that there is a bounded Lipschitz domain $\Omega\subset\mathbb{R}^d$, $d=2,3$. In addition to the true physical part, the domain Ω may also contain artificial layers, for example, representing perfectly matched layers [4]. A general Helmholtz equation is given in the following

$$\mathcal{L}u:=-\nabla^T(\alpha\nabla u)-\frac{\omega^2}{\kappa}u=f \qquad \text{in } \Omega, \tag{1.1a}$$

$$\mathcal{B}u=g \qquad \text{on } \partial\Omega, \tag{1.1b}$$

where $\omega\in\mathbb{C}$ and the coefficient matrix α, the scale field κ and the source term f are all given complex-valued functions. The boundary condition $\mathcal{B}u=g$ can be one or a combination of the following cases,

$$u=g_D \qquad \text{on } \Gamma_D, \tag{1.2a}$$

$$\mathbf{n}^T(\alpha\nabla u)+p_0u=g_R \qquad \text{on } \Gamma_R, \tag{1.2b}$$

$$\mathbf{n}^T(\alpha\nabla u)+p_0u+p_1\mathbf{n}^T(\alpha\nabla_S u)-\nabla_S^T(q_1\Pi_S(\alpha\mathbf{n})u+p_2\alpha\nabla_S u)=g_V \qquad \text{on } \Gamma_V, \tag{1.2c}$$

where $\mathbf{n}$ is the unit outer normal vector, ∇_S is the surface gradient, p_0, p_1, p_2, q_1 are complex-valued functions and Π_S is the orthogonal projection onto the tangential plane of the surface. In order to simplify the notations and facilitate discussion, we will use a simpler Helmholtz equation version that usually is seen in other papers.

We look for the numerical solution of the heterogeneous Helmholtz equation as follows,

$$-\Delta u-k^2(x)u=f \qquad \text{in } \Omega, \tag{1.3a}$$

$$\frac{\partial u}{\partial\mathbf{n}}+\mathbf{i}k(x)u=g_a \qquad \text{on } \partial\Omega, \tag{1.3b}$$

$$[u]=u^+-u^-=g_b \qquad \text{on } \Gamma, \tag{1.3c}$$

$$\left[\frac{\partial u}{\partial\mathbf{n}}\right]=\left(\frac{\partial u}{\partial\mathbf{n}}\right)^+-\left(\frac{\partial u}{\partial\mathbf{n}}\right)^-=g_c \qquad \text{on } \Gamma, \tag{1.3d}$$

where $\mathbf{n}$ is the unit outer normal vector on the boundary $\partial\Omega$ or the interface Γ, $k(x)>0$ is the wavenumber. In practice, $k(x)$ would be the continuous variable coefficient or piecewise constants, not just a constant. The two formulas (1.3c) and (1.3d) describe jump conditions of u and $\frac{\partial u}{\partial\mathbf{n}}$ on the interface Γ. Actually, the numerical solution of the Helmholtz

equation has always been a hot topic in the field of computational mathematics. When classical iterative methods are used to solve for the Helmholtz equation, especially for a large wave number problem, we always suffer some difficulties, for example, pollution effect [2]. In [13], Ernst et al. has carried out a series of researches on classical iterative methods solving for Helmholtz problems. Domain decomposition methods (DDM), as an important kind of iterative method, lose also its effectiveness and efficiency for the Helmholtz equation, if it is not modified properly.

Following our previous article [9, 24], this paper continues to explore the use of deep learning to improve the efficiency of traditional DDM, and to improve the difficult situation of traditional methods when encountering large wave number Helmholtz problems. The deep learning-based domain decomposition method (DeepDDM) integrates the spirit of both deep learning and domain decomposition and inherits the advantages of the two methods. DeepDDM will exchange the subproblem information across the interface in DDM by adjusting the boundary term for solving each subproblem by deep learning. For the Helmholtz equation, a plane wave-based neural network (PWNN) with one hidden layer is very effective in solving the Helmholtz equation with constant wave number [9]. Thus we use PWNN to solve each subproblem in DeepDDM. Benefiting from the simple implementation and mesh-free strategy of using deep learning for PDE, DeepDDM will simplify the implementation of DDM and make DDM more flexible for complex PDE, e.g., those with complex interfaces in the computational domain.

In this paper, a robust DDM for solving the Helmholtz equation recently proposed by Chen et al. in [6] is used as an example to present the DeepDDM algorithm. This paper mainly has the following contributions.

(i) We present the DeepDDM method which uses PWNN to discretize the subproblems divided by the robust DDM [6] for solving Helmholtz equations, since PWNN can solve Helmholtz equations with constant wave number very efficiently [9].

(ii) Many numerical experiments are carried out to illustrate the effectiveness and efficiency of the DeepDDM algorithm. Even for the Helmholtz problem with a large wave number, the DeepDDM algorithm can still converge quickly and has good accuracy. In particular, when the wave number increases, we can keep the iteration number of DeepDDM almost unchanged by increasing the hidden unit of PWNN and training data. This result, the robustness of DeepDDM, coincides with one of the results in [6].

(iii) The number of iterations of DeepDDM is almost the same as that of using finite difference method (FDM) to solve each subproblem in the robust DDM, denote it by FDM-DDM. For large wave number problems, PWNN is faster than FDM [9], which indicates that DeepDDM is a more practical method than FDM-DDM.

(iv) Both PWNN [9] and the robust DDM [6] only consider Helmholtz equations with constant wave number. By combining these two methods, the results of our exper-

iments show that DeepDDM can deal with the problem with piecewise constant wavenumber well.

The outline of this paper is as follows. In Section 2, we briefly review the DDM algorithm designed in [6]. Focusing on a given Helmholtz equation, we introduce the framework of PWNN for solving the Helmholtz equation and the corresponding loss function in Section 3. In Section 4, the profile of the DeepDDM algorithm for solving the Helmholtz equation is given, which is followed by numerical experiments on Helmholtz equations with constant and piecewise constant wave number in Section 5. Finally, we will conclude this paper and present some directions for future work in Section 6.

2 Domain decomposition methods for Helmholtz equation

The indefiniteness of the Helmholtz equation causes that the classical Schwarz method with Dirichlet transmission conditions fails to converge even if overlapping is used. The convergence of the domain decomposition method was proved after the first-order absorbing transmission condition was introduced in Bruno Despres's Ph.D. thesis. In order to get a faster convergence rate, a new under-relaxed algorithm was introduced by Benamou and Despres in [3]. Both theoretical and numerical results make it clear the choice of relaxation parameters has a great influence on the convergence efficiency of the algorithm. Optimized Schwarz methods looking for better parameters were studied to accelerate convergence in [17,18]. Recently, a robust Robin-Robin domain decomposition method and its convergence theorem were presented in [6]. Moreover, there are FETI-H method [15], FETI-DPH method [14] and the source transfer domain decomposition method [7, 8] for the Helmholtz equation. Another efficient preconditioner, sweeping preconditioner, for the Helmholtz equation was presented and studied in [11,12]. There are some summary articles relating this topic, such as [1,13,19,20].

In this article, we propose to combine the deep neural network with the domain decomposition algorithm in [6] to obtain an efficient mesh-free algorithm for heterogeneous Helmholtz equations. Based on the algorithm used in [6], we introduce a domain decomposition algorithm for system (1.3) in the following.

