Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2024-0046

Vol. **38**, No. 1, pp. 248-284
July 2025

Dirichlet-Neumann Learning Algorithm for Solving Elliptic Interface Problems

Qi Sun[1,2], Xuejun Xu[1,2,*] and Haotian Yi[1]

[1] *School of Mathematical Sciences, Tongji University, Shanghai 200092, China.*
[2] *School of Mathematical Sciences, Key Laboratory of Intelligent Computing and Applications (Ministry of Education), Tongji University, Shanghai 200092, China.*

Received 3 March 2024; Accepted (in revised version) 3 June 2024

Abstract. Non-overlapping domain decomposition methods are well-suited for addressing interface problems across various disciplines, where traditional numerical simulations often require the use of interface-fitted meshes or technically designed basis functions. To remove the burden of mesh generation and to effectively tackle with the flux transmission condition, a novel mesh-free scheme, i.e., the Dirichlet-Neumann learning algorithm, is studied in this work for solving the benchmark elliptic interface problems with high-contrast coefficients and irregular interfaces. By resorting to the variational principle, we carry out a rigorous error analysis to evaluate the discrepancy caused by the boundary penalty treatment for each decomposed subproblem, which paves the way for realizing the Dirichlet-Neumann algorithm using neural network extension operators. Through experimental validation on a series of testing problems in two and three dimensions, our methods demonstrate superior performance over other alternatives even in scenarios with inaccurate flux predictions at the interface.

AMS subject classifications: 65N55, 35J20, 68Q32

Key words: Elliptic interface problem, high-contrast coefficient, compensated deep Ritz method, artificial neural networks, deep learning.

1 Introduction

Many problems in science and engineering are carried out with domains separated by curves or surfaces, e.g., the abrupt change in material properties between adjacent regions, from which the interface problem naturally arises. A widely studied benchmark example is the elliptic interface problem with high-contrast coefficients [6, 36, 40], whose solution lies in the Sobolev space $H^{1+\epsilon}(\Omega)$ with $\epsilon>0$ possibly close to zero [44]. Due to the

*Corresponding author. *Email addresses:* qsun_irl@tongji.edu.cn (Q. Sun), xxj@lsec.cc.ac.cn (X. Xu), 2111166@tongji.edu.cn (H. Yi)

limited regularity of solution, classical numerical methods, such as finite difference and finite element methods [3, 34], require the generation of an interface-fitted mesh for the discretization of domain [6], which can be technically involved and time consuming especially when the geometry of interface gets complicated and the dimension of problem increases. To ease the burden of mesh generation, numerical approaches based on unfitted meshes, e.g., the immersed interface method [36] and many others, have emerged as attractive alternatives [40]. However, using unfitted meshes, e.g, a uniform Cartesian mesh, often requires technical adjustments to the basis function to enforce the jump condition with high accuracy [5, 17].

Note that the computational domain has been separated as the union of multiple non-overlapping subdomains, each of which corresponds to a local boundary value problem after endowing the interface with an appropriate boundary condition [52]. As a result, the non-overlapping Dirichlet-Neumann algorithm [52, 68, 69] is developed to address elliptic interface problems, where decomposed subproblems are alternatively solved using mesh-based numerical methods [3, 34, 35]. However, the complex geometry of subdomain interfaces remains a major concern during the meshing process. Fortunately, many domain decomposition methods [9, 52, 60] can be formulated at the continuous level, thereby making it computationally feasible to adopt the meshless deep learning technique [14, 29, 70] as the local problem solver [19]. Thanks to the rapid development of artificial intelligence science, much attention has recently been paid to combining deep learning with insights from domain decomposition methods. The physics-informed neural networks (abbreviated as PINNs in what follows) [29–31, 53], among others [56, 70, 72] has been utilized to solve Dirichlet and Neumann subproblems within the classical Dirichlet-Neumann algorithm [39], which is named "DeepDDM" and applied to several interface problems as a proof of concept. To further enhance its scalability properties, the DeepDDM method is extended with the aid of coarse space correction [45, 52]. Note that in the degenerate case of homogeneous jump conditions, the continuity of averaged solution between neighbouring subdomains, as well as its first and higher-order derivatives, are explicitly enforced through additional penalty terms in a series of papers [22,24,27,43,55,67]. Designing specific network architectures is another way of dealing with the complex geometry and jump condition [10, 21, 61, 64], e.g., using adaptive activation functions [25, 26], augmenting an additional coordinate variable as the input of the solution ansatz [32], replacing neural network structures with extreme learning machines [10, 11] or graph neural networks [59], to name a few. Additionally, an efficient hybrid approach [4, 64] is developed to address the singular and regular solutions using neural network and finite difference methods, respectively.

However, when solving the Dirichlet subproblem using neural networks, the gradient of trained model often exhibits higher errors at the boundary compared to its interior domain, which poses challenges when explicitly enforcing the flux transmission condition along subdomain interfaces. In contrast, a novel Dirichlet-Neumann learning algorithm using neural network extension operators is studied in this work for tackling high-contrast coefficients as well as irregular interfaces, alleviating the issue of inaccurate

Dirichlet-to-Neumann map by resorting to the variational principle. Moreover, a rigorous error analysis is established to estimate the discrepancies caused by the penalty treatment of boundary conditions[†] [7, 47, 71], which also sheds light on the setup of penalty coefficients. Additionally, we present a comparative study to theoretically illustrate the robustness and effectiveness of our methods over the DeepDDM scheme [38], followed by a series of numerical examples to validate our findings.

The rest of this paper is organized as follows. In Section 2, we begin by recalling the Dirichlet-Neumann algorithm for solving elliptic interface problems in the continuous level, then the subproblem solver using mesh-based and mesh-free methods are briefly reviewed and compared. We also conduct experiments to illustrate the motivation behind our work, with a particular focusing on the Dirichlet-to-Neumann map of trained network solutions for the Dirichlet subproblem. To combine domain decomposition methods with deep learning solvers in a consistent manner, a rigorous error analysis of boundary penalty treatment for both Dirichlet and Neumann subproblems is presented in Section 3, followed by implementation details of our Dirichlet-Neumann learning algorithm. Next, numerical results on a series of benchmark problems are reported in Section 4. Finally, we summarize our work in Section 5.

2 Preliminaries and motivation

2.1 Elliptic interface problem with high-contrast coefficients

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain with Lipschitz boundary $\partial\Omega$, which is assumed to be composed of two non-overlapping subdomains as illustrated in Fig. 1, that is,

$$\overline{\Omega} = \overline{\Omega_1 \cap \Omega_2}, \quad \Omega_1 \cap \Omega_2 = \emptyset, \quad \Gamma = \partial\Omega_1 \cap \partial\Omega_2.$$

We consider in this work an elliptic interface problem with high-contrast coefficients and natural jump conditions [40], which is often formally written as:

$$\begin{aligned} -\nabla\cdot(c(x)\nabla u(x)) + u(x) &= f(x) \quad \text{in } \Omega, \\ u(x) &= 0 \quad \text{on } \partial\Omega, \\ [\![u(x)]\!] = 0 \quad \text{and} \quad [\![c(x)\nabla u(x)\cdot \boldsymbol{n}]\!] &= q(x) \quad \text{on } \Gamma. \end{aligned} \tag{2.1}$$

Here, $f(x)$ represents a given function in $L^2(\Omega)$, $\boldsymbol{n} = \boldsymbol{n}_2$ ($\boldsymbol{n}_1$) the unit outer normal vector for subdomain Ω_2 (Ω_1), and $[\![\cdot]\!]$ the difference of quantity across the interface, i.e.,

$$\lim_{x|_{\Omega_1}\to X} u(x) = \lim_{x|_{\Omega_2}\to X} u(x) \quad \text{and} \quad -\lim_{x|_{\Omega_1}\to X} c(x)\nabla u(x)\cdot\boldsymbol{n}_1 - \lim_{x|_{\Omega_2}\to X} c(x)\nabla u(x)\cdot\boldsymbol{n}_2 = q(X)$$

[†]Essential boundary conditions are included as additional penalty terms, posed in a "soft" manner.

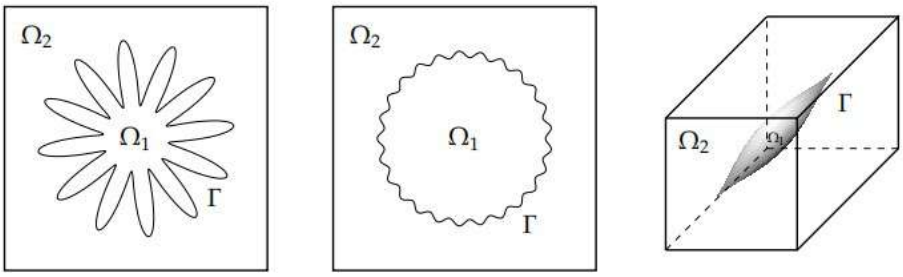


Figure 1: A Lipschitz domain $\Omega\subset\mathbb{R}^2$ or $\mathbb{R}^3$ that is decomposed into two subdregions.

for any point $X\in\Gamma$. In particular, $c(x)$ is a piecewise constant function that has a finite jump of function value across the interface Γ, that is,

$$c(x)=\begin{cases} c_1>0 & \text{in } \Omega_1,\\ c_2\gg c_1 & \text{in } \Omega_2,\end{cases}$$

which is typically caused by the abrupt change in material properties or the interaction of fluid dynamics [40]. Notably, $c_1=c_2$ can be regarded as a degenerate case of (2.1). By setting $V_i=\{v_i\in H^1(\Omega_i)\,|\,v_i|_{\partial\Omega\cap\partial\Omega_i}=0\}$, $V_i^0=H_0^1(\Omega_i)$, and defining

$$b_i(u_i,v_i)=\int_{\Omega_i}(c_i\nabla u_i\cdot\nabla v_i+u_iv_i)dx,\ (f,v_i)_i=\int_{\Omega_i}fv_idx,\text{ and }(q,v_2)_{L^2(\Gamma)}=\int_\Gamma qvds,$$

for $i=1,2$, the Green's formula implies that the weak formulation of interface problem (2.1) reads: find $u_1\in V_1$ and $u_2\in V_2$ such that

$$\begin{aligned} b_1(u_1,v_1)&=(f,v_1)_1 && \text{for any } v_1\in V_1^0,\\ u_1&=u_2 && \text{on } \Gamma,\\ b_2(u_2,v_2)&=(f,v_2)_2+(f,R_1\gamma_0v_2)_1-b_1(u_1,R_1\gamma_0v_2)-(q,v_2)_{L^2(\Gamma)} && \text{for any } v_2\in V_2,\end{aligned}\tag{2.2}$$

where $\gamma_0v=v|_\Gamma$ is the restriction of $v\in H^1(\Omega_i)$ on the interface Γ and $R_i:H_{00}^{\frac{1}{2}}(\Gamma)\to V_i$ any differentiable extension operator.

By choosing a suitable relaxation parameter $\rho\in(0,\rho_{\max})$, an iterative scheme, known as the Dirichlet-Neumann algorithm [52, 68], can be established for solving (2.2): given an initial guess of the unknown solution at the interface $u_\Gamma^{[0]}\in H_{00}^{\frac{1}{2}}(\Gamma)$, then solve for $k\geq0$,

$$1)\ u_1^{[k]}=\underset{u_1\in V_1,\ u_1|_\Gamma=u_\Gamma^{[k]}}{\operatorname{argmin}}\frac{1}{2}b_1(u_1,u_1)-(f,u_1)_1;\tag{2.3}$$

$$2)\ u_2^{[k]}=\underset{u_2\in V_2}{\operatorname{argmin}}\frac{1}{2}b_2(u_2,u_2)-(f,u_2)_2+b_1(u_1^{[k]},R_1\gamma_0u_2)-(f,R_1\gamma_0u_2)_1+(q,u_2)_{L^2(\Gamma)};\tag{2.4}$$

$$3)\ \ u_\Gamma^{[k+1]} = \rho u_2^{[k]} + (1-\rho) u_\Gamma^{[k]} \quad \text{on } \Gamma, \tag{2.5}$$

until certain stopping criteria are met [52, 60]. Here, the flux transmission between (2.3) and (2.4) is enforced without explicitly computing the Dirichlet-to-Neumann map [58].

Remark 2.1. Though the entire solution of problem (2.1) lies in the space $H^{1+\epsilon}(\Omega)$ with $\epsilon > 0$ possibly close to zero [44], solutions of decomposed subproblems (2.3) and (2.4) are typically regular under mild geometric assumptions on the boundary [11, 13]. More precisely, we assume that $u_1^{[k]} \in H^2(\Omega_1)$ and $u_2^{[k]} \in H^2(\Omega_2)$ for error estimates in Section 3.

Remark 2.2. In the case of an inhomogeneous jump condition $[\![u(x)]\!] = p(x) \neq 0$ in (2.1), the update of solution value at the interface (2.5) turns out to be [1, 68]

$$u_\Gamma^{[k+1]} = \rho u_2^{[k]} + (1-\rho) u_\Gamma^{[k]} + p \quad \text{on } \Gamma.$$

2.2 Related work

Traditional numerical methods for tackling elliptic interface problems can be roughly categorized into two groups by using either an interface-fitted or -unfitted mesh in the discretization of domain. Provided a mesh that aligns precisely with the boundary of each subdomain, the former enables absorption of interface jump conditions into the finite element formulation [3], leading to accurate approximations with nearly optimal error bounds [6]. However, it necessitates the requirement of an interface-fitted mesh generator, which can be time consuming for intricate geometries in two and higher dimensions. In contrast, the latter employs interface-unfitted meshes (e.g., the Cartesian mesh) and is renowned for its easiness of mesh generation, but necessitates the construction of basis functions that fulfill the interface jump conditions. A rich literature in this direction includes immersed interface methods [36], extended finite element methods [12], to name a few. In recent years, there has also been a rapid emergence of methods that integrate classical numerical methods with contemporary deep learning techniques, which have achieved success to a certain extent.

For instance, the Dirichlet-Neumann algorithm illustrated in section 2.1, which is often expressed in terms of differential operators [60], namely, for $k \geq 1$,

$$\begin{cases} -\nabla\cdot(c_1\nabla u_1^{[k]}) + u_1^{[k]} = f & \text{in } \Omega_1, \\ u_1^{[k]} = 0 & \text{on } \partial\Omega_1 \setminus \Gamma, \\ u_1^{[k]} = \rho u_2^{[k-1]} + (1-\rho) u_1^{[k-1]} & \text{on } \Gamma, \end{cases} \quad \text{(Dirichlet subproblem)}$$

$$\begin{cases} -\nabla\cdot(c_2\nabla u_2^{[k+1]}) + u_2^{[k+1]} = f & \text{in } \Omega_2, \\ u_2^{[k+1]} = 0 & \text{on } \partial\Omega_2 \setminus \Gamma, \\ c_2\nabla u_2^{[k+1]}\cdot \boldsymbol{n}_2 = -q - c_1\nabla u_1^{[k]}\cdot \boldsymbol{n}_1 & \text{on } \Gamma, \end{cases} \quad \text{(Neumann subproblem)} \tag{2.6}$$

is also a continuous method for solving the elliptic interface problem (2.1), thereby making it feasible to integrate with techniques from the deep learning community [14, 29]. Specifically, one of the most straightforward method [39] is to employ PINNs [53] for solving all the decomposed subproblems in (2.6). It is also worthwhile to note that similar idea has been applied in areas of overlapping domain decomposition methods [37–39,54], where updated interface conditions are of Dirichlet type and are taken from the interior of neighbouring subdomains.

In fact, the advantage of mesh-free feature has spurred the proposal of various neural network models for addressing the elliptic interface problem (2.1). A deep Ritz-type approach is developed in [63] through the usage of one single neural network, whereas an improved procedure involves employing a piecewise neural network on multiple subdomains [15, 17]. In addition, configuring appropriate penalty weights among various loss terms is a critical yet laborious task, which can be mitigated by assigning adaptive weights to different loss terms [62,65]. Note that in the case of homogeneous jump conditions, the continuity of averaged solution between neighbouring subdomains, as well as its first and higher-order derivatives [22,24,27,55,67], can be explicitly enforced through additional penalty terms posed on the interface. A convergence analysis of PINNs in conjunction with domain decomposition techniques for solving elliptic interface problems has recently been established under sufficient regularity assumptions [66]. Another way of dealing with the jump conditions is to design specific network architectures that can capture jump discontinuities across subdomain interfaces [21, 32, 61, 64] or employ hybrid schemes that can take both advantages of neural network and finite difference methods [4, 64].

