D3M: A Deep Domain Decomposition Method for Partial Differential Equations

KE LI [1,2,3], **(Student Member, IEEE), KEJUN TANG** [1,2,3], **TIANFAN WU** [4], **AND QIFENG LIAO** [1]

[1] School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China
[2] Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China
[3] University of Chinese Academy of Sciences, Beijing 100049, China
[4] Viterbi School of Engineering, University of Southern California, Los Angeles, CA 90089, USA

Corresponding author: Qifeng Liao (liaoqf@shanghaitech.edu.cn)

ABSTRACT A state-of-the-art deep domain decomposition method (D3M) based on the variational principle is proposed for partial differential equations (PDEs). The solution of PDEs can be formulated as the solution of a constrained optimization problem, and we design a hierarchical neural network framework to solve this optimization problem. Through decomposing a PDE system into components parts, our D3M builds local neural networks on physical subdomains independently (which can be implemented in parallel), so as to obtain efficient neural network approximations for complex problems. Our analysis shows that the D3M approximation solution converges to the exact solution of the underlying PDEs. The accuracy and the efficiency of D3M are validated and demonstrated with numerical experiments.

INDEX TERMS Domain decomposition, deep learning, mesh-free, parallel computation, PDEs, physics-constrained.

I. INTRODUCTION

Partial differential equations (PDEs) are among the most ubiquitous tools employed in describing computational science and engineering problems. When modeling complex problems, the governing PDEs are typically expensive to solve through traditional numerical methods, e.g., the finite element methods [1]. While principal component analysis [2], [3] (PCA), proper orthogonal decomposition [4], [5] (POD) and reduced basis methods [6]–[11] are classical approaches for model reduction to reduce the computational costs, deep learning [12] currently gains a lot of interests for efficiently solving PDEs. There are mathematical guarantees called universal approximation theorems [13] stating that a single layer neural network can approximate most functions in Sobolev spaces. Although there is still a lack of theoretical frameworks for explaining the effectiveness of multilayer neural networks, deep learning has become a widely used tool. Marvelous successful practices of deep neural networks encourages their applications to different areas, where the curse of dimensionality is a tormenting issue.

The associate editor coordinating the review of this manuscript and approving it for publication was Gang Mei.

New approaches are actively proposed to solve PDEs based on deep learning techniques. Weinan *et al.* [14], [15] connect deep learning with dynamic system and propose a deep Ritz method (DRM) for solving PDEs via variational methods. Raissi *et al.* [16]–[19] develop physics-informed neural networks which combine observed data with PDE models. By leveraging a prior knowledge that the underlying PDE model obeys a specific form, they can make accurate predictions with limited data. Long *et al.* [20] present a feed-forward neural network, called PDE-Net, to accomplish two tasks at the same time: predicting time-dependent behavior of an unknown PDE-governed dynamic system, and revealing the PDE model that generates observed data. Later, Sirignano and Spiliopoulos [21] propose a deep Galerkin method (DGM), which is a meshfree deep learning algorithm to solve PDEs without requiring observed data (solution samples of PDEs). When a steady-state high-dimensional parametric PDE system is considered, Zhu *et al.* [22], [23] propose Bayesian deep convolutional encoder-decoder networks for problems with high-dimensional random inputs. Recently, Wu *et al.* [24], [25] and Zhang and Lin [26] contributes deep learning strategies to discover of physical laws.

When considering computational problems arising in practical engineering, e.g. aeronautics and astronautics, systems

are typically designed by multiple groups along disciplinary. The complexity of solving large-scale problems may take an expensive cost of hardware. The balance of accuracy and generalization is also hard to trade off. For this reason, decomposing a given system into component parts to manage the complexity is a strategy, and the domain decomposition method is a traditional numerical method to achieve this goal. Schwarz [27] proposes an iterative method for solving harmonic functions. Then this method is improved by Sobolev [28], Michlin [29], Lions [30], [31]. Domain decomposition is also employed for optimal design or control [32], for decomposing a complex design task (e.g., decomposition approaches to multidisciplinary optimiza tion [33], [34]), and for uncertainty analysis of models governed by PDEs [35], [36].

In this work, we propose a variational deep learning solver based on domain decomposition methods, which is referred to as the deep domain decomposition method (D3M) to implement parallel computations along physical subdomains. Especially, efficient treatments of complex boundary conditions are developed. Solving PDEs using D3M has several benefits:

A. COMPLEXITY AND GENERALIZATION
D3M manages complexity at the local level. Overfitting is a challenging problem in deep learning. The risk of overfitting can be reduced by splitting the physical domain into subdomains, so that each network focuses on a specific subdomain.