Assuming that $k(x)$ in (1.3) are piecewise constants, i.e., $k(x)=k_1$ in Ω_1, and $k(x)=k_2$ in Ω_2, where Ω_1 and Ω_2 are two non-overlapping subdomains in the domain Ω, and the interface $\Gamma=\bar{\Omega}_1\cap\bar{\Omega}_2$. Let

$$f_1=f|_{\Omega_1},\quad f_2=f|_{\Omega_2}$$

and γ_1, γ_2, μ be three constant parameters whose values shall be determined later. Giving the initial guess value g_1^1, then the domain decomposition iterative procedure can be defined as follows (for $n=1,2,\cdots$):

1. Solve for u_1^n in Ω_1,

$$\begin{cases} -\Delta u_1^n - k_1^2 u_1^n = f_1 & \text{in } \Omega_1, \\ \dfrac{\partial u_1^n}{\partial \mathbf{n}_1} + \mathbf{i}k_1 u_1^n = g_a & \text{on } \partial\Omega_1 \backslash \Gamma, \\ \dfrac{\partial u_1^n}{\partial \mathbf{n}_1} + \gamma_1 u_1^n = g_1^n & \text{on } \Gamma, \end{cases} \tag{2.1}$$

2. Update the transmission condition along the interface Γ,

$$g_2^n = -\frac{\partial u_1^n}{\partial \mathbf{n}_1} + \gamma_2 u_1^n + g_c - \gamma_2 g_b, \tag{2.2}$$

3. Solve for u_2^n in Ω_2,

$$\begin{cases} -\Delta u_2^n - k_2^2 u_2^n = f_2 & \text{in } \Omega_2, \\ \dfrac{\partial u_2^n}{\partial \mathbf{n}_2} + \mathbf{i}k_2 u_2^n = g_a & \text{on } \partial\Omega_2 \backslash \Gamma, \\ \dfrac{\partial u_2^n}{\partial \mathbf{n}_2} + \gamma_2 u_2^n = g_2^n & \text{on } \Gamma, \end{cases} \tag{2.3}$$

4. Update the transmission condition along the interface Γ,

$$g_1^{n+\frac{1}{2}} = -\frac{\partial u_2^n}{\partial \mathbf{n}_2} + \gamma_1 u_2^n + g_c + \gamma_1 g_b, \tag{2.4}$$

5. Relax the transmission condition along the interface,

$$g_1^{n+1} = \mu g_1^{n+\frac{1}{2}} + (1-\mu) g_1^n. \tag{2.5}$$

If g_b and g_c are fixed as 0, i.e., without jump on the interface, the above algorithm is same to the algorithm was used in [6]. The choice of γ_1, γ_2 and μ will affect the convergence rate of this algorithm, but it is not the main concern of this paper. In order to meet the convergence conditions of the algorithm, the real part and the imaginary part of γ_1 and γ_2 should be nonnegative. However, when the boundary condition (1.3b) is $\partial u/\partial \mathbf{n} - \mathbf{i}k(x)u = g_a$, the imaginary part of γ_1 and γ_2 should be negative [6, 21]. For further details, we refer the interested readers to [6]. The specific parameter selection will be introduced in the numerical experiment part of this paper.

3 Neural networks for Helmholtz equations

Neural networks are indeed a powerful tool, which has been proved to be effective in many fields, even in the field of numerical solution of partial differential equations

which is a problem with high complexity. Focusing on a general PDE, some neural network architectures and optimization methods were designed from different perspectives [10, 28, 29]. As a typical case, Raissi et al. introduced Physics-Informed Neural Networks (PINNs) that take initial conditions and boundary conditions as the penalties of the optimization objective loss function in [28], where numerical experiments illustrate that network can achieve good accuracy for both forward and inverse problems. Otherwise, aiming at a kind of PDEs not general equations, some researchers introduced different neural network architectures and activation functions based on a prior knowledge of mathematics and physics, [9, 16, 25, 30] and so on.

We now introduce the physics-informed neural network that is a deep learning framework for solving PDEs [28].

We consider deep fully connected feedforward neural networks. The entire neural network consists of $L+1$ layers, where layer 0 is the input layer and layer L is the output layer. Layers $0<l<L$ are the hidden layers. All of the layers have an activation function, excluding the output layer.

Mathematically, we denote $d_0, d_1, \cdots, d_L$ as a list of integers, with d_0, d_L representing the lengths of the input signal and output signal of the neural network. Define a function $\mathbf{T}_l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_{l+1}}$, $0 \le l < L$,

$$\mathbf{T}_l(\mathbf{x}) = \mathbf{W}_l \mathbf{x} + \mathbf{b}_l, \tag{3.1}$$

where $\mathbf{W}_l \in \mathbb{R}^{d_{l+1} \times d_l}$ and $\mathbf{b}_l \in \mathbb{R}^{d_{l+1}}$. Thus, we can simply represent a deep fully connected feedforward neural network using the composite function $h(\cdot;\Theta) : \mathbb{R}^{d_0} \to \mathbb{R}^{d_L}$,

$$h(\cdot;\Theta) = \mathbf{T}_{L-1} \circ \sigma \circ \mathbf{T}_{L-2} \circ \cdots \circ \mathbf{T}_1 \circ \sigma \circ \mathbf{T}_0, \tag{3.2}$$

where σ is the activation function and $\Theta := \{\mathbf{W}_l, \mathbf{b}_l : 0 \le l < L\}$ represents the collection of all parameters.

Solving a Helmholtz equation such as (1.3) by a deep neural network is a physics-informed minimization problem with the objective $\mathcal{M}(\theta)$ consisting of two terms as follows:

$$\Theta^* = \operatorname{argmin}_\Theta \mathcal{M}(\Theta) := \mathcal{M}_\Omega(\Theta) + \mathcal{M}_{\partial\Omega}(\Theta) \tag{3.3}$$

with

$$\mathcal{M}_\Omega(\Theta) := \frac{1}{N_f} \sum_{i=1}^{N_f} \left| -\Delta h(\mathbf{x}_f^i;\Theta) - k^2 h(\mathbf{x}_f^i;\Theta) \right|^2,$$

$$\mathcal{M}_{\partial\Omega}(\Theta) := \frac{1}{N_g} \sum_{i=1}^{N_g} \left| \frac{\partial h(\mathbf{x}_g^i;\Theta)}{\partial \mathbf{n}} + \mathbf{i}k h(\mathbf{x}_g^i;\Theta) - g(\mathbf{x}_g^i) \right|^2,$$

where $\{\mathbf{x}_f^i\}_{i=1}^{N_f}$ and $\{\mathbf{x}_g^i\}_{i=1}^{N_g}$ are the collocation points in the inside and on the boundary, respectively. The domain term $\mathcal{M}_\Omega$ and boundary term $\mathcal{M}_{\partial\Omega}$ enforce the condition that the desired optimized neural network $h(\cdot;\Theta^*)$ satisfies governed equations and boundary conditions, respectively.

Gradient descent-based methods such as gradient descent, Adam and L-BFGS can be used to solve this kind of optimization problem [5, 22, 23, 27].

Note that PINNs use hyperphysical tangent activation functions as a substitute for ReLU and Sigmoid functions that are heavily used in computer vision and pattern recognition [28]. Sinusoidal representation networks (SIRENs) use the sinusoidal function as the activation function, and apply the periodic property of sinusoidal function to get better approximation effect [30].

Based on the object studied in this paper, Helmholtz equation, whose solution is a complex-valued function and indicates the propagation of waves, we also use complex exponential function $e^{\mathbf{i}x}$ as the activation function which is proposed in [9]. From the results of numerical experiments in [9], the neural network architecture with plane wave basis ($e^{\mathbf{i}x}$) as activation function (PWNN) has obvious advantages in solving the Helmholtz equation.

In the rest of this section, we introduce briefly the complex-valued neural networks architecture using $e^{\mathbf{i}x}$ as the activation function. Because the solution of Helmholtz equation is a complex-valued function, parameters in the used neural network are complex values. To be specific, unlike (3.1) for the space of $\mathbf{W}_l$ and $\mathbf{b}_l$, now we assume the parameters for the last layer $\mathbf{W}_{L-1} \in \mathbb{C}^{d_L \times d_{L-1}}$ and $\mathbf{b}_{L-1} \in \mathbb{C}^{d_L}$. However, the architecture of PWNN is also presented with same expression,

$$h(\cdot;\Theta) = \mathbf{T}_{L-1} \circ \sigma \circ \mathbf{T}_{L-2} \circ \cdots \circ \mathbf{T}_1 \circ \sigma \circ \mathbf{T}_0, \tag{3.4}$$

where the activation function $\sigma(x) = e^{\mathbf{i}x}$. If one expands the expression of two-layer PWNN, then

$$\mathbf{T}_l \circ \sigma \circ \mathbf{T}_{l-1}(\mathbf{x}) = \mathbf{W}_l e^{\mathbf{i}(\mathbf{W}_{l-1}\mathbf{x} + \mathbf{b}_{l-1})} + \mathbf{b}_l. \tag{3.5}$$

In fact, the solution of homogeneous Helmholtz equation can be expressed by plane wave basis functions. Assuming that $u(\mathbf{x})$ is the solution of homogeneous (1.3) and $\mathbf{d}_\theta := (\cos\theta, \sin\theta)$ is the plane wave direction. From Theorem 2.1 in [32], we know that $\forall \epsilon > 0$, $\exists D(\theta) : [0, 2\pi] \mapsto \mathbb{C}$, s.t.

$$\left\| u(\mathbf{x}) - \int_0^{2\pi} D(\theta) e^{\mathbf{i}k\mathbf{d}_\theta \cdot \mathbf{x}} d\theta \right\|_{0,\Omega} < \epsilon. \tag{3.6}$$

In other words, the solution of the homogeneous Helmholtz equation can be written as the integral form of plane waves in multiple directions.