2.3 Motivation of our work

Undoubtedly, flux transmission between neighboring subdomains plays a crucial role in (2.6), however, gradients of trained models often exhibit higher errors at the boundary compared to its interior domain [1, 8, 58, 62], which may degrade the numerical performance. As an illustrative example, we consider the Dirichlet subproblem[†] in (2.6) with

$$\Omega_1=\left\{(x,y)\,\middle|\,\sqrt{x^2+y^2}\le\frac{1}{2}+\frac{1}{4}\sin\left(12\arctan\left(\frac{y}{x}\right)\right)\right\},\qquad \Gamma=\partial\Omega_1,$$

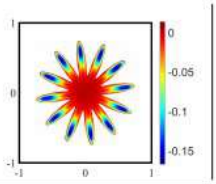
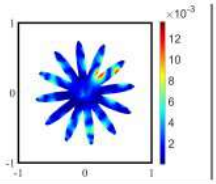
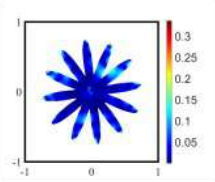
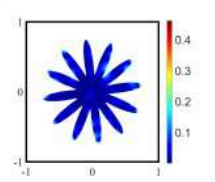
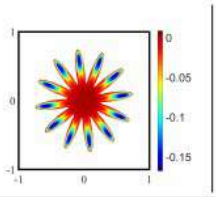
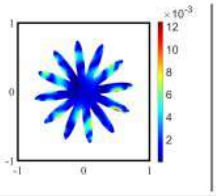
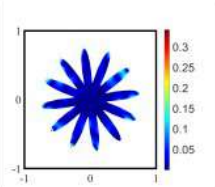
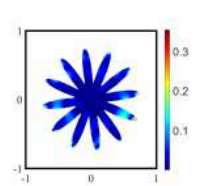
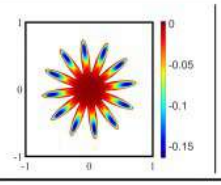
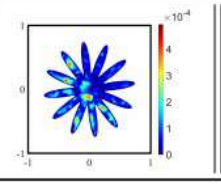
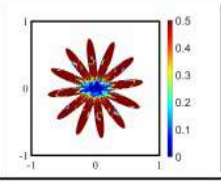
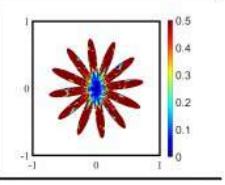
$c_1=1$, $f(x,y)$ and $u_1(x,y)|_\Gamma$ being derived from the exact solution

$$u_1(x,y)=(x^2+y^2)^{\frac{3}{2}}\sin\left(2\pi\sqrt{x^2+y^2}-\pi-\frac{\pi}{2}\sin\left(12\arctan\left(\frac{y}{x}\right)\right)\right).$$

We begin our exploration by studying the soft-constrained PINNs approach [53], adopting the ResNet structure (depth = 8, width = 100) [18] with a constant penalty coefficient

[†]Here, the superscript $[k]$ of $u_i^{[k]}$ is omitted for brevity and $\hat{u}_i$ indicates the network solution for $i=1,2$.

Table 1: Trained models $\hat{u}_1$ of Dirichlet subproblem using different strategies, as well as their error profiles.

	$\hat{u}_1$	$\lvert u_1-\hat{u}_1\rvert$	$\lvert\partial_x(u_1-\hat{u}_1)\rvert$	$\lvert\partial_y(u_1-\hat{u}_1)\rvert$
ResNet soft-constraint $\beta\equiv 400$ [18]				
Transformer soft-constraint adaptive β [62]				
ResNet hard-constraint [57]				

for solving the Dirichlet subproblem. As can be observed from the error profile $|u_1-\hat{u}_1|$ in Table 1, the trained network solution closely aligns with the true solution. However, the accuracy of derivatives $\partial_x\hat{u}_1$ and $\partial_y\hat{u}_1$, particularly at and near the boundary, is not that satisfactory. Next, we use the transformer network architecture, accompanied by an adaptive strategy for fine-tuning the penalty coefficient [62]. Numerical results shown in Table 1 reveal similar error patterns as before, with a marginally improved performance. Finally, we conduct an experiment using the hard-constraint strategy [57], where the satisfactory of Dirichlet boundary conditions is significantly improved. Unfortunately, accuracy in the corresponding derivative values deteriorates as shown in Table 1.

Consequently, when solving the Neumann subproblem through (2.6), the flux transmission condition is approximately enforced by

$$c_2\nabla\hat{u}_2\cdot\boldsymbol{n}_2\approx -q-c_1\nabla\hat{u}_1\cdot\boldsymbol{n}_1 \quad \text{on } \Gamma,$$

which ends up with the following error estimation

$$\nabla\hat{u}_2-\nabla u_2\approx\frac{c_1}{c_2}\left(\nabla\hat{u}_1-\nabla u_1\right) \quad \text{on } \Gamma. \tag{2.7}$$

This indicates that the error incurred by $\nabla\hat{u}_1-\nabla u_1$ along the interface, as shown in Table 1, may propagate to neighbouring subdomains through (2.7). Such an issue is often overlooked due to the impact of high-contrast coefficients $c_1\ll c_2$ but should not be disregarded especially when c_1 and c_2 are of comparable magnitude.

Moreover, considering the experimental observation that the Dirichlet-to-Neumann map of trained network solution[‡] often exhibits higher errors at the interface, it also motivates us to use the interior solution $\nabla\hat{u}_1|_{\Omega_1}$ for flux exchange rather than the straightforward approach in (2.6).

3 Method

In this section, we begin by revisiting variational problems (2.3) and (2.4) from the perspective of network training. Then, the convergence analysis of our Dirichlet-Neumann learning algorithm for solving (2.1) is studied in the weak sense, which also sheds light on the setup of penalty coefficients during the training process. Finally, implementation details for realizing our learning approach is summarized in Algorithm 1, accompanied by discussions on additional tricks to further enhance the numerical performance.

3.1 Variational problem revisited

To realize the iterative scheme (2.3-2.5) using neural networks [14], the essential boundary condition of Dirichlet subproblem (2.3) is treated in a "soft" manner by augmenting the energy functional with boundary penalty terms [29,56,70], namely,

$$\hat{u}_1^{[k]} = \underset{\hat{u}_1\in H^1(\Omega_1)}{\operatorname{argmin}} \frac{1}{2}b_1(\hat{u}_1,\hat{u}_1) - (f,\hat{u}_1)_1 + \frac{\beta_D}{2}\left(\|\hat{u}_1\|^2_{L^2(\partial\Omega_1\cap\partial\Omega)} + \|\hat{u}_1 - u_\Gamma^{[k]}\|^2_{L^2(\Gamma)}\right), \tag{3.1}$$

at the k-th outer iteration, where $\beta_D>0$ is a user-defined penalty coefficient [62].

On the other hand, by extending the local solution $u_2\in V_2$ of Neumann subproblem (2.4) to its neighboring subdomain (not relabelled for notational simplicity),

$$R_1\gamma_0 u_2(x) = u_2(x)\in V_1, \tag{3.2}$$

the modified loss functional associated with our Neumann subproblem (2.4) gives

$$\begin{aligned}\hat{u}_2^{[k]} = \underset{\hat{u}_2\in H^1(\Omega)}{\operatorname{argmin}} &\frac{1}{2}b_2(\hat{u}_2,\hat{u}_2) - (f,\hat{u}_2)_2 + b_1(\hat{u}_1^{[k]},\hat{u}_2) - (f,\hat{u}_2)_1 + \frac{\beta_N}{2}\|\hat{u}_2\|^2_{L^2(\partial\Omega)} \\ &+ (q,\hat{u}_2)_{L^2(\Gamma)},\end{aligned} \tag{3.3}$$

where $\beta_N>0$ represents another penalty coefficient. Obviously, the interior data $\nabla\hat{u}_1^{[k]}|_{\Omega_1}$ is employed for flux exchange in (3.3), rather than the Neumann trace $\nabla\hat{u}_1^{[k]}|_\Gamma$. It is also noteworthy that the minimizer of functional (3.3) is now defined globally over the entire domain, which differs from the traditional mesh-based treatment [60].

[‡]Here and in what follows, we exclude the hard-constraint strategy and defer it for future investigation.

3.2 Error estimates

Before introducing the neural network parametrization of unknown solutions, the error estimation induced by the relaxation from exact boundary or interface conditions to a penalization-based approach are established for (3.1) and (3.3) in what follows.

Theorem 3.1. *Let $u_1^{[k]}$ and $\hat{u}_1^{[k]}$ be the solution of minimization problems* (2.3) *and* (3.1) *respectively, then there holds*

$$\|\hat{u}_1^{[k]}-u_1^{[k]}\|_{H^1(\Omega_1)}\leq C(\Omega_1,u_1^{[k]})\frac{c_1}{\beta_D}\sqrt{\frac{\hat{c}_1}{\check{c}_1}}, \tag{3.4}$$

where $\hat{c}_1=\max\{c_1,1\}$, $\check{c}_1=\min\{c_1,1\}$, and $C(\Omega_1,u_1^{[k]})$ represents a generic constant that depends on the subdomain Ω_1 and the solution $u_1^{[k]}$ of Dirichlet subproblem (2.3).

Proof. Step 1) We first denote by $L_1(\hat{u}_1)$ the loss function of problem (3.1), that is, a functional on $H^1(\Omega_1)$ taking on the form

$$\mathcal{L}_1(\hat{u}_1)=\frac{1}{2}b_1(\hat{u}_1,\hat{u}_1)-(f,\hat{u}_1)_1+\frac{\beta_D}{2}\left(\|\hat{u}_1\|^2_{L^2(\partial\Omega_1\cap\partial\Omega)}+\|\hat{u}_1-u_\Gamma^{[k]}\|^2_{L^2(\Gamma)}\right), \tag{3.5}$$

and then derive optimality conditions that are satisfied by the unique global minimizer. To be precise, the function $\hat{u}_1\in H^1(\Omega_1)$ is decomposed as a sum of two local functions, i.e., $\hat{u}_1=\hat{u}_1^{[k]}+g$ with $\hat{u}_1^{[k]}\in H^1(\Omega_1)$ satisfying

$$\begin{cases} -\nabla\cdot(c_1\nabla\hat{u}_1^{[k]})+\hat{u}_1^{[k]}=f & \text{in } \Omega_1,\\ \hat{u}_1^{[k]}+c_1\beta_D^{-1}\nabla\hat{u}_1^{[k]}\cdot\boldsymbol{n}_1=0 & \text{on } \partial\Omega_1\cap\partial\Omega,\\ \hat{u}_1^{[k]}+c_1\beta_D^{-1}\nabla\hat{u}_1^{[k]}\cdot\boldsymbol{n}_1=u_\Gamma^{[k]} & \text{on } \Gamma, \end{cases} \tag{3.6}$$

in the sense of distributions. Then, by applying the Green's formula [11] to (3.6), a direct calculation of (3.5) implies that* for any $\hat{u}_1\in H^1(\Omega_1)$,

$$\mathcal{L}_1(\hat{u}_1)=\mathcal{L}_1(\hat{u}_1^{[k]})+\int_{\Omega_1}\left(\frac{c_1}{2}|\nabla g|^2+\frac{1}{2}|g|^2\right)dx+\frac{\beta_D}{2}\int_{\partial\Omega_1}|g|^2ds\geq\mathcal{L}_1(\hat{u}_1^{[k]}).$$

Or, equivalently, the unique weak solution of elliptic equations (3.6) is the global minimizer of energy functional (3.5). Notably, when comparing (3.6) with the Dirichlet subproblem (2.3) (written in terms of differential operators), i.e.,

$$\begin{cases} -\nabla\cdot(c_1\nabla u_1^{[k]})+u_1^{[k]}=f & \text{in } \Omega_1,\\ u_1^{[k]}=0 & \text{on } \partial\Omega_1\cap\partial\Omega,\\ u_1^{[k]}=u_\Gamma^{[k]} & \text{on } \Gamma, \end{cases} \tag{3.7}$$

*More details can be found in the technical **Appendix A**.

the Dirichlet boundary condition is modified to be of a Robin type due to the boundary penalty treatment in (3.1).

Step 2) Now, we are ready to quantitatively estimate the error induced from the soft boundary enforcement, that is, the distance between weak solutions of (3.7) and (3.6). To deal with the inhomogeneous boundary conditions in (3.7) and (3.6), let us write $u_1^{[k]}=w_1+g_1$ with an extension $g_1\in V_1$ of $u_\Gamma^{[k]}$ into Ω_1 [13], namely,

$$\begin{cases} -\nabla\cdot(c_1\nabla w_1)+w_1=f & \text{in } \Omega_1,\\ w_1=0 & \text{on } \partial\Omega_1, \end{cases} \quad \begin{cases} -\nabla\cdot(c_1\nabla g_1)+g_1=0 & \text{in } \Omega_1,\\ g_1=0 & \text{on } \partial\Omega_1\cap\partial\Omega,\\ g_1=u_\Gamma^{[k]} & \text{on } \Gamma, \end{cases} \tag{3.8}$$

and $\hat{u}_1^{[k]}=\hat{w}_1+\hat{g}_1$ with another extension $\hat{g}_1\in H^1(\Omega_1)$ of $u_\Gamma^{[k]}$ into Ω_1, that is,

$$\begin{cases} -\nabla\cdot(c_1\nabla\hat{w}_1)+\hat{w}_1=f & \text{in } \Omega_1,\\ \hat{w}_1+c_1\beta_D^{-1}\nabla\hat{w}_1\cdot\boldsymbol{n}_1=0 & \text{on } \partial\Omega_1, \end{cases} \quad \begin{cases} -\nabla\cdot(c_1\nabla\hat{g}_1)+\hat{g}_1=0 & \text{in } \Omega_1,\\ \hat{g}_1+c_1\beta_D^{-1}\nabla\hat{g}_1\cdot\boldsymbol{n}_1=0 & \text{on } \partial\Omega_1\cap\partial\Omega,\\ \hat{g}_1+c_1\beta_D^{-1}\nabla\hat{g}_1\cdot\boldsymbol{n}_1=u_\Gamma^{[k]} & \text{on } \Gamma, \end{cases} \tag{3.9}$$

then it immediately follows from the triangle inequality that

$$\|\hat{u}_1^{[k]}-u_1^{[k]}\|_{H^1(\Omega_1)}=\|(\hat{w}_1+\hat{g}_1)-(w_1+g_1)\|_{H^1(\Omega_1)}\le\|\hat{w}_1-w_1\|_{H^1(\Omega_1)}+\|\hat{g}_1-g_1\|_{H^1(\Omega_1)}.$$