B. MESH-FREE AND DATA-FREE
D3M constructs and trains networks under variational formulation. So, it does not require given data, which can be potentially used for complex and high-dimensional problems.

C. PARALLEL COMPUTATION
The computational procedures of D3M are in parallel for different subdomains. This feature helps D3M work efficiently on large-scale and multidisciplinary problems.

In this work, D3M is developed based on iterative domain decomposition methods. The development of using domain decomposition leads to an independent model-training procedure in each subdomain in an "offline" phase, followed by assembling global solution using pre-computed local information in an "online" phase. Section II reviews iterative overlapping domain decomposition methods. Section III presents the normal variational principle informed neural networks, and our D3M algorithms. A convergence analysis of D3M is discussed in Section IV, and a summary of our full approach is presented in Section V. Numerical studies are discussed in Section VI. Finally, Section VII concludes the paper.

II. OVERLAPPING DOMAIN DECOMPOSITION
The Schwarz method [27] is the most classical example of domain decomposition approach for PDEs, and it is still efficient with variant improvements [37]–[39].
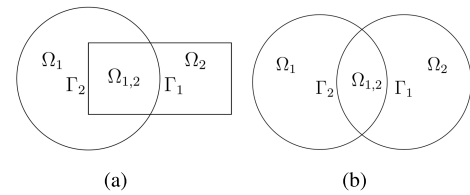

FIGURE 1. Partition into two overlapping subdomains.

Given a classical Poisson's equation

$$\begin{cases} -\Delta u(x, y) = f(x, y), & \text{in } \Omega, \\ u(x, y) = 0, & \text{on} \partial\Omega. \end{cases} \quad (1)$$

We divide Ω into two overlapping subdomains Ω_i, $i = 1, 2$ (see Figure 1), where

$$\Omega = \Omega_1 \cup \Omega_2, \quad \Gamma_1 := \partial\Omega_1 \cap \Omega_2,$$
$$\Gamma_2 := \partial\Omega_2 \cap \Omega_1, \quad \Omega_{1,2} := \Omega_1 \cap \Omega_2. \quad (2)$$

Note that we here review the domain decomposition method using two subdomains, while our D3M considers any number of subdomains.

The original formula named Schwarz alternating method is reviewed as follows. Let u^0 be an initial guess defined in Ω and vanishing on $\partial\Omega$. For $k \geq 0$, we define sequences u_i^k where u_i^k denotes u^k restricted in Ω_i. The u_i^{k+1} is determined from an iteration algorithm:

$$\begin{cases} -\Delta u_1^{k+1/2}(x, y) = f(x, y), & \text{in } \Omega_1, \\ u_1^{k+1/2}(x, y) = u_2^k(x, y), & \text{on } \Gamma_1, \\ u_1^{k+1/2}(x, y) = 0, & \text{on } \partial\Omega_1 \cap \partial\Omega \end{cases} \quad (3)$$

and

$$\begin{cases} -\Delta u_2^{k+1}(x, y) = f(x, y), & \text{in } \Omega_2, \\ u_2^{k+1}(x, y) = u_1^{k+1/2}(x, y), & \text{on } \Gamma_2, \\ u_2^{k+1}(x, y) = 0, & \text{on} \partial\Omega_2 \cap \partial\Omega. \end{cases} \quad (4)$$

III. DEEP DOMAIN DECOMPOSITION METHOD WITH VARIATIONAL PRINCIPLE
Before introducing D3M, we first give a brief introduction of variational principle. In this section, we consider the Poisson's equation and reformulate (1) as a constrained minimization problem, and then we introduce the D3M algorithm.

A. VARIATIONAL PRINCIPLE
The Poisson's equation with the homogeneous Dirichlet boundary condition is (1), and we consider the situation that $f \in \mathcal{L}^2(\Omega)$ and Ω is a square domain in this section. The idea of the standard Deep Ritz method is based on the variational principle. That is, the PDE can be derived by a functional minimization problem as described in the following proposition.

Proposition 1: Solving the Poisson's equation (1) *is equivalent to an optimization problem*

$$\min_u E(u) = \int_\Omega \frac{1}{2}|\nabla u|^2 \, dxdy - \int_\Omega f \cdot u dxdy,$$

s.t. $u = 0$ on $\partial\Omega$.　　　　(5)

The Lagrangian formula of (5) *is given by*

$$L(u, q) = \int_\Omega \frac{1}{2}|\nabla u|^2 dxdy - \int_\Omega u \cdot f\, dxdy + q\int_{\partial\Omega} udxdy, \quad (6)$$

where q is the Lagrange multiplier.