Note that the right hand term of (3.5) can be used as the discrete formula for (3.6) if parameters $\mathbf{W}$ and $\mathbf{b}$ are specifically selected. When solving the Helmholtz equation, because the plane wave form is satisfied, PWNN has obvious advantages over other neural network architectures. The knowledge of plane wave basis functions can guide us to initialize neural network parameters.

If L-BFGS optimization method is used to solve the physics-informed minimization problem (3.3), the profile of the algorithm PWNN for Helmholtz is given in Algorithm 3.1.

Algorithm 3.1 PWNN for Helmholtz equation.

1: Input training data $\mathbf{X}$ and initial parameters Θ^0;
2: **for** $j=0,1,\cdots$ **do**
3: Calculate $\nabla_{\Theta^j}\mathcal{M}(\Theta^j;\mathbf{X})$; ▷ Save recent m steps' gradients
4: Update: $\Theta^{j+1}\leftarrow\Theta^j+\text{L-BFGS}(\{\nabla_{\Theta^k}\mathcal{M}(\Theta^k;\mathbf{X})\}_{k=\max(0,j-m+1)}^{j})$;
5: **end for**

Remark 3.1. L-BFGS is an improved algorithm for the quasi-Newton method: BFGS. In BFGS, the approximate Hesse matrix B_k^{-1} is stored at every step, which wastes a lot of storage space in high dimensional cases. But in L-BFGS, only the recent m steps' iterative information is saved for calculation B_k^{-1} to reduce the storage space of data. We fix $m=50$ in numerical experiments.

4 DeepDDM for Helmholtz equations

It is natural to mix domain decomposition methods with neural networks. In this section, we only present a two-subdomain case as an example for simplification of notations and descriptions. For multi-subdomain cases, it is necessary to use more complex symbols for delivering exactly the algorithm, see [6] for detail.

However, when the boundary conditions are not as simple as the formula (1.3), i.e., when different edges own different boundary conditions, the objective function $\mathcal{M}(\theta)$ given by (3.3) needs to be modified. Taking (2.1) as an example, its objective function is given as follows

$$\mathcal{M}(\Theta)=\mathcal{M}_{\Omega_1}(\Theta)+\mathcal{M}_{\partial\Omega_1\setminus\Gamma}(\Theta)+\mathcal{M}_{\Gamma}(\Theta), \tag{4.1}$$

where

$$\mathcal{M}_{\Omega_1}(\Theta)=\frac{1}{N_f}\sum_{i=1}^{N_f}\left|-\Delta\mathcal{N}(\mathbf{x}_f^i;\Theta)-k^2\mathcal{N}(\mathbf{x}_f^i;\Theta)-f_1(\mathbf{x}_f^i)\right|^2, \tag{4.2a}$$

$$\mathcal{M}_{\partial\Omega_1\setminus\Gamma}(\Theta):=\frac{1}{N_g}\sum_{i=1}^{N_g}\left|\frac{\partial\mathcal{N}(\mathbf{x}_g^i;\Theta)}{\partial\mathbf{n}_1}+\mathbf{i}k\mathcal{N}(\mathbf{x}_g^i;\Theta)-g_a(\mathbf{x}_g^i)\right|^2, \tag{4.2b}$$

$$\mathcal{M}_{\Gamma}(\Theta):=\frac{1}{N_\Gamma}\sum_{i=1}^{N_\Gamma}\left|\frac{\partial\mathcal{N}(\mathbf{x}_\Gamma^i;\Theta)}{\partial\mathbf{n}_1}+\gamma_1\mathcal{N}(\mathbf{x}_\Gamma^i;\Theta)-g_1(\mathbf{x}_\Gamma^i)\right|^2, \tag{4.2c}$$

with $\{\mathbf{x}_\Gamma^i\}_{i=1}^{N_\Gamma}$ are the collocation points on the interface Γ.

Furthermore, we have an algorithm, called DeepDDM in this paper, for the Helmholtz equation, which inherits virtues from domain decomposition and neural network. The profile of the algorithm for the Helmholtz equation is given below.

Algorithm 4.1 DeepDDM for the Helmholtz equation.

1: Initial interface information g_1^1 along Γ;
2: **for** $n=1,2,\cdots$ **do**
3: Retrain the latest model for u_1^n using Algorithm 3.1;
4: Update the transmission condition $g_2^n=-\frac{\partial u_1^n}{\partial \mathbf{n}_1}+\gamma_2 u_1^n+g_c-\gamma_2 g_b$;
5: Retrain the latest model for u_2^n using Algorithm 3.1;
6: Update the transmission condition $g_1^{n+1/2}=-\frac{\partial u_2^n}{\partial \mathbf{n}_2}+\gamma_1 u_2^n+g_c+\gamma_1 g_b$;
7: Relax $g_1^{n+1}=\mu g_1^{n+1/2}+(1-\mu)g_1^n$;
8: **if** $\min\{\|g_1^{n+1}-g_1^n\|,\|g_2^n-g_2^{n-1}\|\}<tol_\Gamma$ **then**
9: STOP;
10: **end if**
11: **end for**

Remark 4.1. In the Step 3 and Step 5 of Algorithm 4, retraining the latest model, i.e., the network parameters obtained in the previous iteration are used as the initialization of the current training network, makes subsequent training processes faster than the first training process.

When stochastic gradient descent is used to train neural network parameters, different from the random sampling in the field of computer vision, we always require each batch of training data including information of boundary conditions. From the perspective of PDE theory, it is guaranteed that a numerical PDE corresponding to each batch optimization problem is well-posed. Moreover, each batch has a different amount of training data for accelerating the training process. This idea coincides naturally with the idea of multigrid with coarse grid and fine grid [31]. A batch with less internal training data corresponds to a coarse grid, and a batch with more internal training data corresponds to a fine grid.

5 Experiments

In this section, we present a series of numerical experiments to illustrate the property of our algorithm.

5.1 Settings

We use Intel Core-i7-10700 and 32 GB of DDR4 RAM for calculation in the framework of TensorFlow. For FDM, we use *scipy.sparse.linalg.spsolve* as a direct solver of sparse linear equations, which provides a wrapper of SuperLU sparse direct solver in SciPy. The domain of interest is a bounded square domain $[0,1]^2$ in where waves propagate. When using domain decomposition splits the whole domain into multiple subdomains, we always split it along x-axis.

For the algorithm given in Section 2, how to choose the best relaxation parameters γ_1, γ_2, μ is still an unanswered problem. Empirically, one can solve numerically a min-max problem to have a good parameter choice [17, 18]. However, it needs to solve min-max problems repeatedly for different physical parameters and discrete parameters, which will cause expensive computational cost and is not conducive to the description and interpretation of numerical experiments. In fact, the selection of relaxation parameters is not the focus of this paper. In this paper, we will select appropriate parameters based on experience and some exploratory numerical experiments to be given in Tables 2, Table 3 and Table 5 to make the DeepDDM have good results for different cases.