Step 3) Based on the variational formulation, the extension function g_1 in (3.8) satisfies

$$b_1(g_1,\hat{g}_1)=\int_{\Omega_1}(c_1\nabla g_1\cdot\nabla\hat{g}_1+g_1\hat{g}_1)\,dx=\int_{\partial\Omega_1}(c_1\nabla g_1\cdot\boldsymbol{n_1})\hat{g}_1\,ds=(c_1\nabla g_1\cdot\boldsymbol{n_1},\hat{g}_1)_{L^2(\partial\Omega_1)},$$

with $\hat{g}_1\in H^1(\Omega_1)$ being used as the test function, while the extension function $\hat{g}_1$ of (3.9) is the minimizer of energy functional*

$$\begin{aligned}\mathcal{F}_1(\hat{g}_1)=&\frac{1}{2}b_1(\hat{g}_1,\hat{g}_1)+\frac{\beta_D}{2}\left(\|\hat{g}_1\|^2_{L^2(\partial\Omega_1\cap\partial\Omega)}+(\hat{g}_1-2u_\Gamma^{[k]},\hat{g}_1)_{L^2(\Gamma)}\right)-b_1(g_1,\hat{g}_1)\\&+(c_1\nabla g_1\cdot\boldsymbol{n_1},\hat{g}_1)_{L^2(\partial\Omega_1)}\\=&\frac{1}{2}b_1(\hat{g}_1-g_1,\hat{g}_1-g_1)+\frac{\beta_D}{2}\Big(\|\hat{g}_1+c_1\beta_D^{-1}\nabla g_1\cdot\boldsymbol{n}_1\|^2_{L^2(\partial\Omega_1\cap\partial\Omega)}\\&+\|\hat{g}_1+c_1\beta_D^{-1}\nabla g_1\cdot\boldsymbol{n}_1-u_\Gamma^{[k]}\|^2_{L^2(\Gamma)}\Big)-\frac{1}{2}b_1(g_1,g_1)\\&-\frac{\beta_D}{2}\left(\|c_1\beta_D^{-1}\nabla g_1\cdot\boldsymbol{n}_1\|^2_{L^2(\partial\Omega_1\cap\partial\Omega)}+\|c_1\beta_D^{-1}\nabla g_1\cdot\boldsymbol{n}_1-u_\Gamma^{[k]}\|^2_{L^2(\Gamma)}\right),\end{aligned}$$

*Here, we do not distinguish between the non-optimal and the optimal solution for notational simplicity.

and from which we can conclude that the function $\hat{g}_1$ is also the minimizer of

$$\begin{aligned}\mathcal{G}_1(\hat{g}_1)=&\frac{1}{2}b_1(\hat{g}_1-g_1,\hat{g}_1-g_1)+\frac{\beta_D}{2}\Big(\|\hat{g}_1+c_1\beta_D^{-1}\nabla g_1\cdot\boldsymbol{n}_1\|^2_{L^2(\partial\Omega_1\cap\partial\Omega)}\\&+\|\hat{g}_1+c_1\beta_D^{-1}\nabla g_1\cdot\boldsymbol{n}_1-u_\Gamma^{[k]}\|^2_{L^2(\Gamma)}\Big).\end{aligned}$$

On the one hand, by defining $\check{c}_1=\min\{c_1,1\}$, it is obvious that

$$\begin{aligned}\mathcal{G}_1(\hat{g}_1)\geq&\frac{1}{2}b_1(\hat{g}_1-g_1,\hat{g}_1-g_1)=\int_{\Omega_1}\left(\frac{c_1}{2}|\nabla(\hat{g}_1-g_1)|^2+\frac{1}{2}|\hat{g}_1-g_1|^2\right)dx\\\geq&\frac{\check{c}_1}{2}\|\hat{g}_1-g_1\|^2_{H^1(\Omega_1)}.\end{aligned}\tag{3.10}$$

On the other hand, due to the fact that $g_1\in H^2(\Omega_1)$ under mild assumptions [11], we have by using the trace theorem [41] that $(\nabla g_1\cdot\boldsymbol{n}_1)|_{\partial\Omega_1}\in H^{\frac{1}{2}}(\partial\Omega_1)$ and therefore there exists a function $\phi\in H^1(\Omega_1)$ such that $\phi|_{\partial\Omega_1}=-(\nabla g_1\cdot\boldsymbol{n}_1)|_{\partial\Omega_1}$.

Then, by choosing $\bar{g}=c_1\beta_D^{-1}\phi+g_1$ and using boundary conditions of g_1 (3.8), the optimality of $\hat{g}_1$ among all functions in $H^1(\Omega_1)$ implies that

$$\begin{aligned}\mathcal{G}_1(\hat{g}_1)\leq\mathcal{G}_1(\bar{g})=&c_1^2\beta_D^{-2}\int_{\Omega_1}\left(\frac{c_1}{2}|\nabla\phi|^2+\frac{1}{2}|\phi|^2\right)dx\\&+\frac{\beta_D}{2}\left(\|g_1\|^2_{L^2(\partial\Omega_1\cap\partial\Omega)}+\|g_1-u_\Gamma^{[k]}\|^2_{L^2(\Gamma)}\right)\leq\frac{\hat{c}_1}{2}c_1^2\beta_D^{-2}\|\phi\|^2_{H^1(\Omega_1)},\end{aligned}$$

where $\hat{c}_1=\max\{c_1,1\}$. As a direct result, we have by (3.10) that

$$\|\hat{g}_1-g_1\|_{H^1(\Omega_1)}\leq\frac{c_1}{\beta_D}\sqrt{\frac{\hat{c}_1}{\check{c}_1}}\|\phi\|_{H^1(\Omega_1)}.$$

Step 4) It remains to show that the solution $\hat{w}_1\in H^1(\Omega_1)$ of Robin problem (3.9) can converge to the solution $w_1\in H_0^1(\Omega_1)$ of Dirichlet problem (3.8) as $\beta_D\to\infty$ [7,71]. Similar as before, by employing $\hat{w}_1\in H^1(\Omega_1)$ as the test function in (3.8) and resorting to the variational problem of (3.9), the weak solution $\hat{w}_1\in H^1(\Omega_1)$ minimizes

$$\begin{aligned}\mathcal{I}_1(\hat{w}_1)=&\frac{1}{2}b_1(\hat{w}_1,\hat{w}_1)-(f,\hat{w}_1)_1+\frac{\beta_D}{2}\|\hat{w}_1\|^2_{L^2(\partial\Omega_1)}-b_1(w_1,\hat{w}_1)+(f,\hat{w}_1)\\&+(c_1\nabla w_1\cdot\boldsymbol{n_1},\hat{w}_1)_{L^2(\partial\Omega_1)}\\=&\frac{1}{2}b_1(\hat{w}_1-w_1,\hat{w}_1-w_1)+\frac{\beta_D}{2}\|\hat{w}_1+c_1\beta_D^{-1}\nabla w_1\cdot\boldsymbol{n}_1\|^2_{L^2(\partial\Omega_1)}\\&-\frac{1}{2}b_1(w_1,w_1)-\frac{\beta_D}{2}\|c_1\beta_D^{-1}\nabla w_1\cdot\boldsymbol{n}_1\|^2_{L^2(\partial\Omega_1)},\end{aligned}$$

and therefore is also the minimizer of the energy functional

$$\mathcal{J}_1(\hat{w}_1)=\frac{1}{2}b_1(\hat{w}_1-w_1,\hat{w}_1-w_1)+\frac{\beta_D}{2}\|\hat{w}_1+c_1\beta_D^{-1}\nabla w_1\cdot\boldsymbol{n}_1\|^2_{L^2(\partial\Omega_1)}.$$

Note that $w_1 \in H^2(\Omega_1)$ under the same geometric assumption, the trace theorem implies that $(\nabla w_1 \cdot \boldsymbol{n}_1)|_{\partial\Omega_1} \in H^{\frac{1}{2}}(\partial\Omega_1)$ and therefore there exists a function $\varphi \in H^1(\Omega_1)$ such that $\varphi|_{\partial\Omega_1} = -(\nabla w_1 \cdot \boldsymbol{n}_1)|_{\partial\Omega_1}$ [41]. As a consequence, by employing a particular function $\bar{w} = c_1\beta_D^{-1}\varphi + w_1 \in H^1(\Omega_1)$ and using the boundary condition of w_1 in (3.8), we have

$$\begin{aligned}\mathcal{J}_1(\hat{w}_1) \le \mathcal{J}_1(\bar{w}) &= c_1^2\beta_D^{-2}\int_{\Omega_1}\left(\frac{c_1}{2}|\nabla\varphi|^2 + \frac{1}{2}|\varphi|^2\right)dx + \frac{\beta_D}{2}\|w_1\|^2_{L^2(\partial\Omega_1)} \\ &\le \frac{\hat{c}_1}{2}c_1^2\beta_D^{-2}\|\varphi\|^2_{H^1(\Omega_1)}.\end{aligned}$$

On the other hand, it is obvious that

$$\begin{aligned}\mathcal{J}_1(\hat{w}_1) \ge \frac{1}{2}b_1(\hat{w}_1 - w_1, \hat{w}_1 - w_1) &= \int_{\Omega_1}\left(\frac{c_1}{2}|\nabla(\hat{w}_1 - w_1)|^2 + \frac{1}{2}|\hat{w}_1 - w_1|^2\right)dx \\ &\ge \frac{\check{c}_1}{2}\|\hat{w}_1 - w_1\|^2_{H^1(\Omega_1)},\end{aligned}$$

which leads to the error estimation

$$\|\hat{w}_1 - w_1\|_{H^1(\Omega_1)} \le \frac{c_1}{\beta_D}\sqrt{\frac{\hat{c}_1}{\check{c}_1}}\|\varphi\|_{H^1(\Omega_1)},$$

that completes the proof. □

Theorem 3.2. *Assume that $\beta_D \to \infty$ in* (3.1) *(or* **Theorem 3.1***), let $u_2^{[k]}$ and $\hat{u}_2^{[k]}$ be the solution of minimization problems* (2.4) *and* (3.3) *respectively, then there holds*[*]

$$\|\hat{u}_2^{[k]}|_{\Omega_2} - u_2^{[k]}\|_{H^1(\Omega_2)} \le C(\Omega_2, u_2^{[k]})\frac{c_2}{\beta_N}\sqrt{\frac{\hat{c}_2}{\check{c}_2}}, \tag{3.11}$$

where $\hat{c}_2 = \max\{c_2, 1\}$, $\check{c}_2 = \min\{c_2, 1\}$, and $C(\Omega_2, u_2^{[k]})$ represents a generic constant that depends on the subdomain Ω_2 and the solution $u_2^{[k]}$ of Neumann subproblem (2.4).

Proof. Step 1) We first denote by $L_2(\hat{u}_2)$ the loss function of problem (3.3), that is, a functional on $H^1(\Omega)$

$$\mathcal{L}_2(\hat{u}_2) = \frac{1}{2}b_2(\hat{u}_2, \hat{u}_2) - (f, \hat{u}_2)_2 + b_1(\hat{u}_1^{[k]}, \hat{u}_2) - (f, \hat{u}_2)_1 + (q, \hat{u}_2)_{L^2(\Gamma)} + \frac{\beta_N}{2}\|\hat{u}_2\|^2_{L^2(\partial\Omega)}, \tag{3.12}$$

and then derive optimality conditions that are satisfied by its global minimizer. Notably, the function $\hat{u}_2 \in H^1(\Omega)$ is defined over the entire domain, which greatly differs from the standard Neumann subproblem (2.4) that only depends on the subdomain Ω_2.

[*]Here, the minimizer $\hat{u}_2^{[k]} \in H^1(\Omega)$ of functional (3.3) is restricted on the subdomain Ω_2 (denoted by $\hat{u}_2^{[k]}|_{\Omega_2}$).

As such, we decompose $\hat{u}_2 \in H^1(\Omega)$ as a sum of two global functions, namely, $\hat{u}_2 = \hat{u}_2^{[k]} + g$, where the restriction of $\hat{u}_2^{[k]} \in H^1(\Omega)$ on subdomain Ω_2 (not relabelled) is required to satisfy the equations

$$\begin{cases} -\nabla\cdot(c_2\nabla\hat{u}_2^{[k]}) + \hat{u}_2^{[k]} = f & \text{in } \Omega_2, \\ \hat{u}_2^{[k]} + c_2\beta_N^{-1}\nabla\hat{u}_2^{[k]}\cdot\boldsymbol{n}_2 = 0 & \text{on } \partial\Omega_2\cap\partial\Omega, \\ c_2\nabla\hat{u}_2^{[k]}\cdot\boldsymbol{n}_2 = -q - c_1\nabla\hat{u}_1^{[k]}\cdot\boldsymbol{n}_1 & \text{on } \Gamma, \end{cases} \tag{3.13}$$

in the sense of distributions. The extension of function $\hat{u}_2^{[k]}|_{\Omega_2}$ to the other subdomain Ω_1 is required to be weakly differentiable and to satisfy the Robin boundary condition

$$\hat{u}_2^{[k]} + c_1\beta_N^{-1}\nabla\hat{u}_1^{[k]}\cdot\boldsymbol{n}_1 = 0 \quad \text{on } \partial\Omega_1\cap\partial\Omega, \tag{3.14}$$

in the sense of distributions. Then, by applying the Green's formula to equations (3.13), (3.6), and using the jump condition (2.1), it can be deduced directly from (3.12) that*

$$\mathcal{L}_2(\hat{u}_2) = \mathcal{L}_2(\hat{u}_2^{[k]}) + \int_{\Omega_2}\left(\frac{c_2}{2}|\nabla g|^2 + \frac{1}{2}|g|^2\right)dx + \frac{\beta_N}{2}\int_{\partial\Omega}|g|^2 ds \geq \mathcal{L}_2(\hat{u}_2^{[k]}),$$

namely, the global minimizer of (3.12) can be characterized by the function $\hat{u}_2^{[k]} \in H^1(\Omega)$ that satisfies (3.13) and (3.14). It's worth noting that only the restricted solution $\hat{u}_2^{[k]}|_{\Omega_2} \in H^1(\Omega_2)$, or equivalently, the weak solution of subproblem (3.13) is relevant for the following error estimation, which is still denoted by $\hat{u}_2^{[k]}$ for short in the remaining of this proof. It is also noteworthy that when compared to the original Neumann subproblem (2.4) (written in terms of differential operators), that is,

$$\begin{cases} -\nabla\cdot(c_2\nabla u_2^{[k]}) + u_2^{[k]} = f & \text{in } \Omega_2, \\ u_2^{[k]} = 0 & \text{on } \partial\Omega_2\cap\partial\Omega, \\ c_2\nabla u_2^{[k]}\cdot\boldsymbol{n}_2 = -q - c_1\nabla u_1^{[k]}\cdot\boldsymbol{n}_1 & \text{on } \Gamma, \end{cases} \tag{3.15}$$

a Robin boundary condition is imposed on $\partial\Omega_2\cap\partial\Omega$ instead of the Dirichlet type, while the interface condition can be rigorously maintained as $\beta_D \to \infty$ in (3.1) or **Theorem 3.1**.