Definition 1: $\mathcal{H}(\mathbf{div})$ *denotes symmetric tensor-fields in* H^1 *space, in which functions are square integrable and have square integrable divergence.*

We employ a mixed residual loss [23] following Hellinger-Reissner principle [40]. With an additional variable $\tau \in \mathcal{H}(\mathbf{div})$, which represents flux, we can turn Equation (1) into

$$\begin{cases} \tau = -\nabla u, & \text{in } \Omega, \\ \nabla \cdot \tau = f, & \text{in } \Omega. \end{cases} \quad (7)$$

The mixed residual loss is

$$L(\tau, u, q) = \int_\Omega [(\tau + \nabla u)^2 + (\nabla \cdot \tau - f)^2]dxdy + q\int_{\partial\Omega} udxdy. \quad (8)$$

B. VARIATIONAL PRINCIPLE INFORMED NEURAL NETWORKS

Though the Poisson's equation (1) is reformulated as an optimization problem, it is intractable to find the optimum in an infinite-dimensional function space. Instead, we seek to approximate the solution $u(x, y)$ by neural networks. We utilize $\mathbf{N}_u(x, y; \theta_u)$, $\mathbf{N}_\tau(x, y; \theta_\tau)$ to approximate the solution u and the flux τ in domain Ω, where θ_u and θ_τ are the parameters to train. The input is the spatial variable in Ω, and the outputs represent the function value corresponding to the input. With these settings, we can train a neural network by variational principle to represent the solution of Poisson's equation. The functional minimization problem (8) turns into the following optimization problem

$$\min_{\theta = \{\theta_u, \theta_\tau\}} \int_\Omega \left[(\mathbf{N}_\tau + \nabla\mathbf{N}_u)^2 + (\nabla \cdot \mathbf{N}_\tau - f)^2 \right] dxdy + q\int_{\partial\Omega} \mathbf{N}_u^2 dxdy. \quad (9)$$

Remark 1: In practical implementation, $\mathbf{N}_u$ *and* $\mathbf{N}_\tau$ *are embedded in one network* $\mathbf{N}$ *parameterized with* θ, *and the two outputs of* $\mathbf{N}$ *denote the function values of* $\mathbf{N}_u$ *and* $\mathbf{N}_\tau$ *respectively.*

Therefore, the infinite-dimensional optimization problem (5) is transformed into a finite-dimensional optimization problem (9). Our goal is to find the optimal (or sub optimal) parameters θ to minimize the loss in (9). To this end, we choose a mini-batch of points randomly sampled in Ω. These data points can give an estimation of the integral in (9) and the gradient information to update the parameters θ. For example, a mini-batch points $\{(x_i, y_i)\}_{i=1}^{m+n}$ are drawn in $\bar{\Omega}$ randomly, where $\{(x_i, y_i)\}_{i=1}^{m}$ in Ω and $\{(x_i, y_i)\}_{i=m+1}^{m+n}$ on $\partial\Omega$.

Then the parameters can be updated by using optimization approaches

$$\theta^{(k+1)} = \theta^{(k)} - \nabla_\theta \frac{1}{m} \sum_{i=1}^{m} [(\mathbf{N}_\tau^{(i)} + \nabla\mathbf{N}_u^{(i)})^2 + (\nabla \cdot \mathbf{N}_\tau^{(i)} - f)^2]$$

$$- \nabla_\theta \frac{1}{n} \sum_{j=1}^{n} (q \cdot \mathbf{N}_u^{(j)})^2. \quad (10)$$

C. IMPLEMENTATION DETAILS FOR NEURAL NETWORKS

This section provides details for the architecture of our neural networks.