For each subproblem, we use PWNN with one hidden layer and *Units* indicates the number of hidden neurons. The stop criterion of outer iteration is that the difference of two adjacent interface conditions is less than the given threshold, as mentioned above, in formulas,

$$\max_i \|g_i^n - g_i^{n-1}\|_2 < 10^{-2}. \tag{5.1}$$

The network training process is set as follows: we use the L-BFGS optimizer and the stop criterion is that $\|\nabla \mathcal{M}\|_\infty < 2\times 10^{-16}$, or that we exceeded the maximum number of allowed iterations, set as 3000 here. The training data include sample points inside of domain and on the boundary, using N_f and N_g for the number of samples. The training points in the inside are randomly selected, while the training points on the boundary are uniformly selected. More training points usually lead to smaller approximation error of neural network and higher numerical solution accuracy. Meanwhile, more training points also mean more training time and more expensive computational costs. We have to trade-off precision against time and cost. Fortunately, it has been shown in [9] that $N_f = C$ and $N_g = \mathcal{O}(k)$ is enough for PWNN to achieve high accuracy, and we follow this strategy in the experiments. The test points used to estimate the relative L_2 errors are uniformly sampled by row and column, with 40000 in all. For the finite difference method (FDM), we use the nine-point finite difference scheme with third-order accuracy. We use a uniform square grid, and *Mesh* represents how many cells we divide the calculation area into. We have the expression of error in the following

$$\mathcal{E} = \frac{\|u_* - u_h\|_2}{\|u_*\|_2}. \tag{5.2}$$

5.2 Helmholtz equations with constant wave number

In this section, we consider a Helmholtz equation with a constant wave number k in the whole domain.

$$\begin{cases} -\Delta u - k^2 u = 0 & \text{in } \Omega, \\ \dfrac{\partial u}{\partial \mathbf{n}} + \mathbf{i}ku = g & \text{on } \partial\Omega, \end{cases} \tag{5.3}$$

Table 1: Relative L^2 error ε and time cost of PWNN and FDM under different *Units* or meshsize on problem (5.3).

	PWNN				FDM			
	Units #parameters	$\mathcal{E}$	Total time	Train time	*Mesh* #freedoms	$\mathcal{E}$	Total time	Solve time
$k=20$	16 #66	3.6e-2	6.4s	1.3s	20*20 #882	8.1e-2	0s	0s
	18 #74	4.9e-3	6.5s	1.4s	40*40 #3362	1.5e-2	0.1s	0s
	20 #82	4.5e-4	6.5s	1.4s	60*60 #7442	4.6e-3	0.2s	0.1s
$k=60$	48 #194	2.0e-3	6.8s	1.7s	60*60 #7442	8.9e-2	0.2s	0.1s
	54 #218	2.5e-4	6.4s	1.3s	120*120 #29282	1.1e-2	1.4s	1.1s
	60 #242	2.0e-6	6.4s	1.3s	180*180 #65522	3.1e-3	4.8s	4.1s
$k=100$	80 #322	1.4e-3	6.6s	1.3s	100*100 #20402	7.1e-2	0.9s	0.6s
	90 #362	5.0e-6	6.4s	1.3s	200*200 #80802	9.2e-3	5.8s	4.9s
	100 #402	2.4e-6	6.4s	1.3s	300*300 #181202	2.7e-3	19.1s	17.1s

Table 2: Relative L^2 error ε (the number of outer iterations I_S) of DeepDDM and FDM-DDM under different parameter μ on problem (5.3). Here $k=20$, $\gamma_1=\gamma_2=k(1+\mathbf{i})$.

	#parameters \ μ	0.2	0.4	0.6	0.8	1.0
DeepDDM	#98	5.1e-2(25)	2.0e-2(16)	1.0e-2(12)	8.4e-3(9)	7.1e-3(8)
FDM-DDM	#7442	1.1e-2(20)	1.1e-2(12)	1.0e-2(9)	1.0e-2(8)	1.0e-2(8)

The exact solution is set as

$$u_*(\mathbf{x}) = J_0(k|\mathbf{x}-\mathbf{s}|),$$

where J_0 denotes the Bessel function of the first kind and order 0 and $\mathbf{s}=(0.5,0.5)$. We substitute the exact solution into (5.3) to compute g.

We solve the problem without using DDM first. We use PWNN and FDM to solve the problem. Because PWNN satisfies the plane wave expansion of the Helmholtz equation, only a few internal sample points are needed, the numbers of training data are fixed to $N_f=200$, $N_g=2k\times 4$.

In Table 1, the relative L^2 errors and time costs are shown for different *Units* or *Mesh* for PWNN and FDM. As k increases, the time cost of PWNN does not change significantly. This is because the number of internal sample points of PWNN does not change with k. When k is not very large, the second derivative of internal sample points is required, which is the main time-consuming place. When $k=100$, it is obvious that PWNN achieves higher accuracy than FDM in a shorter time. This indicates that PWNN is more efficient than FDM in solving large wave number problems.

To illustrate the effectiveness of DeepDDM and investigate its properties, we present the results of the two-subdomain case. In practice, we set $\Omega_1:=[0,0.5]\times[0,1]$, $\Omega_2:=[0.5,1]\times[0,1]$, and $tol_\Gamma=10^{-2}$. The numbers of training data are fixed to $N_f=200$, $N_g=2k\times 4$, $N_\Gamma=2k$. When $k=20$, we run some simple test on the choice of γ_1, γ_2 and μ, the results are shown in Table 2 and Table 3. After comprehensive test results, for different wave number k, we let $\gamma_1=\gamma_2=\sqrt{10k}(1+\mathbf{i})$, $\mu=1$.

Table 3: Relative L^2 error ε (the number of outer iterations I_S) of DeepDDM and FDM-DDM under different parameter γ_1,γ_2 on problem (5.3). Here $k=20$, $\mu=1$.

	$\gamma_1=\gamma_2=$ / #parameters	$20+10\mathbf{i}$	$20+30\mathbf{i}$	$20+20\mathbf{i}$	$10+10\mathbf{i}$	$\sqrt{200}+\sqrt{200}\mathbf{i}$
DeepDDM	#98	6.6e-3(11)	7.4e-3(9)	7.1e-3(8)	3.7e-3(7)	3.6e-3(7)
FDM-DDM	#7442	8.6e-3(11)	1.2e-2(9)	1.0e-2(8)	1.2e-2(7)	1.1e-2(7)

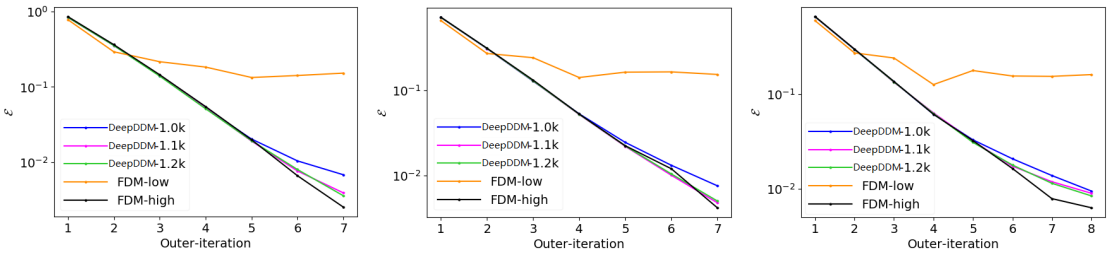
Table 4: Relative L^2 error ε (the number of outer iterations I_S) of DeepDDM and FDM-DDM under different *Units* or meshsize on problem (5.3).

	DeepDDM		FDM-DDM	
	Units #parameters	$\mathcal{E}(I_S)$	*Mesh* #freedoms	$\mathcal{E}(I_S)$
$k=20$	20 #82	6.8e-3(7)	20*10 #462	1.5e-1(8)
	22 #90	3.9e-3(7)	40*20 #1722	3.3e-2(7)
	24 #98	3.6e-3(7)	60*30 #3782	1.1e-2(7)
$k=60$	60 #242	3.9e-3(8)	60*30 #3782	1.6e-1(9)
	66 #266	4.7e-3(7)	120*60 #14762	2.9e-2(8)
	72 #290	5.0e-3(7)	180*90 #32942	7.6e-3(7)
$k=100$	100 #402	8.7e-3(9)	100*50 #10302	1.6e-1(10)
	110 #442	9.0e-3(8)	200*100 #40602	2.8e-2(9)
	120 #482	7.9e-3(9)	300*150 #90902	8.3e-3(9)

In Table 4, the relative L^2 errors and the number of outer iterations are shown for different *Units* or *Mesh* for DeepDDM and FDM-DDM. Because the wave number is the same, we use the same network structure of PWNN in the two regions. Also for FDM, we use the same number of grids in Ω_1 and Ω_2. As the degrees of freedom (DOFs) increases, the errors of solutions can reach the order of 10^{-3}. Otherwise, if we focus on the number of outer iterations, DeepDDM and FDM-DDM are almost the same, and the general trend indicates that the number of outer iterations remains approximately constant no matter how the *Units* or *Mesh* changes.