Step 2) To simplify the error analysis, we assume that the parameter $\beta_D \to \infty$, namely, $\hat{u}_1^{[k]} = u_1^{[k]}$ in (3.13) and (3.15). With the weak solution $\hat{u}_2^{[k]} \in H^1(\Omega_2)$ being used as the test function, the integration by parts for subproblem (3.15) implies that

$$b_2(u_2^{[k]}, \hat{u}_2^{[k]}) + (q + c_1\nabla u_1^{[k]}\cdot\boldsymbol{n}_1, \hat{u}_2^{[k]})_{L^2(\Gamma)} - (c_2\nabla u_2^{[k]}\cdot\boldsymbol{n}_2, \hat{u}_2^{[k]})_{L^2(\partial\Omega_2\cap\partial\Omega)} = (f, \hat{u}_2^{[k]})_2,$$

*More details can be found in the technical **Appendix B**.

which can be employed to reformulate the energy functional of subproblem (3.13), i.e.,

$$\begin{aligned}\mathcal{F}_2(\hat{u}_2^{[k]}) &= \frac{1}{2}b_2(\hat{u}_2^{[k]},\hat{u}_2^{[k]}) - (f,\hat{u}_2^{[k]})_2 + (q + c_1\nabla\hat{u}_1^{[k]}\cdot\boldsymbol{n}_1,\hat{u}_2^{[k]})_{L^2(\Gamma)} + \frac{\beta_N}{2}\|\hat{u}_2^{[k]}\|^2_{L^2(\partial\Omega_2\cap\partial\Omega)} \\ &= \frac{1}{2}b_2(\hat{u}_2^{[k]},\hat{u}_2^{[k]}) - b_2(u_2^{[k]},\hat{u}_2^{[k]}) + (c_2\nabla u_2^{[k]}\cdot\boldsymbol{n}_2,\hat{u}_2^{[k]})_{L^2(\partial\Omega_2\cap\partial\Omega)} + \frac{\beta_N}{2}\|\hat{u}_2^{[k]}\|^2_{L^2(\partial\Omega_1\cap\partial\Omega)} \\ &= \frac{1}{2}b_2(\hat{u}_2^{[k]} - u_2^{[k]},\hat{u}_2^{[k]} - u_2^{[k]}) + \frac{\beta_N}{2}\|\hat{u}_2^{[k]} + c_2\beta_N^{-1}\nabla u_2^{[k]}\cdot\boldsymbol{n}_2\|^2_{L^2(\partial\Omega_2\cap\partial\Omega)} \\ &\quad - \frac{1}{2}b_2(u_2^{[k]},u_2^{[k]}) - \frac{\beta_N}{2}\|c_2\beta_N^{-1}\nabla u_2^{[k]}\cdot\boldsymbol{n}_2\|^2_{L^2(\partial\Omega_2\cap\partial\Omega)}.\end{aligned}$$

As a result, we conclude that the weak solution $\hat{u}_2^{[k]} \in H^1(\Omega_2)$ of (3.13) is also the minimizer of functional

$$\mathcal{G}_2(\hat{u}_2^{[k]}) = \frac{1}{2}b_2(\hat{u}_2^{[k]} - u_2^{[k]},\hat{u}_2^{[k]} - u_2^{[k]}) + \frac{\beta_N}{2}\|\hat{u}_2^{[k]} + c_2\beta_N^{-1}\nabla u_2^{[k]}\cdot\boldsymbol{n}_2\|^2_{L^2(\partial\Omega_2\cap\partial\Omega)}.$$

Clearly, be defining $\check{c}_2 = \min\{c_2,1\}$, it is obvious that

$$\begin{aligned}\mathcal{G}_2(\hat{u}_2^{[k]}) &\geq \frac{1}{2}b_2(\hat{u}_2^{[k]} - u_2^{[k]},\hat{u}_2^{[k]} - u_2^{[k]}) = \int_{\Omega_2}\left(\frac{c_2}{2}|\nabla(\hat{u}_2^{[k]} - u_2^{[k]})|^2 + \frac{1}{2}|\hat{u}_2^{[k]} - u_2^{[k]}|^2\right)dx \\ &\geq \frac{\check{c}_2}{2}\|\hat{u}_2^{[k]} - u_2^{[k]}\|^2_{H^1(\Omega_2)}. \end{aligned} \tag{3.16}$$

On the other hand, note that $u_2^{[k]} \in H^2(\Omega_2)$ under same assumptions [11, 13], we have by the trace theorem [41] that $(\nabla u_2^{[k]}\cdot\boldsymbol{n}_2)|_{\partial\Omega_2} \in H^{\frac{1}{2}}(\partial\Omega_2)$, and therefore there exists a function $\zeta \in H^1(\Omega_2)$ such that $\zeta|_{\partial\Omega_2} = -(\nabla u_2^{[k]}\cdot\boldsymbol{n}_2)|_{\partial\Omega_2}$. Moreover, the trace operator has a continuous linear right inverse and hence there holds

$$\|\zeta\|_{H^1(\Omega_2)} \leq C\|\nabla u_2^{[k]}\cdot\boldsymbol{n}_2\|_{H^{\frac{1}{2}}(\partial\Omega_2)} \leq C\|u_2^{[k]}\|_{H^2(\Omega_2)},$$

where $C>0$ is a generic constant that only depends on the subdomain Ω_2 [2].

Then by setting $\bar{u}_2 = c_2\beta_N^{-1}\zeta + u_2^{[k]}$ and using the boundary condition $u_2^{[k]} = 0$ on $\partial\Omega_2\cap\partial\Omega$ in (3.15), the optimality of $\hat{u}_2^{[k]}$ among all functions in $H^1(\Omega_2)$ implies that

$$\begin{aligned}\mathcal{G}_2(\hat{u}_2^{[k]}) &\leq \mathcal{G}_2(\bar{u}_2) = c_2^2\beta_N^{-2}\int_{\Omega_1}\left(\frac{c_1}{2}|\nabla\zeta|^2 + \frac{1}{2}|\zeta|^2\right)dx + \frac{\beta_N}{2}\|u_2^{[k]}\|^2_{L^2(\partial\Omega_1\cap\partial\Omega)} \\ &\leq \frac{\hat{c}_2}{2}c_2^2\beta_N^{-2}\|\zeta\|^2_{H^1(\Omega_1)},\end{aligned}$$

where $\hat{c}_2 = \max\{c_2,1\}$. Consequently, we have by (3.16) that

$$\|\hat{u}_2^{[k]} - u_2^{[k]}\|_{H^1(\Omega_1)} \leq \frac{c_2}{\beta_N}\sqrt{\frac{\hat{c}_2}{\check{c}_2}}\|\zeta\|_{H^1(\Omega_1)},$$

which completes the proof. □

In other words, by sending penalty coefficients $\beta_D \to \infty$ in (3.1) and $\beta_N \to \infty$ in (3.3), minimizers of our relaxed optimization problems (3.1, 3.3) could converge to that of the classical Dirichlet-Neumann algorithm (2.3, 2.4), which provides the theoretical groundwork for establishing the Dirichlet-Neumann learning algorithm. Moreover, our quantitative error analysis also indicates that the choice of penalty coefficient is closely tied to values of high-contrast coefficients, e.g., $\beta_D = 400c_1$ and $\beta_N = 400c_2$, differing from other work that set $\beta_D = \beta_N$ [39] during training.

3.3 Dirichlet-Neumann learning algorithm

Next, unknown solutions in (3.1) and (3.3) are parametrized using neural networks [14]

$$\hat{u}_1^{[k]}(x) = \hat{u}_1(x;\theta_1^{[k]}) \quad \text{and} \quad \hat{u}_2^{[k]}(x) = \hat{u}_2(x;\theta_2^{[k]}),$$

where $\theta_i^{[k]}$ represents the collection of trainable parameters at the k-th outer iteration for $i = 1, 2$. For example, fully-connected neural networks [51], ResNets [18], or other kinds of architectures can be adopted to construct the solution ansatz (more details about our model setup are included in **Appendix C**). Extensive studies have thus been conducted on the approximation error [7, 20, 47], which is not discussed here. Moreover, thanks to the mesh-free feature of neural networks, the extension operator in (3.2) can be realized in a very straightforward way, that is,

$$R_1\gamma_0\hat{u}_2(x,\theta_2) = \hat{u}_2(x,\theta_2), \tag{3.17}$$

and is required to be differentiable within Ω_1 and to satisfy the zero boundary values on $\partial\Omega_1 \cap \partial\Omega$ through an additional penalty term in (3.3). One can also employ a piecewise neural network [17] to realize the extension operation (3.17).

Next, to discretize the functionals (3.1) and (3.3), the routine way of generating training sample points inside each subdomain and at its boundary is to use the Monte Carlo method or its variants [46], namely,

$$X_{\Omega_i} = \{x_n^{\Omega_i}\}_{n=1}^{N_{\Omega_i}}, \quad X_{D_i} = \{x_n^{D_i}\}_{n=1}^{N_{D_i}}, \quad \text{and} \quad X_\Gamma = \{x_n^\Gamma\}_{n=1}^{N_\Gamma},$$

where $D_i := \partial\Omega_i \cap \partial\Omega$, N_{Ω_i}, N_{D_i}, and N_Γ denote the sample size of training datasets X_{Ω_i}, X_{D_i}, and X_Γ, respectively. As a result, by defining the empirical loss functions

$$L_{\Omega_i}(\hat{u}_i) = \frac{1}{N_{\Omega_i}} \sum_{n=1}^{N_{\Omega_i}} \left(\frac{c_i}{2} |\nabla \hat{u}_i(x_n^{\Omega_i};\theta_i)|^2 - f(x_n^{\Omega_i})\hat{u}_i(x_n^{\Omega_i};\theta_i) \right),$$

$$L_{D_i}(\hat{u}_j) = \frac{1}{N_{D_i}} \sum_{n=1}^{N_{D_i}} |\hat{u}_j(x_n^{D_i};\theta_j)|^2,$$

$$L_{\Gamma_D}(\hat{u}_1,u_\Gamma^{[k]})=\frac{1}{N_\Gamma}\sum_{n=1}^{N_\Gamma}|\hat{u}_1(x_n^\Gamma;\theta_1)-u_\Gamma^{[k]}(x_n^\Gamma)|^2,\quad L_{\Gamma_N}(\hat{u}_2)=\frac{1}{N_\Gamma}\sum_{n=1}^{N_\Gamma}q(x_n^\Gamma)\hat{u}_2(x_n^\Gamma;\theta_2),$$

$$L_N(\hat{u}_2,\hat{u}_1)=\frac{1}{N_{\Omega_1}}\sum_{n=1}^{N_{\Omega_1}}\Big(c_1\nabla\hat{u}_1(x_n^{\Omega_1};\theta_1^{[k]})\cdot\nabla\hat{u}_2(x_n^{\Omega_1};\theta_2)-f(x_n^{\Omega_1})\hat{u}_2(x_n^{\Omega_1};\theta_2)\Big),$$

for $1\le i,j\le 2$, the learning task associated with our Dirichlet subproblem (3.1) reads

$$\theta_1^{[k]}=\underset{\theta_1}{\arg\min}L_{\Omega_1}(\hat{u}_1)+\frac{\beta_D}{2}\Big(L_{D_1}(\hat{u}_1)+L_{\Gamma_D}(\hat{u}_1,u_\Gamma^{[k]})\Big). \tag{3.18}$$

We also note that as an alternative to the deep Ritz method (3.18) [70], the Dirichlet subproblem (3.1) or (2.6) can also be solved using PINNs [53], which is known to empirically work better for local problems with sufficient smooth solutions. To be precise, by incorporating the residual into the loss function, the learning task of Dirichlet subproblem in (2.6) can also be formulated as

$$\theta_1^{[k]}=\underset{\theta_1}{\arg\min}L_{\Omega_1}^{\mathrm{PINN}}(\hat{u}_1)+\frac{\beta_D}{2}\left(L_{D_1}(\hat{u}_1)+L_{\Gamma_D}(\hat{u}_1)\right), \tag{3.19}$$

where

$$L_{\Omega_i}^{\mathrm{PINN}}(\hat{u}_i)=\frac{1}{N_{\Omega_i}}\sum_{n=1}^{N_{\Omega_i}}\Big|-\nabla\cdot\big(c_i\nabla\hat{u}_i(x_n^{\Omega_i};\theta_i)\big)+\hat{u}_i(x_n^{\Omega_i};\theta_i)-f(x_n^{\Omega_i})\Big|^2\quad\text{for } i=1,2.$$

On the other hand, the learning task of Neumann subproblem (3.3) takes on the form

$$\theta_2^{[k]}=\underset{\theta_2}{\arg\min}L_{\Omega_2}(\hat{u}_2)+L_N(\hat{u}_2,\hat{u}_1^{[k]})+L_{\Gamma_N}(\hat{u}_2)+\frac{\beta_N}{2}\left(L_{D_1}(\hat{u}_2)+L_{D_2}(\hat{u}_2)\right). \tag{3.20}$$

It is noteworthy that though the loss value of (3.18) or (3.19) continues to decrease as the training proceeds, the gradient of trained model is often observed to possess higher precision inside the domain rather than at its interface [1,8,58,62], which may degenerate the performance when explicitly exchanging the flux data through (2.6). Fortunately, by employing our learning algorithm (3.20), the flux transmission condition is enforced in a variational way, employing $\nabla\hat{u}_1^{[k]}|_{\Omega_1}$ rather than $\nabla\hat{u}_1^{[k]}|_\Gamma$ that allows for better precision.

In addition, domain decomposition leads to simpler functions to be learned on each subdomain, which allows us to incorporate the residual loss into (3.20), that is,

$$\begin{aligned}\theta_2^{[k]}=\underset{\theta_2}{\arg\min}\ & L_{\Omega_2}(\hat{u}_2)+L_N(\hat{u}_2,\hat{u}_1^{[k]})+L_{\Gamma_N}(\hat{u}_2)+\frac{\beta_N}{2}\left(L_{D_1}(\hat{u}_2)+L_{D_2}(\hat{u}_2)\right)\\&+\lambda_N L_{\Omega_2}^{\mathrm{PINN}}(\hat{u}_2),\end{aligned}$$

Algorithm 1 Dirichlet-Neumann Learning Algorithm

% *Initialization*
– specify the network architecture $\hat{u}_i(x;\theta_i)$ $(i=1,2)$ for each subproblem;
– generate the Monte Carlo sampling points X_Γ, X_{Ω_i}, and X_{D_i} for $i=1,2$;
% *Outer Iteration Loop*
Start with the initial guess $u_\Gamma^{[0]}$ of unknown solution values at the interface Γ;
for $k \leftarrow 0$ to K (maximum number of outer iterations) **do**
  while stopping criteria are not satisfied **do**
    % *Dirichlet Subproblem-Solving via the Deep Ritz or PINNs Approach*

$$\theta_1^{[k]} = \arg\min_{\theta_1} L_{\Omega_1}(\hat{u}_1) + \frac{\beta_D}{2}\left(L_{D_1}(\hat{u}_1) + L_\Gamma(\hat{u}_1, u_\Gamma^{[k]})\right),$$

    % *Neumann Subproblem-Solving via our Compensated Deep Ritz Method*

$$\theta_2^{[k]} = \arg\min_{\theta_2} L_{\Omega_2}(\hat{u}_2) + L_N(\hat{u}_2, \hat{u}_1^{[k]}) + L_{\Gamma_N}(\hat{u}_2) + \frac{\beta_N}{2}\left(L_{D_1}(\hat{u}_2) + L_{D_2}(\hat{u}_2)\right),$$

    % *Update of Unknown Solution Values at Interface*

$$u_\Gamma^{[k+1]}(x_n^\Gamma) = \rho\hat{u}_2(x_n^\Gamma;\theta_2^{[k]}) + (1-\rho)u_\Gamma^{[k]}(x_n^\Gamma),\ \ i=1,\cdots,N_\Gamma,$$

  end while
end for

where $\lambda_N \in \mathbb{R}_+$ is set by the user. The last loss term can be regarded as a regularization term that helps improve the approximation accuracy and prevent the training process (3.20) from getting trapped in trivial solutions especially when $c_2 \gg 1$.

More specifically, the detailed iterative scheme is summarized in Algorithm 1 and Fig. 2, where the stopping criteria can be constructed by measuring the difference in numerical solutions between two consecutive iterations [17,39]. Moreover, our learning algorithm can be naturally modified to adapt to a parallel computing environment [68]. We also note that, in addition to the previously mentioned approximation error, the generalization error of trained network models has been extensively studied [7,42,66]. However, the quantification of optimization error [16,28,33] remains an open issue due to the highly non-convex nature of the loss landscape and the substantial number of trainable parameters. Therefore, a comprehensive error analysis of our proposed learning algorithm is deferred to future research.