For giving a direct-viewing impression, we show the implementation with a plain vanilla densely connected neural network to introduce how the structure works in Figure 2. For illustration only, the network depicted consists of 2 layers with 6 neurons in each layer. The network takes input variables x, y and outputs $u, \tau = [\tau_x, \tau_y]$. The number of neurons in each layer is M and σ denotes an element-wise operator

$$\sigma(x) = (\phi(x_1), \phi(x_2), \dots, \phi(x_M)), \quad (11)$$

where ϕ is called the activation function. There are some commonly used activation functions such as the sigmoidal function, the tanh function, the rectified linear units (ReLU) function [41], and the Leaky ReLU function [42]. We employ the tanh function (our target function is twice differentiable), and the automatic differentiation is obtained by using PyTorch [43]. The total loss function comprises the residual loss terms L_1, L_2 and the Lagrangian term which guarantees the constraint conditions. The parameters are trained with backpropagating gradients of the loss function and the optimizer is L-BFGS [44] where the learning rate is 0.5. In practice, the model architecture of neural networks is the residual network (ResNet) [45]. These residual networks are easier to optimize, and it can gain accuracy from considerably increased depth. The structure of ResNet improves the result of deep networks, because there are more previous information retained. A brief illustration of ResNet is in Figure 3.

D. DEEP DOMAIN DECOMPOSITION METHOD

In this part, we propose the main algorithms of our D3M. Because the physics-constrained neural network is mesh-free, we improve Schwarz alternating method with a better inhomogeneous D3M sampling method at junctions Γ_i to accelerate convergence. We note the performance of normal deep variational networks and mixed residual networks can deteriorate when the underlying problem has inhomogeneous boundary conditions. Our treatment to overcome this weakness is to introduce the following boundary function.

Definition 2 (Boundary function): A smooth function $\mathfrak{g}(x, y)$ *is called a boundary function associated with* Ω *if*

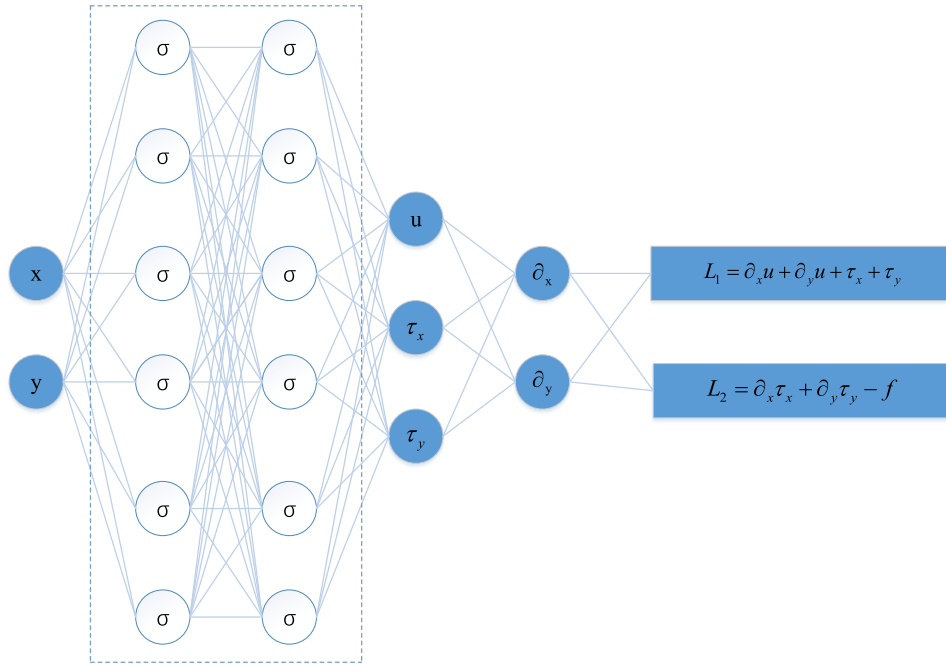$$\mathfrak{g}(x, y) = e^{-a \cdot d(x, y, \partial\Omega)} u(x, y), \quad (x, y) \in \Omega, \quad (12)$$

FIGURE 2. Illustration of the neural networks. x, y are inputs, u, τ_x, τ_y are outputs and the dashed box with σ means the architecture of plain fully-connected neural networks.
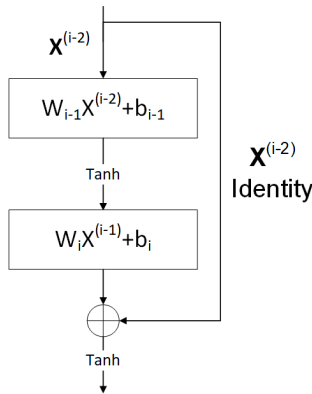

FIGURE 3. The residual network building block of our method.