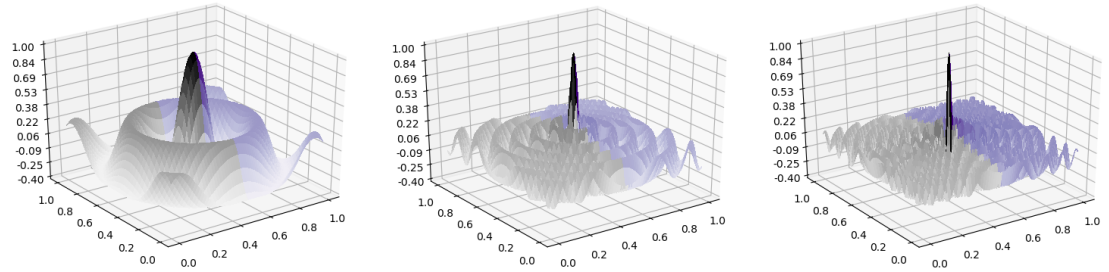
The processes of error convergence for different k are shown in Fig. 1. Here DeepDDM-1.0k represents using PWNN with $Units=1.0k$ to solve subproblems, and DeepDDM-1.1k, DeepDDM-1.2k represent similar meanings. FDM-low represents the using FDM method with a grid size of $h=1/k$ to solve subproblems, while $h=1/5k$ for FDM-high. It can be seen that the convergence speed of these methods is similar at the beginning, but there are differences in the following iterations. This is because the initial error mainly comes from the wrong interface conditions. As the algorithm progresses, the error tends to be caused by the insufficient DOFs and the optimization error of the neural network.

Fig. 2 is the simulation results of DeepDDM with $Units=1.2k$ for wave number 20, 60 and 100. The black and purple waveforms respectively represent the numerical solutions in Ω_1 and Ω_2. As we can see, the obtained waveforms in this figure are very clear, and the wave has more vibration as k increases.



(a) $k=20$ (b) $k=60$ (c) $k=100$

Figure 1: The change in relative L^2 error along with out-iteration on problem (5.3).



(a) $k=20$ (b) $k=60$ (c) $k=100$

Figure 2: Wavefield for different wave number k obtained by DeepDDM on problem (5.3).

5.3 Helmholtz equations with piecewise constant wave number

In this section, we consider a Helmholtz equation with a piecewise constant wave number. Let's focus on the following problem first.

$$\begin{cases} -\Delta u - k_1^2 u = \delta_{\mathbf{s}} & \text{in } \Omega_1, \\ -\Delta u - k_2^2 u = 0 & \text{in } \Omega_2, \\ [u] = 0 & \text{on } \Gamma, \\ \left[\dfrac{\partial u}{\partial \mathbf{n}}\right] = 0 & \text{on } \Gamma, \\ \dfrac{\partial u}{\partial \mathbf{n}} - \mathbf{i}k_1 u = 0 & \text{on } \partial\Omega_1 \backslash \Gamma, \\ \dfrac{\partial u}{\partial \mathbf{n}} - \mathbf{i}k_2 u = 0 & \text{on } \partial\Omega_2 \backslash \Gamma, \end{cases} \tag{5.4}$$

where $\Omega_1 := [0,0.5]\times[0,1]$, $\Omega_2 := [0.5,1]\times[0,1]$, $\Gamma := \{(x,y)|x=0.5,\ y\in[0,1]\}$. We set the wave source to $\mathbf{s}=(0.25,0.5)$. There is no analytical solution to this problem, and we use the numerical solution of the third order FDM with more than 4 million DOFs as the exact solution.

Table 5: Relative L^2 error ε (the number of outer iterations I_S) of DeepDDM and FDM-DDM under different parameter γ_1,γ_2 on problem (5.5). Here $k_1=20$, $k_2=60$, $\mu=1$.

	#parameters \ $\gamma_1=\gamma_2=$	$20-20\mathbf{i}$	$60-60\mathbf{i}$	$\sqrt{1200}-\sqrt{1200}\mathbf{i}$
DeepDDM	#388	3.9e-3(10)	1.0e-2(11)	4.3e-3(9)
FDM-DDM	#36724	3.3e-3(10)	3.5e-3(11)	3.7e-3(9)

In Ω_1, as the non-zero wave source term makes the exact solution $u_*|_{\Omega_1}$ not satisfying the plane wave expansion, we use PWNN to approximate $u_*(\mathbf{x})-G(\mathbf{x},\mathbf{s})$. Here $G(\mathbf{x},\mathbf{s})$ is the fundamental solution of the Helmholtz equation of constant wave number k_1,

$$-\Delta G(\mathbf{x},\mathbf{s})-k_1^2G(\mathbf{x},\mathbf{s})=\delta_{\mathbf{s}} \quad \text{in } \mathbb{R}^2, \quad G(\mathbf{x},\mathbf{s})=\frac{\mathbf{i}}{4}H_0^{(1)}(k_1|\mathbf{x}-\mathbf{s}|).$$

Here $H_0^{(1)}(z)$ for $z\in\mathbb{C}$, is the first Hankel function of order zero. In each out iteration, we will add or subtract the change in boundary conditions caused by $G(\mathbf{x},\mathbf{s})$ at $\partial\Omega_1\backslash\Gamma$ and Γ correspondingly, thus problem (5.4) becomes

$$\begin{cases} -\Delta\hat{u}-k_1^2\hat{u}=0 & \text{in } \Omega_1,\\ -\Delta\hat{u}-k_2^2\hat{u}=0 & \text{in } \Omega_2,\\ [\hat{u}]=-G(\mathbf{x},\mathbf{s}) & \text{on } \Gamma,\\ \left[\dfrac{\partial\hat{u}}{\partial\mathbf{n}}\right]=-\dfrac{\partial G(\mathbf{x},\mathbf{s})}{\partial\mathbf{n}} & \text{on } \Gamma,\\ \dfrac{\partial\hat{u}}{\partial\mathbf{n}}-\mathbf{i}k_1\hat{u}=0 & \text{on } \partial\Omega_1\backslash\Gamma,\\ \dfrac{\partial\hat{u}}{\partial\mathbf{n}}-\mathbf{i}k_2\hat{u}=0 & \text{on } \partial\Omega_2\backslash\Gamma. \end{cases} \tag{5.5}$$

Because the wave numbers are different in Ω_1 and Ω_2, we will use solvers with different DOFs in the two regions. The *Mesh* of FDM and the *Units* of PWNN will be proportional to wave number k. Let $tol_\Gamma=10^{-2}$. When $k_1=20$,$k_2=60$, we keep $\mu=1$ and run some simple test on the choice of γ_1,γ_2, the results are shown in Table 5. After comprehensive test results, for different k_1,k_2, we let $\gamma_1=\gamma_2=\sqrt{k_1k_2}(1-\mathbf{i})$.

The relative L^2 errors and the number of outer iterations for different *Units* or *Mesh* for DeepDDM and FDM-DDM are shown in Table 6. Here $Units_1$ and $Units_2$ represent the width of PWNN in Ω_1 and Ω_2, respectively, while $Mesh_1$ and $Mesh_2$ have similar meanings. It can be seen that in the case of discontinuous media, DeepDDM and FDM-DDM also have almost the same number of outer iterations, which shows the scalability of DeepDDM. A sharp interface problem is also considered ($k_1=1$, $k_2=200$), DeepDDM can still handle this kind of problem and converges faster than FDM-DDM.

The process of error convergence for different k_1, k_2 are shown in Fig. 3. In the case of piecewise constant wave number, we can get a conclusion similar to that in the

Table 6: Relative L^2 error ε (the number of outer iterations I_S) of DeepDDM and FDM-DDM under different $Units$ or meshsize on problem (5.5).