4 Numerical experiments

In this section, numerical experiments on a series of interface problems (2.1) are carried out to showcase the effectiveness of Dirichlet-Neumann learning algorithm (referred to as DNLA hereafter, with the type of deep learning solver used for solving the Dirichlet sub-

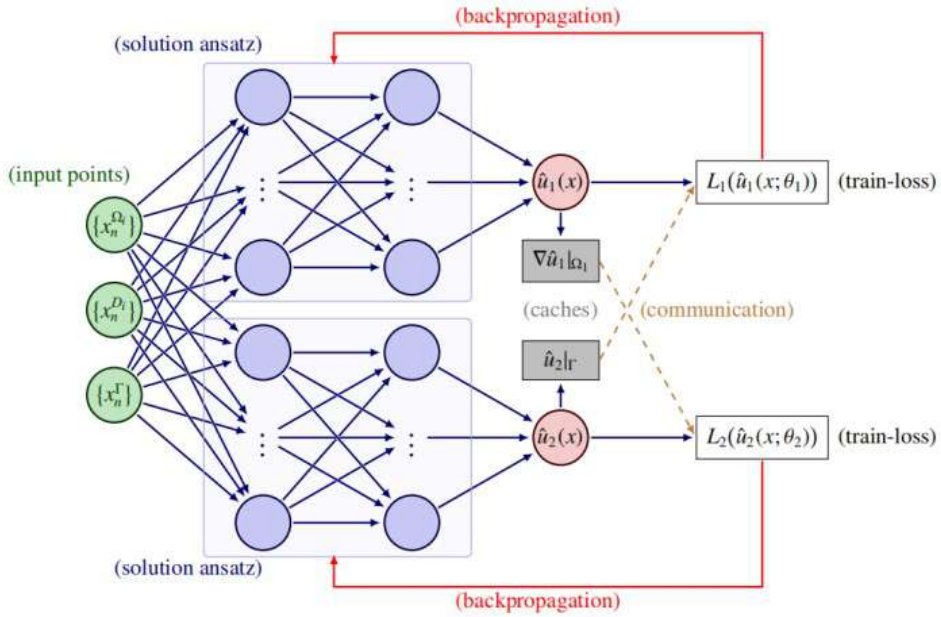


Figure 2: Computational graph of our Dirichlet-Neumann learning algorithm for (2.6).

problem specified in the bracket). Throughout all experiments, we employ ResNets [18] for addressing our local problems, deferring the exploration of more intricate architectures [21,23] to future investigations. Additionally, to meet the smoothness requirement of the extension operator (3.17) (see also (3.2) or (3.15)), we select the hyperbolic tangent activation function rather than the ReLU activation function [14].

As a comparison to our algorithm, we also conduct numerical experiments using the most straightforward approach, that is, employing PINNs [53] for solving all the decomposed subproblems (2.6) as proposed in [39], which is referred to as "DeepDDM". For a fair comparison, the same relaxation parameter ρ is set for both equations (2.6) and (2.5), and the mean value and standard deviation of the discrete relative L^2 error

$$\epsilon=\frac{\sqrt{\sum_{m=1}^{M}\left|u(x_m)-\hat{u}^{[k]}(x_m)\right|^2}}{\sqrt{\sum_{m=1}^{M}|u(x_m)|^2}},\quad \text{where}\quad \hat{u}^{[k]}(x)=\begin{cases}\hat{u}_1(x;\theta_1^{[k]}) & \text{if } x\in\overline{\Omega}_1,\\ \hat{u}_2(x;\theta_2^{[k]}) & \text{if } x\in\Omega_2,\end{cases}$$

along outer iterations are reported over 5 independent simulations. Here, $\{x_m\}_{m=1}^M$ represents testing points that are uniformly distributed over the domain Ω. When the maximum number of outer iterations is reached, i.e., $k=K$ in Algorithm 1, the trained model is denoted by $\hat{u}(x)$ instead of $\hat{u}^{[K]}(x;\theta)$ or $\hat{u}(x;\theta^{[K]})$ for notational simplicity.

Additional details regarding our experimental setup, such as the depth and width of network, training and testing datasets, and penalty coefficients, can be found **Appendix**

C. The stopping criterion of our method requires that either the relative L_2 error of two consecutive iterations is less than 0.001 or the number of outer iterations reaches 30. All experiments are conducted using PyTorch [49,50] on Nvidia GeForce RTX 3090 cards[§].

4.1 Flower-shape interface problem in two dimension

First, we consider an interface problem (2.1) whose solution is continuous [40], i.e.,

$$\begin{aligned} -\nabla\cdot(c_i\nabla u_i(x,y))+u_i(x,y)&=f_i(x,y) \quad \text{in } \Omega_i,\\ u_2(x,y)&=g_2(x,y) \quad \text{on } \partial\Omega_2\cap\partial\Omega,\\ u_1(x,y)=u_2(x,y) \text{ and } -c_1\nabla u_1(x,y)\cdot\boldsymbol{n}_1-c_2\nabla u_2(x,y)\cdot\boldsymbol{n}_2&=q(x,y) \quad \text{on } \Gamma=\partial\Omega_1, \end{aligned} \tag{4.1}$$

for $i=1,2$, where

$$\Omega=(-1,1)^2 \text{ and } \Omega_1=\left\{(x,y)\,\middle|\,\sqrt{x^2+y^2}\le\frac{1}{2}+\frac{1}{4}\sin\left(12\arctan\frac{y}{x}\right)\right\}.$$

The source term $f_i(x,y)$, boundary data $g_2(x,y)$ and jump condition $q(x,y)$ in equations (4.1) are derived from the exact solution

$$u(x,y)=\begin{cases} c_1^{-1}(x^2+y^2)^{\frac{3}{2}}\sin\left(2\pi\sqrt{x^2+y^2}-\pi-\frac{\pi}{2}\sin(12\arctan\frac{y}{x})\right) & \text{for } (x,y)\in\Omega_1,\\ c_2^{-1}(x^2+y^2)^{\frac{3}{2}}\sin\left(2\pi\sqrt{x^2+y^2}-\pi-\frac{\pi}{2}\sin(12\arctan\frac{y}{x})\right) & \text{for } (x,y)\in\Omega_2. \end{cases}$$

More specifically, the exact solution is shown in Fig. 3, where the high-contrast coefficients are given by $(c_1,c_2)=(1,1)$ and $(1,10^3)$ respectively. Clearly, both cases should be resolved with a reasonable good accuracy to meet the robustness requirement regarding varying coefficients. The initial guess of the unknown value at interface is set to be

$$u_\Gamma^{[0]}(x,y)=u(x,y)-1000x(x-1)y(y-1).$$

Table 2 shows the error profile $|\hat{u}(x,y)-u(x,y)|$ of different methods in a typical simulation[¶], while the discrete relative L^2 error is reported in Table 3. Clearly, problem (4.1) with a high-contrast coefficient $(c_1,c_2)=(1,10^3)$ could be effectively solved via the DeepDDM scheme [39]. However, its effectiveness degrades when coefficients are set to be $(c_1,c_2)=(1,1)$. On the other hand, our methods consistently demonstrate promising performance in all scenarios.

In addition, we report the error profile $\|\nabla\hat{u}-\nabla u\|_{\ell^2}$ of trained network solutions in Table 4. Obviously, the precision of Dirichlet-to-Neumann map for Dirichlet subproblem is not as satisfactory as that of the gradient in the interior domain, therefore our learning methods have demonstrated superior performance in both cases of Table 3. Moreover, it can be inferred from (3.20) that the Neumann subproblem solver could benefit from a good approximation of $\nabla u_1^{[k]}$ within the subdomain Ω_1, thus DNLA (PINNs) empirically performs better than DNLA (deep Ritz) as shown in Table 2.

[§]Source code is available on GitHub at https://github.com/AI4SC-TJU/DNLA-IntfcProb.

[¶]The corresponding iterative solutions are depicted in **Appendix C** owing to space constraints.

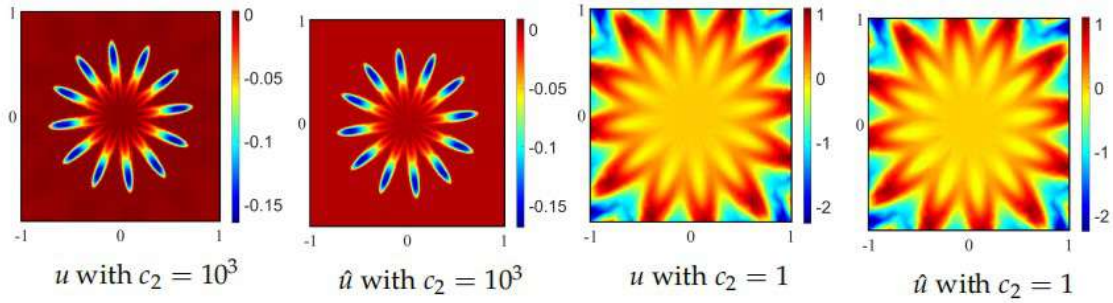
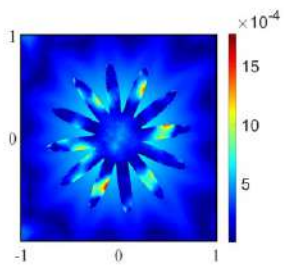
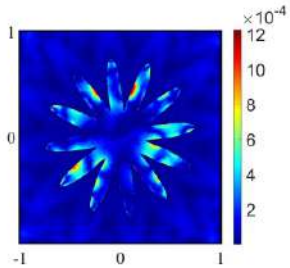
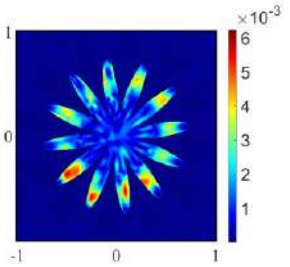
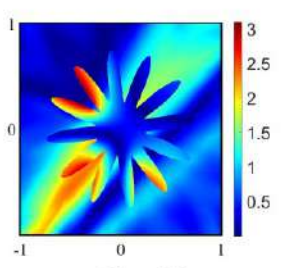
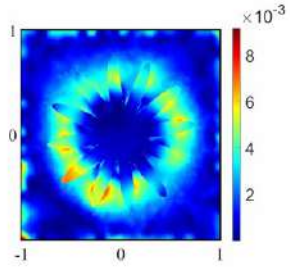
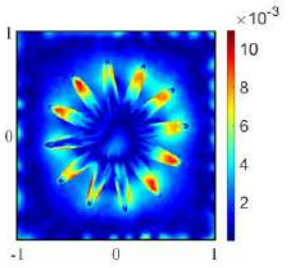


Figure 3: Exact and network solutions for testing example (4.1) with $(c_1,c_2) = (1,1)$ and $(1,10^3)$, where the DNLA (PINNs) is employed.

Table 2: Error profile $|\hat{u}-u|$ of different methods for testing example (4.1).

	DeepDDM	DNLA (PINNs)	DNLA (Deep Ritz)
$c_1 = 1$ $c_2 = 10^3$ $\rho = 1$	 $K = 3$	 $K = 2$	 $K = 3$
$c_1 = 1$ $c_2 = 1$ $\rho = 0.5$	 $K = 15$	 $K = 15$	 $K = 15$

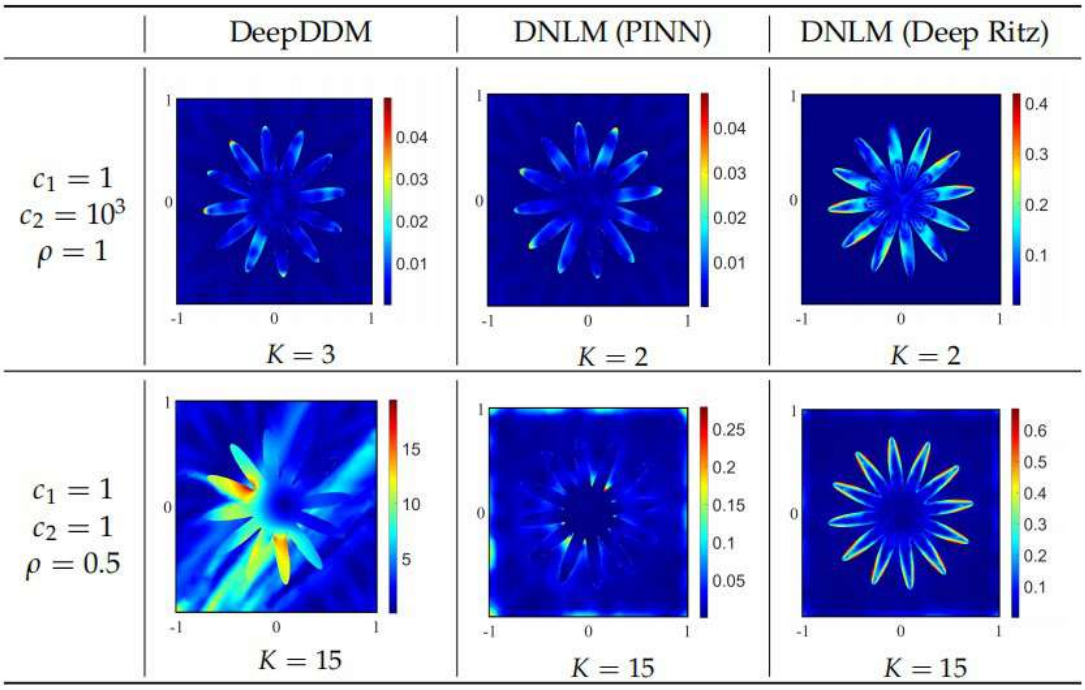
4.2 Citrus interface problem in three dimension

In addition to the mesh-free feature, which is attractive for addressing intricate interface geometries, another key advantage of using artificial neural networks is their ability to tackle challenges stemming from the curse of dimensionality. To this end, we consider a

Table 3: Relative L^2 errors (mean $\pm$ standard deviation over 5 runs) for example (4.1).

Coefficients \ Outer Iterations		1	2	10	15
$(1,10^3)$	DeepDDM	296.91 $\pm$ 0.17	0.58$\pm$ 0.33	0.04 $\pm$ 0.02	-
	DNLA (PINNs)	296.17 $\pm$ 0.14	0.02$\pm$ 0.01	0.01 $\pm$ 0.00	-
	DNLA (Deep Ritz)	295.49 $\pm$ 0.45	0.07 $\pm$ 0.03	0.05 $\pm$ 0.01	-
$(1,1)$	DeepDDM	52.21 $\pm$ 2.22	37.85 $\pm$ 3.39	8.64 $\pm$ 2.63	3.12 $\pm$ 1.46
	DNLA (PINNs)	31.01 $\pm$ 0.02	15.52 $\pm$ 0.00	0.07 $\pm$ 0.00	0.02 $\pm$ 0.00
	DNLA (Deep Ritz)	30.96 $\pm$ 0.05	15.48$\pm$ 0.03	0.07 $\pm$ 0.00	0.03 $\pm$ 0.01

Table 4: Error profiles $\|\nabla\hat{u}-\nabla u\|_{\ell^2}$ of different methods for testing example (4.1).



three-dimensional problem with homogeneous jump conditions, i.e.,

$$\begin{aligned} -\nabla\cdot(c_i\nabla u_i(x,y)) &= f_i(x,y) && \text{in } \Omega_i, \\ u_2(x,y) &= g_2(x,y) && \text{on } \partial\Omega_2\cap\partial\Omega, \\ u_1(x,y)=u_2(x,y) \text{ and } -c_1\nabla u_1(x,y)\cdot\boldsymbol{n}_1 - c_2\nabla u_2(x,y)\cdot\boldsymbol{n}_2 &= 0 && \text{on } \Gamma=\partial\Omega_1, \end{aligned} \tag{4.2}$$

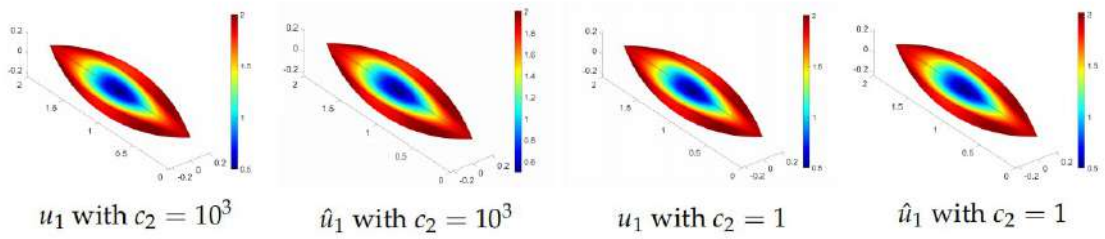


Figure 4: Exact and network solutions for the Dirichlet subproblem of testing example (4.2) with $(c_1,c_2) = (1,10^3)$ and $(1,1)$, where the DNLA (PINNs) is employed.