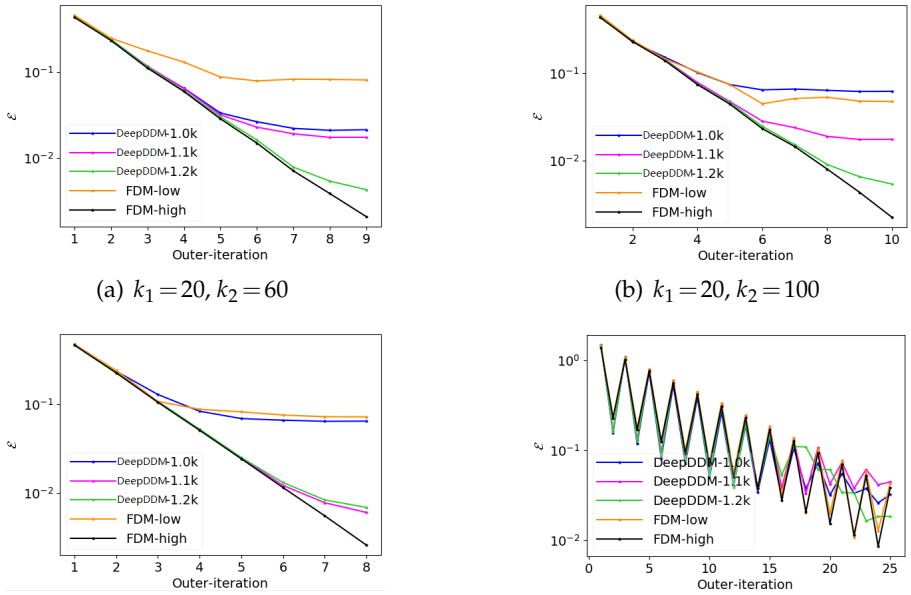
	DeepDDM				FDM-DDM			
	$Units_1$	$Units_2$	#parameters	$\mathcal{E}(I_S)$	$Mesh_1$	$Mesh_2$	#freedoms	$\mathcal{E}(I_S)$
$k_1=20, k_2=60$	20	60	#324	2.2e-2(9)	20*10	60*30	#4244	8.0e-2(10)
	22	66	#356	1.7e-2(9)	40*20	120*60	#16484	1.1e-2(9)
	24	72	#388	4.3e-3(9)	60*30	180*90	#36724	3.7e-3(9)
$k_1=20, k_2=100$	20	100	#484	6.2e-2(10)	20*10	100*50	#10764	4.9e-2(11)
	22	110	#532	1.8e-2(10)	40*20	200*100	#42324	8.6e-3(10)
	24	120	#580	5.4e-3(10)	60*30	300*150	#94684	3.5e-3(10)
$k_1=60, k_2=100$	60	100	#644	6.4e-2(8)	60*30	100*50	#14084	7.3e-2(9)
	66	110	#708	5.6e-3(8)	120*60	200*100	#55364	1.1e-2(9)
	72	120	#772	6.5e-3(8)	180*90	300*150	#123844	3.3e-3(9)
$k_1=1, k_2=200$	20	200	#884	2.7e-2(28)	20*10	200*100	#41064	5.7e-3(44)
	22	220	#972	3.7e-2(28)	40*20	400*200	#162924	1.5e-3(44)
	24	240	#1060	1.8e-2(25)	60*30	600*300	#365584	1.0e-3(44)

case of constant wave number, see Fig. 1. This shows that DeepDDM can deal with the Helmholtz equation with piecewise constant wave number well, and is no longer limited to the problem with constant wave number. In the case of $k_1=1$, $k_2=200$, it is too small to use $Units_1$ and $Mesh_1$ corresponding to k_1, we set $Units_1=20$, $Units_2=200$ for DeepDDM-1.0k, $Mesh_1=20*10$, $Mesh_2=200*100$ for FDM-low, $Mesh_1=100*50$, $Mesh_2=1000*500$ for FDM-high, and similar for DeepDDM-1.1k and DeepDDM-1.2k. Although the large difference between k_1 and k_2 brings oscillation to the error in the process of out iteration, it still shows a downward trend on the whole. Fig. 4 is the simulation results of DeepDDM with $Units_1=1.2k_1$, $Units_2=1.2k_2$ for different k_1, k_2. We can see the changes in vibration and amplitude caused by different wave numbers in the left and right half domains.

In the former problem, the source of the wave is inside the region, and the solution contains all directions. Then we focus on the exact solution that only exists with few unknown directions of plane wave refraction and reflection at the interfaces.

$$\begin{cases} -\Delta u - k_1^2 u = 0 & \text{in } \Omega_1, \\ -\Delta u - k_2^2 u = 0 & \text{in } \Omega_2, \\ [u] = 0 & \text{on } \Gamma, \\ \left[\dfrac{\partial u}{\partial \mathbf{n}}\right] = 0 & \text{on } \Gamma, \\ \dfrac{\partial u}{\partial \mathbf{n}} + \mathbf{i}k_1 u = g_1 & \text{on } \partial\Omega_1 \backslash \Gamma, \\ \dfrac{\partial u}{\partial \mathbf{n}} + \mathbf{i}k_2 u = g_2 & \text{on } \partial\Omega_2 \backslash \Gamma, \end{cases} \tag{5.6}$$

where $\Omega_1:=[-0.5,0]\times[-0.5,0.5]$, $\Omega_2:=[0,0.5]\times[-0.5,0.5]$, $\Gamma:=\{(x,y)|x=0, y\in[-0.5,0.5]\}$.



(a) $k_1=20, k_2=60$ (b) $k_1=20, k_2=100$

(c) $k_1=60, k_2=100$ (d) $k_1=1, k_2=200$

Figure 3: The change in relative L^2 error along with out-iteration on problem (5.5).

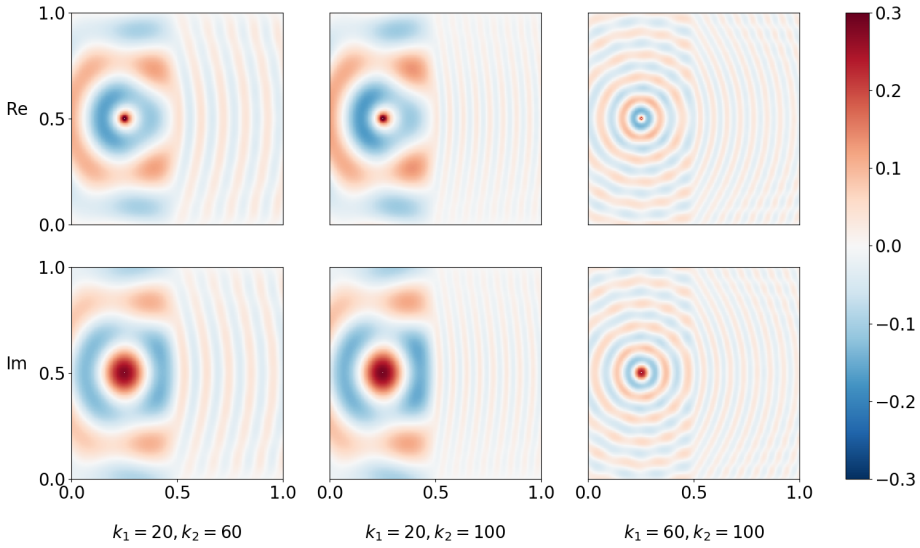
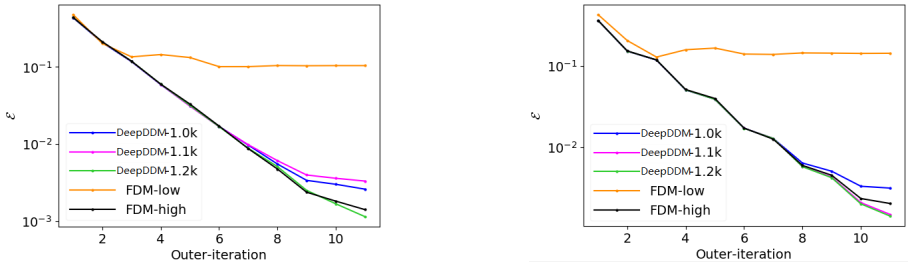


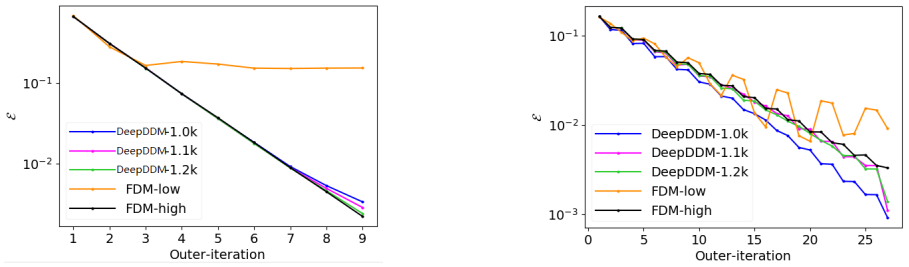
Figure 4: Wavefield contour for different wave number k obtained by DeepDDM on problem (5.5).

The exact solution is set as

$$\begin{cases} u_* = \sum_{i=1}^{d} \left(e^{\mathbf{i}k_1 \mathbf{I}_i^T \mathbf{x}} - a_i e^{\mathbf{i}k_1 \mathbf{R}_i^T \mathbf{x}} \right) & \text{in } \Omega_1, \\ u_* = \sum_{i=1}^{d} (1-a_i) e^{\mathbf{i}k_2 \mathbf{T}_i^T \mathbf{x}} & \text{in } \Omega_2, \end{cases} \tag{5.7}$$



(a) $k_1=20, k_2=60$ (b) $k_1=20, k_2=100$



(c) $k_1=60, k_2=100$ (d) $k_1=1, k_2=200$

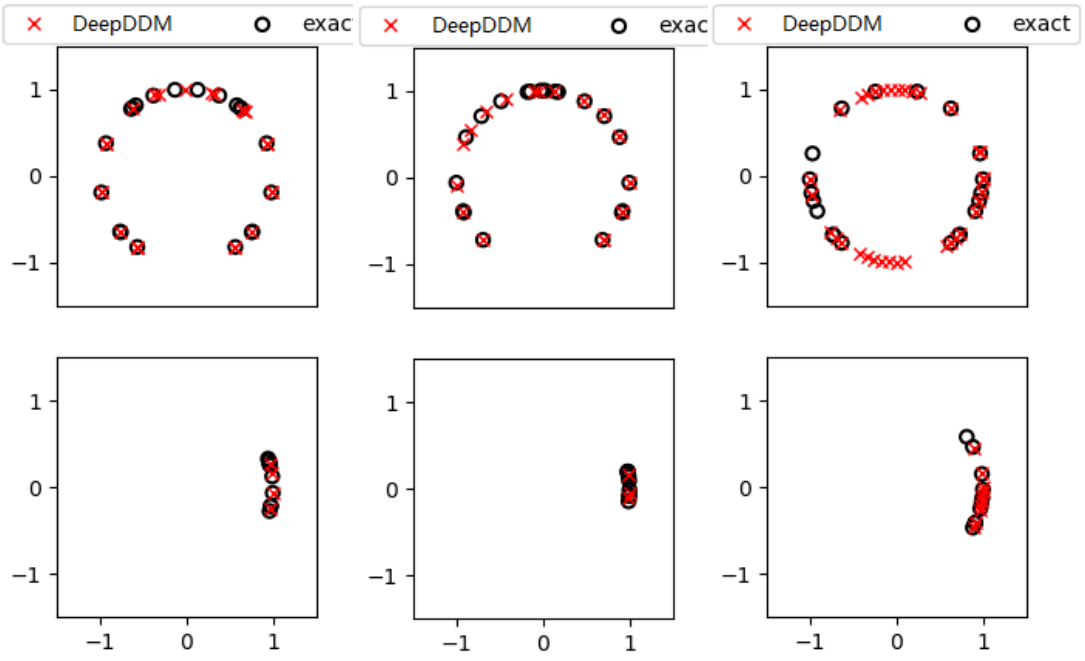
Figure 5: The change in relative L^2 error along with out-iteration on problem (5.6).

where $I_i=(\cos\theta_{I,i},\sin\theta_{I,i})^T$, $R_i=(-\cos\theta_{I,i},\sin\theta_{I,i})^T$, $T_i=(\cos\theta_{T,i},\sin\theta_{T,i})^T$ represents the direction of the incident, reflected and transmitted waves respectively. We have $k_2/k_1=\sin\theta_{I,i}/\sin\theta_{T,i}$ for each $0<i\leq d$, and $a_i=\sin(\theta_{I,i}-\theta_{T,i})/\sin(\theta_{I,i}+\theta_{T,i})$. Here we set $d=10$ and randomly generate $\{I_i\}$. We substitute the exact solution into (5.6) to compute g_1 and g_2. Follow the experience of our previous experiment, we let $tol_\Gamma=10^{-2}$, $\gamma_1=\gamma_2=\sqrt{k_1k_2}(1+\mathbf{i})$, $\mu=1$.

The relative L^2 errors and the number of outer iterations for different *Units* or *Mesh* for DeepDDM and FDM-DDM at this case are shown in Table 7. As can be seen, we have obtained a conclusion similar to the previous experiment, that DeepDDM has almost the same convergence rate as FDM-DDM. In the sharp interface problem ($k_1=1$, $k_2=200$), DeepDDM also has a faster convergence rate than FDM-DDM.

In Fig. 5, we plot the process of error convergence for different k_1, k_2 on problem (5.6). In this case, the error of DeepDDM-1.2k is even lower than that of FDM-DDM-high. This is because PWNN can find the direction of the exact solution, which is difficult for other methods. Same as before, in the case of $k_1=1$, $k_2=200$, we set $Units_1=20$, $Units_2=200$ for DeepDDM-1.0k, $Mesh_1=20*10$, $Mesh_2=200*100$ for FDM-low, $Mesh_1=100*50$, $Mesh_2=1000*500$ for FDM-high, and similar for DeepDDM-1.1k and DeepDDM-1.2k. The large difference between k_1 and k_2 also brings oscillation to the error in the process of out iteration in this case.

Fig. 6 shows the comparison of the direction predicted by DeepDDM-1.2K with the



(a) $k_1=20, k_2=60$ (b) $k_1=20, k_2=100$ (c) $k_1=60, k_2=100$

Figure 6: The directions $(\cos\theta,\sin\theta)$ of exact solutions and DeepDDM solutions on problem (5.6). The upper part and the lower part represent the direction of the wave in Ω_1 and Ω_2, respectively.

Table 7: Relative L^2 error ε(The number of outer iterations I_S) of DeepDDM and FDM-DDM under different $Units$ or meshsize on problem (5.6).

	DeepDDM				FDM-DDM			
	$Units_1$	$Units_2$	#parameters	$\mathcal{E}(I_S)$	$Mesh_1$	$Mesh_2$	#freedoms	$\mathcal{E}(I_S)$
$k_1=20, k_2=60$	20	60	#324	2.6e-3(11)	20*10	60*30	#4244	1.0e-1(11)
	22	66	#356	3.3e-3(11)	40*20	120*60	#16484	1.7e-2(11)
	24	72	#388	1.1e-3(11)	60*30	180*90	#36724	5.2e-3(11)
$k_1=20, k_2=100$	20	100	#484	3.1e-3(11)	20*10	100*50	#10764	1.4e-1(11)
	22	110	#532	1.5e-3(11)	40*20	200*100	#42324	2.1e-2(11)
	24	120	#580	1.4e-3(11)	60*30	300*150	#94684	6.4e-3(11)
$k_1=60, k_2=100$	60	100	#644	3.3e-3(9)	60*30	100*50	#14084	1.5e-1(9)
	66	110	#708	2.9e-3(9)	120*60	200*100	#55364	2.6e-2(9)
	72	120	#772	2.4e-3(9)	180*90	300*150	#123844	7.4e-3(9)
$k_1=1, k_2=200$	20	200	#884	9.1e-4(27)	20*10	200*100	#41064	1.2e-2(37)
	22	220	#972	1.2e-3(28)	40*20	400*200	#162924	2.2e-3(37)
	24	240	#1060	7.0e-4(29)	60*30	600*300	#365584	6.0e-4(36)

direction of the exact solution. For the sake of brevity, we have only drawn the predicted directions with an amplitude of bigger than 0.1. The upper part represents the directions of incident and reflection, while the lower part represents the directions of refraction. It can be seen that almost all directions of the exact solution are captured by DeepDDM,

which proves that DeepDDM does have the ability to find exact directions in Helmholtz equations with piecewise constant wave numbers.