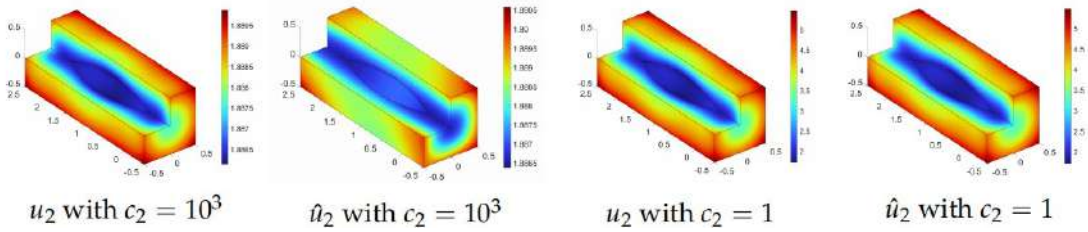


Figure 5: Exact and network solutions for the Neumann subproblem of testing example (4.2) with $(c_1,c_2) = (1,1)$ and $(1,10^3)$, where the DNLA (PINNs) is employed.

for $i = 1, 2$, where

$$\Omega = (-0.5,0.5)\times(-0.5,2.5)\times(-0.5,0.5),$$
$$\Omega_1 = \{(x,y,z)|16(x^2+z^2)+y^3(y-2)^3<0\}.$$

The source term $f_i(x,y)$ and boundary data $g_2(x,y)$ are derived from the exact solution

$$u(x,y,z) = \begin{cases} 2c_1^{-1}\ln\gamma(x,y,z)+0.5 & \text{for } (x,y,z)\in\Omega_1, \\ 2c_2^{-1}\ln\gamma(x,y,z)+(c_1^{-1}-c_2^{-1})\ln 4+0.5 & \text{for } (x,y,z)\in\Omega_2, \end{cases} \tag{4.3}$$

where $\gamma(x,y,z) = y(y-2)^3+16(x^2+z^2)+2$. More specifically, the exact solution is shown in Fig. 4 and Fig. 5, where the high-contrast coefficients are given by $(c_1,c_2) = (1,1)$ and $(1,10^3)$ respectively. The initial guess of the unknown value at interface is set to be

$$u_\Gamma^{[0]}(x,y,z) = u(x,y,z)-1000\cos(2\pi x)\sin(2\pi(y-1))\sin(2\pi z).$$

Considering the increased complexity of problem (4.2), the maximum number of epochs for each subproblem is $3k$, with an initial learning rate set to 10^{-4}.

Error profiles $|\hat{u}_i-u_i|$, $\|\nabla\hat{u}_i-\nabla u_i\|_{\ell^2}$, and relative L^2 errors for different learning methods in a typical simulation are displayed in Table 6, Table 7, and Table 5, respectively. The solution of Dirichlet subproblem is depicted on a citrus subdomain with a

Table 5: Relative L^2 errors (mean $\pm$ standard deviation over 5 runs) for example (4.2).

Coefficients \ Outer Iterations		1	2	10	15
$(1,10^3)$	DeepDDM	76.69 ± 0.27	0.08 ± 0.00	0.01 ± 0.00	-
	DNLA (PINNs)	62.46 ± 1.43	0.00 ± 0.00	0.00 ± 0.00	-
	DNLA (deep Ritz)	63.58 ± 1.04	0.01 ± 0.00	0.00 ± 0.00	-
$(1,1)$	DeepDDM	54.41 ± 0.66	21.37 ± 2.77	0.02 ± 0.00	0.01 ± 0.00
	DNLA (PINNs)	36.26 ± 3.16	23.36 ± 1.63	0.11 ± 0.00	0.01 ± 0.00
	DNLA (Deep Ritz)	50.41 ± 0.23	25.06 ± 0.06	0.10 ± 0.00	0.02 ± 0.00

quarter of its volume removed and the same is applied to the Neumann subproblem. Notably, by fine-tuning the hyperparameter at each outer iteration, it can be observed from Table 6 that the trained network solution of DeepDDM approach aligns closely with the true solution for $(c_1,c_2)=(1,1)$. However, its corresponding Dirichlet-to-Neumann map still exhibits significant errors as shown in Table 7.

On the contrary, our learning algorithms rely on a variational approach rather than directly exchanging the Dirichlet-to-Neumann map. Moreover, as indicated in Table 7, the accuracy of gradient for Dirichlet subproblem is higher within the interior domain than at the boundary, therefore achieving superior performance as shown in Table 5.

4.3 Octagonal interface problem in two dimension

Finally, we consider the following elliptic interface problem

$$\begin{aligned} -\nabla\cdot(c_i\nabla u_i(x,y)) &= f_i(x,y) && \text{in } \Omega_i,\\ u_2(x,y) &= g_2(x,y) && \text{on } \partial\Omega_2\cap\partial\Omega,\\ u_1(x,y)=u_2(x,y) \text{ and } -c_1\nabla u_1(x,y)\cdot\boldsymbol{n}_1-c_2\nabla u_2(x,y)\cdot\boldsymbol{n}_2 &= q(x,y) && \text{on } \Gamma=\partial\Omega_1, \end{aligned} \tag{4.4}$$

for $i=1,2$, where

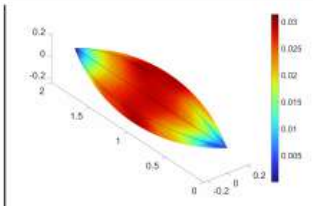
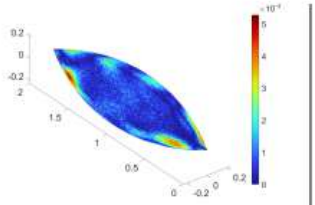
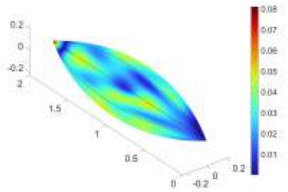
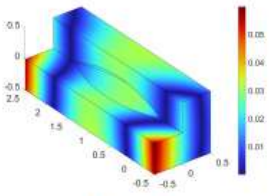
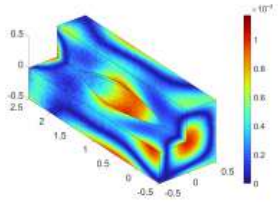
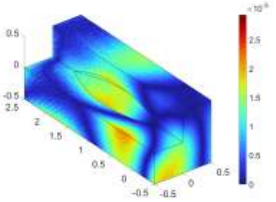
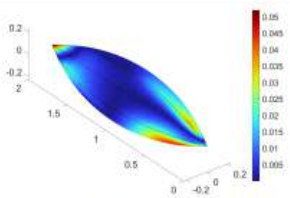
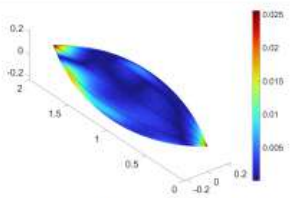
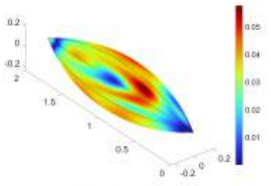
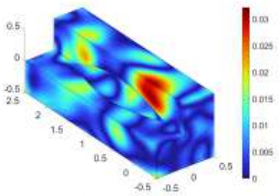
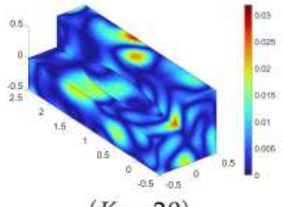
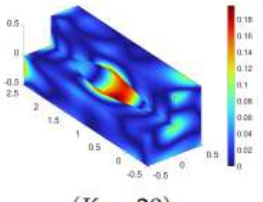
$$\Omega=(-1,1)\times(-1,1) \text{ and } \Omega_1=\left\{(x,y)\,\middle|\,\sqrt{x^2+y^2}\le\frac{3(676-26\cos(25\arctan\frac{y}{x}))}{6250}\right\}.$$

Differing from example (4.1), the exact solution of (4.4) is set to be

$$u(x,y)=0.5+(x^2+y^2)^{\frac{3}{2}}\sin\left(8\pi\sqrt{x^2+y^2}-\frac{3\pi(81-9\cos(8\arctan\frac{y}{x}))}{80}\right),$$

thereby the force term of (4.4) may manifest multiscale phenomena due to the impact of high-contrast coefficients. When solving the Neumann subproblem (2.6) via the usage of

Table 6: Error profile $|\hat{u}_i - u_i|$ of different methods for testing example (4.2).

	DeepDDM	DNLA (PINNs)	DNLA (Deep Ritz)
$c_1 = 1$ $c_2 = 10^3$ $\rho = 1.0$	 $(K = 4)$	 $(K = 2)$	 $(K = 2)$
	 $(K = 4)$	 $(K = 2)$	 $(K = 2)$
$c_1 = 1$ $c_2 = 1$ $\rho = 0.5$	 $(K = 20)$	 $(K = 20)$	 $(K = 20)$
	 $(K = 20)$	 $(K = 20)$	 $(K = 20)$

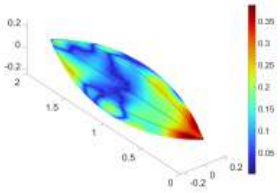
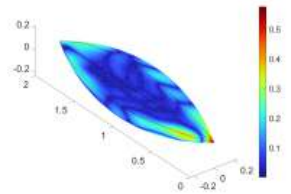
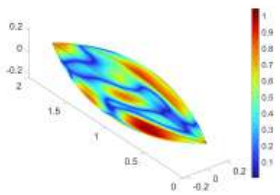
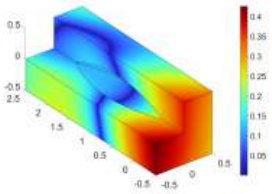
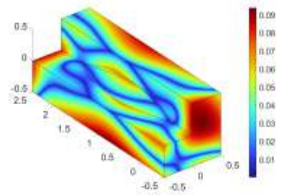
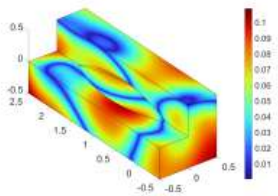
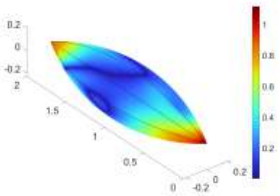
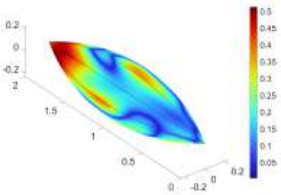
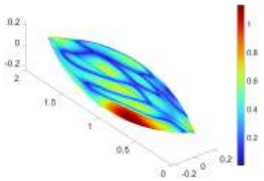
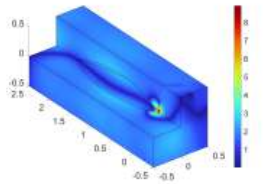
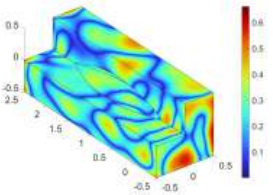
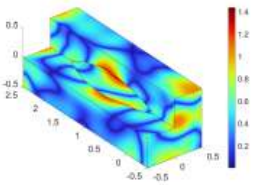
PINNs [53], the inclusion of a relatively large force function within the residual loss can pose additional difficulties for network training.

Similar to previous examples, we set $(c_1,c_2) = (1,10^3)$ and $(1,1)$ to assess the robustness with respect to varying coefficients, and the exact solution is displayed in Fig. 6. Moreover, the initial guess of the unknown solution's value at interface is set to be

$$u_\Gamma^{[0]}(x,y) = u(x,y) - 100\cos(100\pi x)\cos(100\pi y),$$

while error profiles $|\hat{u}_i - u_i|$, $\|\hat{u}_i - \nabla u_i\|_{\ell^2}$, and relative L^2 errors for different methods in a typical simulation are presented in Table 9, Table 10, and Table 8, respectively.

Table 7: Error profiles $\|\nabla\hat{u}_i-\nabla u_i\|_{\ell^2}$ of different methods for testing example (4.2).

	DeepDDM	DNLA (PINNs)	DNLA (Deep Ritz)
$c_1=1$ $c_2=10^3$ $\rho=1.0$	 $(K=4)$	 $(K=2)$	 $(K=2)$
	 $(K=4)$	 $(K=2)$	 $(K=2)$
$c_1=1$ $c_2=1$ $\rho=0.5$	 $(K=20)$	 $(K=20)$	 $(K=20)$
	 $(K=25)$	 $(K=20)$	 $(K=20)$

As can be seen from Table 9 and Table 8, the performance of DeepDDM approach is unsatisfactory even for high-contrast coefficients $(c_1,c_2)=(1,10^3)$, primarily attributable to the large force function within the residual loss term. In contrast, our learning algorithms circumvent this issue by utilizing the variational form (3.20). Additionally, our methods demonstrate superior performance even in the case of inaccurate Dirichlet-to-Neumann map shown in Table 10.

Notably, our method utilizes variational loss functionals to enforce flux transmission and addresses the decomposed subproblem through a standard network training process, thereby possessing the potential to handle higher-dimensional problems.

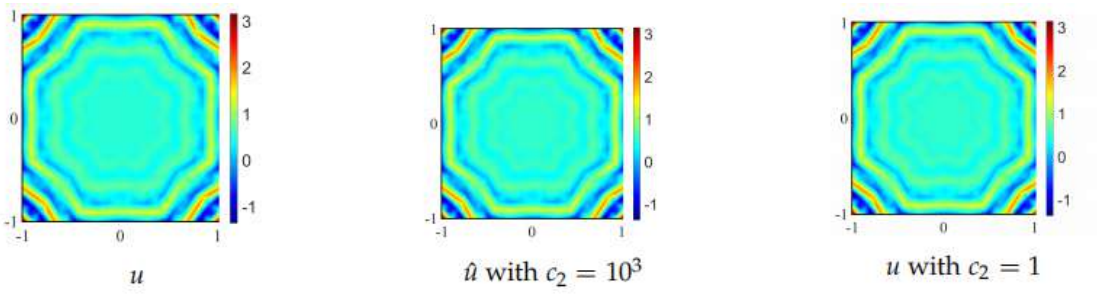


Figure 6: Exact and network solutions for testing example (4.4) with $(c_1,c_2)=(1,1)$ and $(1,10^3)$, where the DNLA (PINNs) is employed.

Table 8: Relative L^2 errors (mean $\pm$ standard deviation over 5 runs) for example (4.4).

Coefficients \ Outer Iterations		1	2	4	10
$(1,10^3)$	DeepDDM	11.06$\pm$0.85	1.45$\pm$0.51	1.51$\pm$0.38	1.52$\pm$0.77
	DNLA (PINN)	10.53$\pm$1.12	0.04$\pm$0.01	0.03$\pm$0.01	-
	DNLA (deep Ritz)	11.62$\pm$0.73	0.04$\pm$0.01	0.04$\pm$0.01	-
$(1,1)$	DeepDDM	18.95$\pm$2.96	10.88$\pm$1.79	7.52$\pm$0.79	3.35$\pm$0.25
	DNLA (PINN)	11.59$\pm$1.16	5.35$\pm$1.36	1.41$\pm$0.27	0.03$\pm$0.01
	DNLA (Deep Ritz)	12.93$\pm$2.54	6.43$\pm$1.31	1.67$\pm$0.34	0.03$\pm$0.01

5 Conclusion

Motivated by the continuous formulation of classical Dirichlet-Neumann algorithm, the adoption of deep learning techniques as subproblem solvers has recently attracted considerable research interest owing to their meshless feature and effectiveness in addressing high-dimensional problems. However, when employing artificial neural networks to solve the Dirichlet subproblem, it is often empirically observed that the gradient of trained network solution often exhibits higher errors at the boundary compared to its interior domain. Such a pattern of error distribution may result in a degradation of overall performance when the flux transmission condition is explicitly enforced between neighbouring subdomains. Thanks to the variational principle, the exchange of flux data can be facilitated using the interior gradient instead of the Neumann trace, thereby motivating the development of our Dirichlet-Neumann learning algorithm. Moreover, a rigorous error analysis is established to obtain the error bound of boundary penalty treatment for both the Dirichlet and Neumann subproblems, which also sheds light on the setup of penalty coefficients. Finally, a wide variety of numerical examples are carried out to validate the effectiveness and robustness of our methods, achieving promising performance even in the presence of inaccurate Dirichlet-to-Neumann map.

Table 9: Error profile $|\hat{u}_i - u_i|$ of different methods for testing example (4.4).