6 Conclusions

We have introduced DeepDDM, a novel framework bridging deep learning, plane wave method and domain decomposition, to approximate solutions of the Helmholtz equation. The presented approach showcases a series of promising results for Helmholtz equations with constant and piecewise constant wave numbers. These numerical results demonstrate that the convergence rate of DeepDDM is close to that of FDM-DDM. Furthermore, DeepDDM takes less time for each out-iteration than FDM-DDM in the case of large wave numbers, which indicates that DeepDDM is more efficient than FDM-DDM for both constant and piecewise constant wave number problems.

This work presents some experiments to provide insights for theoretical study. We will develop DeepDDM for some more general cases such as the unbounded Helmholtz equation in the future.

Acknowledgements

This work was partially supported by National Key R&D Program of China Nos. 2019YFA0709600, 2019YFA0709602, China NSF under the grant numbers Nos. 11831016, 12171468, 11771440, 12071069, the Fundamental Research Funds for the Central Universities (No. JGPY202101), the Innovation Foundation of Qian Xuesen Laboratory of Space Technology.

References

[1] Carlos JS Alves, Nuno F. M. Martins, and Svilen S. Valtchev, *Domain decomposition methods with fundamental solutions for helmholtz problems with discontinuous source terms,* Comput. Math. Appl., 88 (2021), pp. 16–32.

[2] Ivo M. Babuska and Stefan A. Sauter, *Is the pollution effect of the fem avoidable for the Helmholtz equation considering high wave numbers?*, SIAM J. Numer. Anal., 34(6) (1997), pp. 2392–2423.

[3] Jean-David Benamou and Bruno Després, *A domain decomposition method for the Helmholtz equation and related optimal control problems,* J. Comput. Phys., 136(1) (1997), pp. 68–82.

[4] Jean-Pierre Berenger, *A perfectly matched layer for the absorption of electromagnetic waves,* J. Comput. Phys., 114(2) (1994), pp. 185–200.

[5] Léon Bottou, *Online algorithms and stochastic approximations,* David Saad, editor, Online Learning and Neural Networks, Cambridge University Press, Cambridge, UK, 1998.

[6] WENBIN CHEN, YONGXIANG LIU, AND XUEJUN XU, *A robust domain decomposition method for the Helmholtz equation with high wave number,* ESAIM: Math. Model. Numer. Anal., 50(3) (2016), pp. 921–944.
[7] ZHIMING CHEN AND XUESHUANG XIANG, *A source transfer domain decomposition method for Helmholtz equations in unbounded domain,* SIAM J. Numer. Anal., 51(4) (2013), pp. 2331–2356.
[8] ZHIMING CHEN AND XUESHUANG XIANG, *A source transfer domain decomposition method for Helmholtz equations in unbounded domain Part II: Extensions,* Numer. Math. Theory Methods Appl., 6(3) (2013), pp. 538–555.
[9] TAO CUI, ZIMING WANG, AND XUESHUANG XIANG, *An efficient neural network method with plane wave activation functions for solving helmholtz equation,* Comput. Math. Appl., 111 (2022), pp. 34–49.
[10] WEINAN E AND BING YU, *The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems,* Commun. Math. Statis., 6(1) (2018), pp. 1–12.
[11] BJÖRN ENGQUIST AND LEXING YING, *Sweeping preconditioner for the Helmholtz equation: hierarchical matrix representation,* Commun. Pure Appl. Math., 64(5) (2011), pp. 697–735.
[12] BJÖRN ENGQUIST AND LEXING YING, *Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers,* Multiscale Model. Simul., 9(2) (2011), pp. 686–710.
[13] OLIVER G. ERNST AND MARTIN J. GANDER, *Why it is difficult to solve Helmholtz problems with classical iterative methods,* Numer. Anal. Multiscale Problems, (2012), pp. 325–363.
[14] CHARBEL FARHAT, PHILIP AVERY, RADEK TEZAUR, AND JING LI, *FETI-DPH: a dual-primal domain decomposition method for acoustic scattering,* J. Comput. Acoust., 13(03) (2005), pp. 499–524.
[15] CHARBEL FARHAT, ANTONINI MACEDO, AND RADEK TEZAUR, *FETI-H: A scalable domain decomposition method for high frequency exterior Helmholtz problems,* Eleventh International Conference on Domain Decomposition Method, pages 231–241, Citeseer, 1999.
[16] WEI FENG AND HAIBO HUANG, *Fast prediction of immiscible two-phase displacements in heterogeneous porous media with convolutional neural network,* Adv. Appl. Math. Mech., 13(1) (2021), pp. 140–162.
[17] MARTIN J. GANDER, LAURENCE HALPERN, AND FRÉDÉRIC MAGOULES, *An optimized Schwarz method with two-sided Robin transmission conditions for the Helmholtz equation,* Int. J. Numer. Methods Fluids, 55(2) (2007), pp. 163–175.
[18] MARTIN J. GANDER, FRÉDÉRIC MAGOULES, AND FRÉDÉRIC NATAF, *Optimized Schwarz methods without overlap for the Helmholtz equation,* SIAM J. Sci. Comput., 24(1) (2002), pp. 38–60.
[19] MARTIN J. GANDER AND HUI ZHANG, *Domain decomposition methods for the Helmholtz equation: A numerical investigation,* Domain Decomposition Methods in Science and Engineering XX, pages 215–222, Springer, 2013.
[20] MARTIN J. GANDER AND HUI 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(1) (2019), pp. 3–76.
[21] FRANK IHLENBURG, Finite Element Analysis of Acoustic Scattering, volume 132, Springer Science & Business Media, 2006.
[22] KRZYSZTOF C. KIWIEL, *Convergence and efficiency of subgradient methods for quasiconvex minimization,* Math. Program., 90(1) (2001), pp. 1–25.
[23] YANN A. LECUN, LÉON BOTTOU, GENEVIEVE B. ORR, AND KLAUS-ROBERT MÜLLER, *Efficient backprop,* Neural Networks: Tricks of the Trade, pages 9–48, Springer, 2012.
[24] WUYANG LI, XUESHUANG XIANG, AND YINGXIANG XU, *Deep domain decomposition method:*

Elliptic problems, Mathematical and Scientific Machine Learning, pages 269–286, PMLR, 2020.

[25] JULIA LING, ANDREW KURZAWSKI, AND JEREMY TEMPLETON, *Reynolds averaged turbulence modelling using deep neural networks with embedded invariance,* J. Fluid Mech., 807 (2016), pp. 155–166.

[26] JEAN-CLAUDE NÉDÉLEC, *Acoustic and electromagnetic equations: integral representations for harmonic problems*, volume 144, Springer Science & Business Media, 2013.

[27] JORGE NOCEDAL, *Updating quasi-Newton matrices with limited storage,* Math. Comput., 35(151) (1980), pp. 773–782.

[28] MAZIAR RAISSI, PARIS PERDIKARIS, AND GEORGE E. KARNIADAKIS, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,* J. Comput. Phys., 378 (2019), pp. 686–707.

[29] JUSTIN SIRIGNANO AND KONSTANTINOS SPILIOPOULOS, *DGM: A deep learning algorithm for solving partial differential equations,* J. Comput. Phys., 375 (2018), pp. 1339–1364.

[30] VINCENT SITZMANN, JULIEN MARTEL, ALEXANDER BERGMAN, DAVID LINDELL, AND GORDON WETZSTEIN, *Implicit neural representations with periodic activation functions,* Adv. Neural Information Process. Systems, 33 (2020).

[31] ULRICH TROTTENBERG, CORNELIUS W. OOSTERLEE, AND ANTON SCHULLER, Multigrid, Elsevier, 2000.

[32] DEYUE ZHANG, FUMING MA, AND ENXI ZHENG, *A Herglotz wavefunction method for solving the inverse Cauchy problem connected with the Helmholtz equation,* J. Comput. Appl. Math., 237(1) (2013), pp. 215–222.