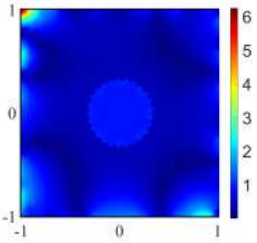
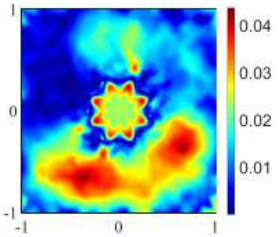
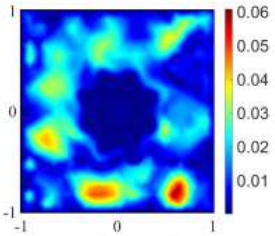
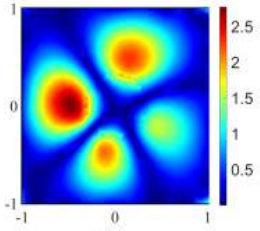
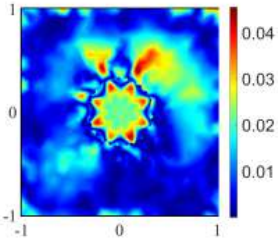
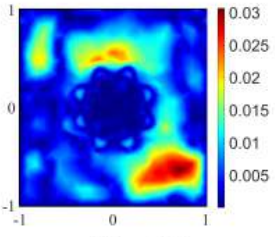
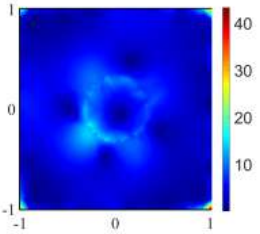
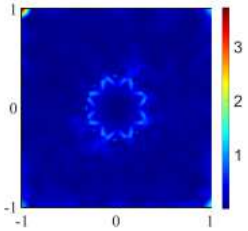
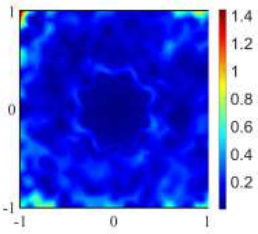
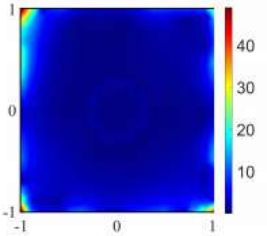
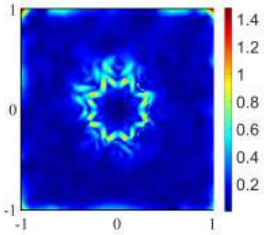
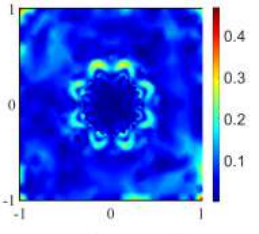
	DeepDDM	DNLA (PINNs)	DNLA (Deep Ritz)
$c_1 = 1$ $c_2 = 10^3$ $\rho = 1.0$	 $(K = 4)$	 $(K = 4)$	 $(K = 4)$
$c_1 = 1$ $c_2 = 1$ $\rho = 0.5$	 $(K = 15)$	 $(K = 10)$	 $(K = 10)$

Table 10: Error profiles $\|\nabla\hat{u}_i - \nabla u_i\|_{\ell^2}$ of different methods for testing example (4.4).

	DeepDDM	DNLA (PINNs)	DNLA (Deep Ritz)
$c_1 = 1$ $c_2 = 10^3$ $\rho = 1.0$	 $(K = 2)$	 $(K = 4)$	 $(K = 4)$
$c_1 = 1$ $c_2 = 1$ $\rho = 0.5$	 $(K = 15)$	 $(K = 10)$	 $(K = 10)$

We believe that our theoretical and experimental studies could be generalized to the Robin-Robin algorithm [48, 52], while substantial improvements can be made by using coarse grid correction [45], adaptive sampling strategy [17], special network structures [38], and more comprehensive error analysis [7, 71].

Acknowledgments

The research of Xuejun Xu is supported in part by the National Natural Science Foundation of China (grants 12071350 and 12331015). The research of Qi Sun is supported in part by the National Natural Science Foundation of China (grant 12201465), the Science and Technology Commission of Shanghai Municipality (grant 23JC1400502), and the Shanghai Municipal Science and Technology Major Project (No. 2021SHZDZX0100). This research was conducted using the computational resources and services of the HPC Center at the School of Mathematical Sciences, Tongji University. The authors are grateful to the anonymous reviewers for their valuable feedback.

References

[1] Chandrajit Bajaj, Luke McLennan, Timothy Andeen, and Avik Roy. Recipes for when physics fails: Recovering robust learning of physics informed neural networks. *Machine Learning: Science and Technology*, 4(1):015013, 2023.

[2] James H Bramble. Interpolation between Sobolev spaces in Lipschitz domains with an application to multigrid theory. *Mathematics of Computation*, 64(212):1359–1365, 1995.

[3] Susanne C Brenner, L Ridgway Scott, and L Ridgway Scott. *The Mathematical Theory of Finite Element Methods*, volume 3. Springer, 2008.

[4] Chang Che-Chia, Dai Chen-Yang, Hu Wei-Fan, Lin Te-Sheng, and Lai Ming-Chih. A hybrid neural-network and MAC scheme for Stokes interface problems. *East Asian Journal on Applied Mathematics*, 2024.

[5] Long Chen, Huayi Wei, and Min Wen. An interface-fitted mesh generator and virtual element methods for elliptic interface problems. *Journal of Computational Physics*, 334:327–348, 2017.

[6] Zhiming Chen and Jun Zou. Finite element methods and their convergence for elliptic and parabolic interface problems. *Numerische Mathematik*, 79(2):175–202, 1998.

[7] Duan Chenguang, Jiao Yuling, Lai Yanming, Lu Xiliang, Quan Qimeng, and Jerry Zhijian Yang. Deep Ritz methods for Laplace equations with Dirichlet boundary condition. *CSIAM Transactions on Applied Mathematics*, 3(4):761–791, 2022.

[8] Tim Dockhorn. A discussion on solving partial differential equations using neural networks. *arXiv preprint arXiv:1904.07200*, 2019.

[9] Victorita Dolean, Pierre Jolivet, and Frédéric Nataf. *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation*. SIAM, 2015.

[10] Suchuan Dong and Zongwei Li. Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 387:114129, 2021.

[11] Lawrence C Evans. *Partial Differential Equations*, volume 19. American Mathematical Society, 2010.
[12] Thomas-Peter Fries and Ted Belytschko. The extended/generalized finite element method: An overview of the method and its applications. *International Journal for Numerical Methods in Engineering*, 84(3):253–304, 2010.
[13] David Gilbarg, Neil S Trudinger, David Gilbarg, and NS Trudinger. *Elliptic Partial Differential Equations of Second Order*, volume 224(2). Springer, 1977.
[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
[15] Hailong Guo and Xu Yang. Deep unfitted Nitsche method for elliptic interface problems. *Communications in Computational Physics*, 31(4):1162–1179, 2022.
[16] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234. PMLR, 2016.
[17] Cuiyu He, Xiaozhe Hu, and Lin Mu. A mesh-free method using piecewise deep neural network for elliptic interface problems. *Journal of Computational and Applied Mathematics*, 412:114–358, 2022.
[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
[19] Alexander Heinlein, Axel Klawonn, Martin Lanser, and Janine Weber. Combining machine learning and domain decomposition methods for the solution of partial differential equations–A review. *GAMM-Mitteilungen*, 44(1):e202100001, 2021.
[20] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
[21] Wei-Fan Hu, Te-Sheng Lin, and Ming-Chih Lai. A discontinuity capturing shallow neural network for elliptic interface problems. *Journal of Computational Physics*, 469:111576, 2022.
[22] Zheyuan Hu, Ameya D. Jagtap, George Em Karniadakis, and Kenji Kawaguchi. When do extended physics-informed neural networks (XPINNs) improve generalization? *SIAM Journal on Scientific Computing*, 44(5):A3158–A3182, 2022.
[23] Zheyuan Hu, Ameya D Jagtap, George Em Karniadakis, and Kenji Kawaguchi. Augmented physics-informed neural networks (APINNs): A gating network-based soft domain decomposition methodology. *Engineering Applications of Artificial Intelligence*, 126:107183, 2023.
[24] Ameya D Jagtap and George Em Karniadakis. Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020.
[25] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proceedings of the Royal Society A*, 476(2239):20200334, 2020.
[26] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.
[27] Ameya D Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113–028, 2020.
[28] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and

proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pages 795–811. Springer, 2016.

[29] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[30] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[31] Isaac E Lagaris, Aristidis C Likas, and Dimitris G Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, 2000.

[32] Ming-Chih Lai, Che-Chia Chang, Wei-Syuan Lin, Wei-Fan Hu, and Te-Sheng Lin. A shallow Ritz method for elliptic problems with singular sources. *Journal of Computational Physics*, page 111547, 2022.

[33] Yunwen Lei. Stability and generalization of stochastic optimization with nonconvex and nonsmooth problems. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 191–227. PMLR, 2023.

[34] Randall J LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. SIAM, 2007.

[35] Randall J LeVeque et al. *Finite Volume Methods for Hyperbolic Problems*, volume 31. Cambridge University Press, 2002.

[36] Randall J LeVeque and Zhilin Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Journal on Numerical Analysis*, 31(4):1019–1044, 1994.

[37] Ke Li, Kejun Tang, Tianfan Wu, and Qifeng Liao. D3M: A deep domain decomposition method for partial differential equations. *IEEE Access*, 8:5283–5294, 2019.

[38] Sen Li, Yingzhi Xia, Yu Liu, and Qifeng Liao. A deep domain decomposition method based on Fourier features. *Journal of Computational and Applied Mathematics*, 423:114963, 2023.

[39] Wuyang Li, Xueshuang Xiang, and Yingxiang Xu. Deep domain decomposition method: Elliptic problems. In *Mathematical and Scientific Machine Learning*, pages 269–286. PMLR, 2020.

[40] Zhilin Li and Kazufumi Ito. *The Immersed Interface Method: Numerical Solutions of PDEs Involving Interfaces and Irregular Domains*. SIAM, 2006.

[41] Jacques Louis Lions and Enrico Magenes. *Non-homogeneous Boundary Value Problems and Applications: Vol. 1*, volume 181. Springer Science & Business Media, 2012.

[42] Yulong Lu, Jianfeng Lu, and Min Wang. A priori generalization analysis of the deep Ritz method for solving high dimensional elliptic partial differential equations. In *Conference on Learning Theory*, pages 3196–3241. PMLR, 2021.

[43] Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

[44] D Mercier. Minimal regularity of the solutions of some transmission problems. *Mathematical Methods in the Applied Sciences*, 26(4):321–348, 2003.

[45] Valentin Mercier, Serge Gratton, and Pierre Boudier. A coarse space acceleration of deep-DDM. *arXiv preprint arXiv:2112.03732*, 2021.

[46] Nicholas Metropolis and Stanislaw Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[47] Johannes Müller and Marius Zeinhofer. Error estimates for the deep Ritz method with boundary penalty. In *Mathematical and Scientific Machine Learning*, pages 215–230. PMLR, 2022.

[48] Xuyang Na and Xuejun Xu. Domain decomposition methods for diffusion problems with discontinuous coefficients revisited. *Communications in Computational Physics*, 35(1):212–238, 2024.

[49] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

[50] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[51] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588, 2009.

[52] Alfio Quarteroni and Alberto Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press, 1999.

[53] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[54] Hailong Sheng and Chao Yang. PFNN-2: A domain decomposed penalty-free neural network method for solving partial differential equations. *Communications in Computational Physics*, 32(4):980–1006, 2022.

[55] Khemraj Shukla, Ameya D Jagtap, and George Em Karniadakis. Parallel physics-informed neural networks via domain decomposition. *Journal of Computational Physics*, 447:110683, 2021.

[56] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

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

[58] Qi Sun, Xuejun Xu, and Haotian Yi. Domain decomposition learning methods for solving elliptic problems. *SIAM Journal on Scientific Computing*, 46(4):A2445–A2474, 2024.

[59] Ali Taghibakhshi, Nicolas Nytko, Tareq Uz Zaman, Scott MacLachlan, Luke Olson, and Matthew West. Learning interface conditions in domain decomposition solvers. *Advances in Neural Information Processing Systems*, 35:7222–7235, 2022.

[60] Andrea Toselli and Olof Widlund. *Domain Decomposition Methods: Algorithms and Theory*, volume 34. Springer Science & Business Media, 2004.

[61] Yu-Hau Tseng, Te-Sheng Lin, Wei-Fan Hu, and Ming-Chih Lai. A cusp-capturing PINN for elliptic interface problems. *Journal of Computational Physics*, 491:112359, 2023.

[62] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[63] Zhongjian Wang and Zhiwen Zhang. A mesh-free method for interface problems using the deep learning approach. *Journal of Computational Physics*, 400:108963, 2020.

[64] Hu Wei-Fan, Lin Te-Sheng, Tseng Yu-Hau, and Lai Ming-Chih. An efficient neural-network

and finite-difference hybrid method for elliptic interface problems with applications. *Communications in Computational Physics*, 33(4):1090–1105, 2023.

[65] Sidi Wu and Benzhuo Lu. INN: Interfaced neural networks as an accessible meshless approach for solving interface PDE problems. *Journal of Computational Physics*, 470:111588, 2022.

[66] Sidi Wu, Aiqing Zhu, Yifa Tang, and Benzhuo Lu. Convergence of physics-informed neural networks applied to linear second-order elliptic interface problems. *Communications in Computational Physics*, 33(2):596–627, 2023.

[67] Wei Wu, Xinlong Feng, and Hui Xu. Improved deep neural networks with domain decomposition in solving partial differential equations. *Journal of Scientific Computing*, 93(1):1–34, 2022.

[68] Daoqi Yang. A parallel nonoverlapping Schwarz domain decomposition method for elliptic interface problems. 1997.

[69] Daoqi Yang. Finite elements for elliptic problems with wild coefficients. *Mathematics and Computers in Simulation*, 54(4-5):383–395, 2000.

[70] Bing Yu et al. The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[71] Jiao Yuling, Lai Yanming, Lo Yisu, Wang Yang, and Yang Yunfei. Error analysis of deep Ritz methods for elliptic equations. *Analysis and Applications*, 22(01):57–87, 2024.

[72] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, 2020.

Appendices

A Detailed proof of Step 1) in Theorem 3.1

Recall that the function $\hat{u}_1 \in H^1(\Omega_1)$ is decomposed as a sum of two local functions, *i.e.*, $\hat{u}_1 = \hat{u}_1^{[k]} + g$, then

$$\begin{aligned}\mathcal{L}_2(\hat{u}_2) &= \frac{1}{2} b_1(\hat{u}_1^{[k]}+g, \hat{u}_1^{[k]}+g) + \frac{\beta_D}{2}\left(\|\hat{u}_1^{[k]}+g\|^2_{L^2(\partial\Omega_1\cap\partial\Omega)} + \|\hat{u}_1^{[k]}+g-u_\Gamma^{[k]}\|^2_{L^2(\Gamma)}\right) \\ &\quad -(f,\hat{u}_1^{[k]}+g)_1 \\ &= \mathcal{L}_1(\hat{u}_1^{[k]}) + b_1(\hat{u}_1^{[k]},g) - (f,g)_1 + \beta_D\left((\hat{u}_1^{[k]},g)_{L^2(\partial\Omega_1\cap\partial\Omega)} + (\hat{u}_1^{[k]}-u_\Gamma^{[k]},g)_{L^2(\Gamma)}\right) \\ &\quad + \frac{1}{2} b_1(g,g) + \frac{\beta_D}{2}\|g\|^2_{L^2(\partial\Omega_1)}.\end{aligned}$$

Note that the function $\hat{u}_1^{[k]} \in H^1(\Omega_1)$ is required to satisfy Eq. (3.6) in the sense of distributions, that is,

$$b_1(\hat{u}_1^{[k]},g) + \beta_D\left((\hat{u}_1^{[k]},g)_{L^2(\partial\Omega_1\cap\partial\Omega)} + (\hat{u}_1^{[k]}-u_\Gamma^{[k]},g)_{L^2(\Gamma)}\right) = (f,g)_1 \quad \text{for any } g \in H^2(\Omega_1),$$

and therefore we arrive at

$$\mathcal{L}_1(\hat{u}_1)=\mathcal{L}_1(\hat{u}_1^{[k]})+\int_{\Omega_1}\left(\frac{c_1}{2}|\nabla g|^2+\frac{1}{2}|g|^2\right)dx+\frac{\beta_D}{2}\int_{\partial\Omega_1}|g|^2ds\geq\mathcal{L}_1(\hat{u}_1^{[k]}),$$

for any $\hat{u}_1\in H^1(\Omega_1)$.

B Detailed proof of Step 1) in Theorem 3.2

Recall that the function $\hat{u}_2\in H^1(\Omega)$ is decomposed as a sum of two global functions, i.e., $\hat{u}_2=\hat{u}_2^{[k]}+g$, then

$$\begin{aligned}\mathcal{L}_2(\hat{u}_2)=&\frac{1}{2}b_2(\hat{u}_2^{[k]}+g,\hat{u}_2^{[k]}+g)-(f,\hat{u}_2^{[k]}+g)_2+b_1(\hat{u}_1^{[k]},\hat{u}_2^{[k]}+g)-(f,\hat{u}_2^{[k]}+g)_1\\&+(q,\hat{u}_2^{[k]}+g)_{L^2(\Gamma)}+\frac{\beta_N}{2}\|\hat{u}_2^{[k]}+g\|^2_{L^2(\partial\Omega)}\\=&\,\mathcal{L}_2(\hat{u}_2^{[k]})+b_2(\hat{u}_2^{[k]},g)-(f,g)_2+b_1(\hat{u}_1^{[k]},g)-(f,g)_1+(q,g)_{L^2(\Gamma)}\\&+\beta_N(\hat{u}_2^{[k]},g)_{L^2(\partial\Omega)}+\frac{1}{2}b_2(g,g)+\frac{\beta_N}{2}\|g\|^2_{L^2(\partial\Omega)},\end{aligned}$$

for any $g\in H^1(\Omega)$ defined over the entire domain. Note that $\hat{u}_1^{[k]}\in H^1(\Omega_1)$ and $\hat{u}_2^{[k]}|_{\Omega_2}\in H^1(\Omega_2)$ are required to satisfy (3.6) and (3.13) in the sense of distributions, namely,

$$b_1(\hat{u}_1^{[k]},g_1)-(c_1\nabla\hat{u}_1^{[k]}\cdot\boldsymbol{n}_1,g_1)_{L^2(\partial\Omega_1)}=(f,g_1)_1\quad\text{for any } g_1\in H^1(\Omega_1),$$

and

$$b_2(\hat{u}_2^{[k]},g_2)-(c_2\nabla\hat{u}_2^{[k]}\cdot\boldsymbol{n}_2,g_2)_{L^2(\partial\Omega_2)}=(f,g_2)_2\quad\text{for any } g_2\in H^1(\Omega_2),$$

respectively, we then have by the boundary conditions imposed in (3.14) and (3.13) that

$$\begin{aligned}\mathcal{L}_2(\hat{u}_2)=&(c_1\nabla\hat{u}_1^{[k]}\cdot\boldsymbol{n}_1+\beta_N\hat{u}_2^{[k]},g)_{L^2(\partial\Omega_1\cap\partial\Omega)}+(c_2\nabla\hat{u}_2^{[k]}\cdot\boldsymbol{n}_2+\beta_N\hat{u}_2^{[k]},g)_{L^2(\partial\Omega_2\cap\partial\Omega)}\\&+\,(c_1\nabla\hat{u}_1^{[k]}\cdot\boldsymbol{n}_1+c_2\nabla\hat{u}_2^{[k]}\cdot\boldsymbol{n}_2+q,g)_{L^2(\Gamma)}+\frac{1}{2}b_2(g,g)+\frac{\beta_N}{2}\|g\|^2_{L^2(\partial\Omega)}+\mathcal{L}_2(\hat{u}_2^{[k]})\\=&\,\mathcal{L}_2(\hat{u}_2^{[k]})+\int_{\Omega_2}\left(\frac{c_2}{2}|\nabla g|^2+\frac{1}{2}|g|^2\right)dx+\frac{\beta_N}{2}\int_{\partial\Omega}|g|^2ds\geq\mathcal{L}_2(\hat{u}_2^{[k]}),\end{aligned}$$

for any $\hat{u}_2\in H^1(\Omega)$.

C

The hyperparameter configuration used for our numerical experiments is summarized in Table 11, followed by the iterative network solutions using different deep learning-based methods for each numerical example.

Table 11: Hyperparameter configuration used for our experiments.

		Training Datasets (N_{Ω_i}, N_{D_i}, N_Γ)	Penalty Coefficients (β_D, β_N)	Network (depth, width)	Trainable Parameters
$c_2=10^3$	DeepDDM	$(20k,5k,5k)$	$2k$	$(4,100)$	20701
	DNLA (PINNs)	$(20k,5k,5k)$	$(2k,2m)$	$(4,100)$	20701
	DNLA (deep Ritz)	$(20k,5k,5k)$	$(2k,2m)$	$(4,100)$	20701
$c_2=1$	DeepDDM	$(2k,5k,5k)$	$(2k,2k)$	$(4,100)$	20701
	DNLA (PINNs)	$(2k,5k,5k)$	$(2k,2k)$	$(4,100)$	20701
	DNLA (deep Ritz)	$(2k,5k,5k)$	$(2k,2k)$	$(4,100)$	20701

Table 12: Iterative solutions $\hat{u}^{[k]}$ of different methods for (4.1) with $(c_1,c_2)=(1,10^3)$.

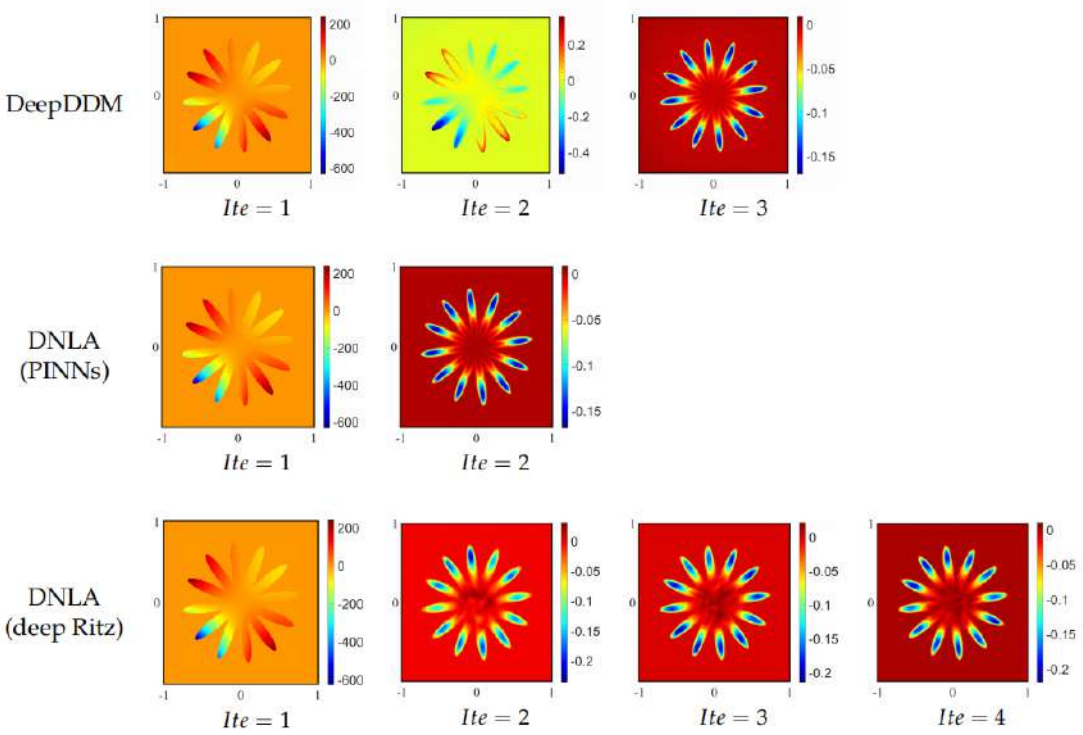


Table 13: Iterative solutions $\hat{u}^{[k]}$ of different methods for (4.1) with $(c_1,c_2)=(1,1)$.

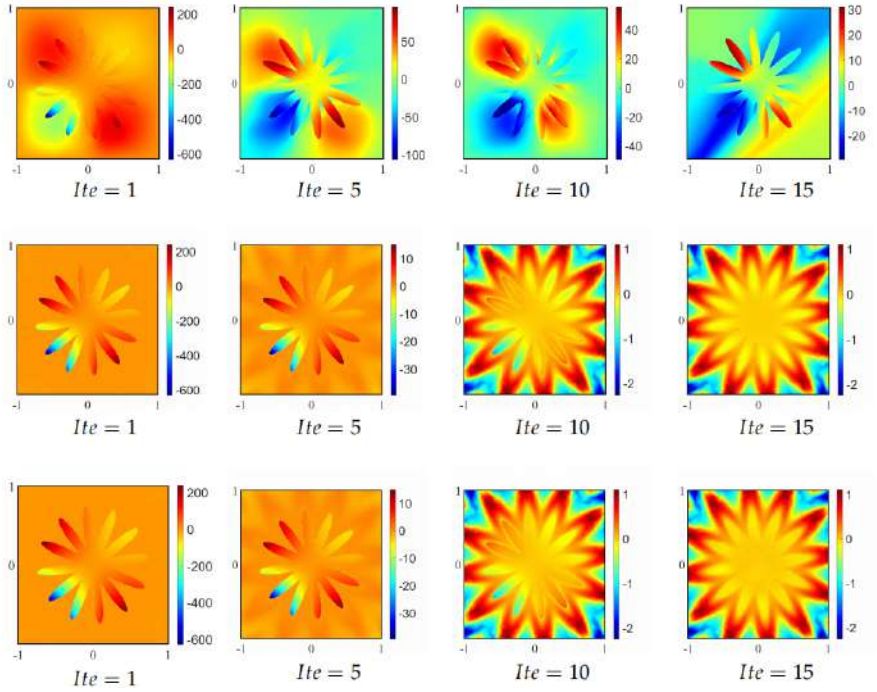


Table 14: The iterative solutions $\hat{u}^{[k]}(x,y;\theta)$ of different methods for numerical example (4.4) with $(c_1,c_2)=(1,10^3)$.

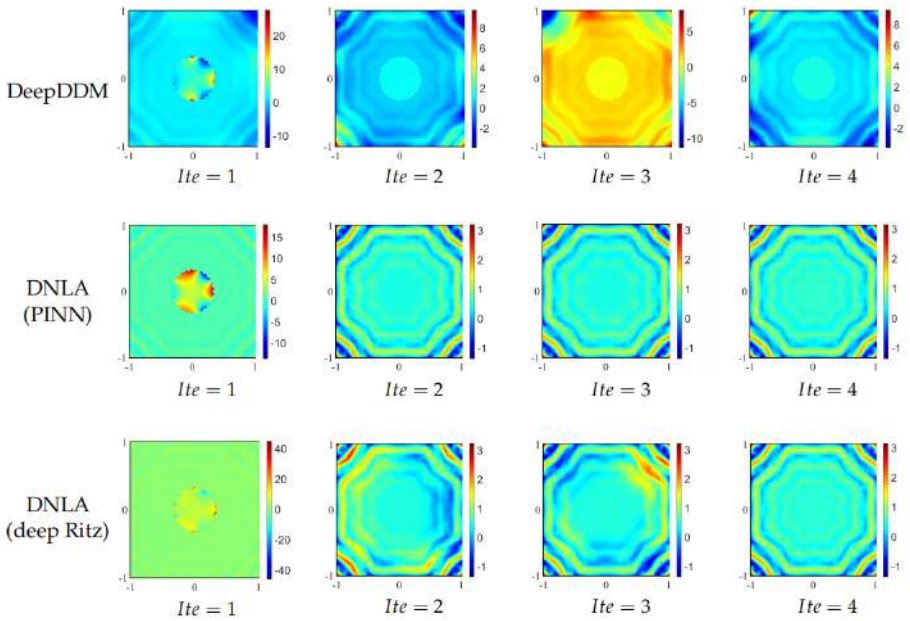


Table 15: The iterative solutions $\hat{u}^{[k]}(x,y;\theta)$ of different methods for numerical example (4.4) with $(c_1,c_2)=(1,1)$.

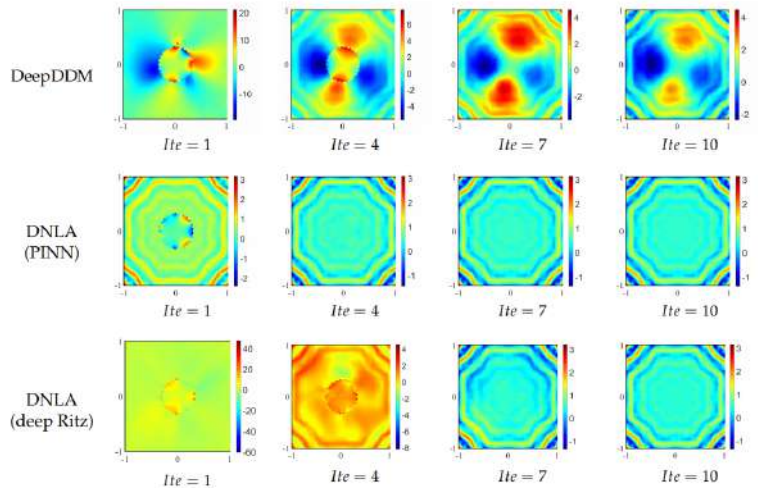


Table 16: Iterative solutions $\hat{u}^{[k]}$ of different methods for (4.2) with $(c_1,c_2)=(1,1)$.

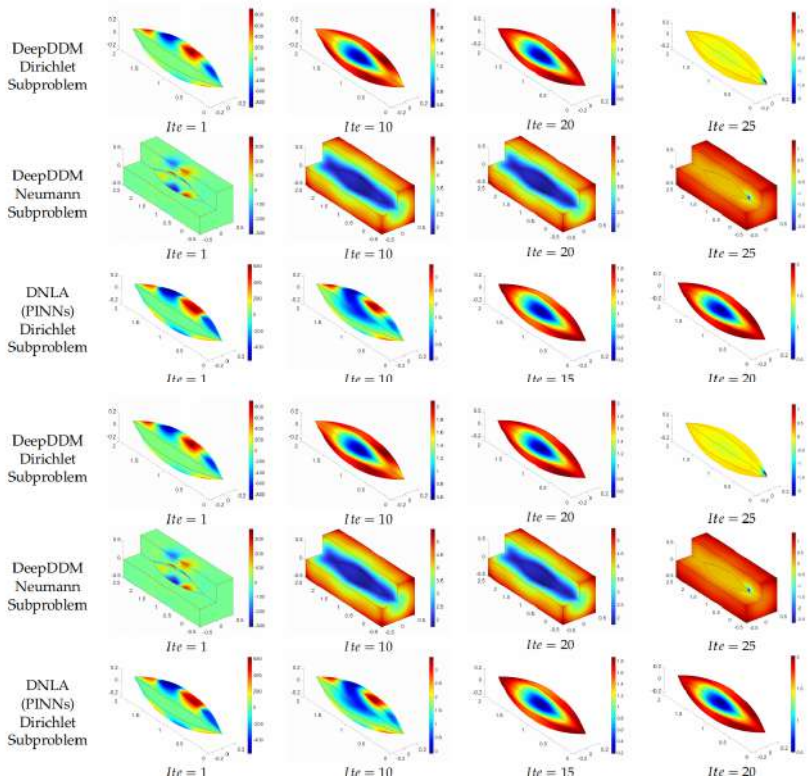


Table 17: Iterative solutions $\hat{u}^{[k]}$ of different methods for (4.2) with $(c_1,c_2)=(1,10^3)$.