where $a \gg 1$ *is a coefficient, the notation $d(x, y, \partial\Omega)$ denotes the shortest Euclidean distance between (x, y) and $\partial\Omega$. If the point (x, y) is on the boundary, $\mathfrak{g}(x, y) = u(x, y)$. If not, the value of $\mathfrak{g}(x, y)$ decreases to zero sharply. And we define $v := u - \mathfrak{g}$, where v satisfies*

$$\begin{cases} -\Delta v = f + \Delta\mathfrak{g}, & \text{in } \Omega, \\ v = 0, & \text{on } \partial\Omega. \end{cases} \quad (13)$$

Letting $v_i = u_i - \mathfrak{g}_i$ on each subdomain, Equation (7) can be represented as

$$\begin{cases} \tau = -\nabla v_i, & \text{in } \Omega_i, \\ \nabla \cdot \tau = f + \Delta\mathfrak{g}_i, & \text{in } \Omega_i. \end{cases} \quad (14)$$

The mixed residual loss is

$$L(\tau_i, v_i, q) = \int_{\Omega_i} [(\tau_i + \nabla v_i)^2 + (\nabla \cdot \tau_i - f - \Delta\mathfrak{g}_i)^2]dxdy$$

$$+ q \int_{\partial\Omega_i} v_i^2 \, dxdy,$$

$$\approx \frac{1}{m_1}\sum_{k=1}^{m_1}[(\tau_i^{(k)} + \nabla v_i^{(k)})^2 + (\nabla \cdot \tau_i^{(k)} - f - \Delta\mathfrak{g}_i)^2]$$

$$+ \frac{1}{m_2}\sum_{j=1}^{m_2} q \cdot (v_i^{(j)})^2. \quad (15)$$

It should be noted that, the integration is completed by Monte Carlo, such that the domain decomposition reduces the variance of samples significantly with the same number of data because the area of samples becomes smaller.

The procedure of D3M is as follows. We first divide the domain Ω into d subdomains, and each two neighboring subdomains are overlapping. The local solution of PDEs on each subdomain is replaced by neural networks which can be trained through the variational principle, where the global solution on the whole domain consists of these local solutions on subdomains. To be more precise, let Γ_i denote decomposed junctions, θ is initial weights of neural networks, η is the threshold of accuracy, S_i and g_i are the samples generated in Ω_i and on interface Γ_i to evaluate the output of networks in each iteration, $\hat{S}_i$ are training samples in subdomain Ω_i, $\hat{g}_i$ are training samples on Γ_i, n is training time in each iteration, m_1 and m_2 are batch sizes, N_u is the neural networks for u,

---

Algorithm 1 Deep Domain Decomposition

1: **Input:** $\Omega = (x_0, x_1) \times (y_0, y_1)$, p, Γ_i, η, θ, n, m_1, m_2.
2: **Initialize:** $\epsilon = 10 \times \eta$, $k = 0$, $gv_i^0 = \mathbf{0}$, S_i, g_i.
3: Divide the physical domain Ω into $\Omega_1, \cdots, \Omega_p$.
4: **while** $\epsilon > \eta$ **do**
5:     Run Algorithm (2) in each subdomain in parallel.
6:     $\epsilon = \frac{1}{p} \sum_{i=1}^{p} \|Sol_i^{(k+1)} - Sol_i^{(k)}\|_2^2$.
7:     $k = k + 1$.
8: **end while**
9: Merge p parts $Sol_i^{(k)}$ and get $Dnn_{sol}^{(k)}$.
10: **Return:** $Dnn_{sol}^{(k)}$.

Algorithm 2 Training for Subdomain Ω_i

1: **Input:** S_i, gv_i^k, g_i^k, n, m_1, m_2.
2: Construct function $\mathfrak{g}_i$ using value of gv_i^k, $v_i = \mathbf{N}_u - \mathfrak{g}_i$.
3: **for** n steps **do**
4:     Sample minibatch of m_1 samples $\hat{S}_i = \{(x_i, y_i)\}_{i=1}^{m_1}$ in Ω_i.
5:     Sample minibatch of m_2 samples $\hat{g}_i = \{(x_i, y_i)\}_{i=1}^{m_2}$ on $\partial\Omega_i$.
6:     Update the parameters θ_i by descending its stochastic gradient:

$$\theta_i^{(k+1)} = \theta_i^{(k)} - \nabla_\theta \frac{1}{m_1} \sum_{k=1}^{m_1} [(\mathbf{N}_\tau^{(k)} + \nabla v_i^{(k)})^2$$

$$+ (\nabla \cdot \mathbf{N}_\tau^{(k)} - f - \Delta\mathfrak{g}_i)^2] - \nabla_\theta \frac{1}{m_2} \sum_{j=1}^{m_2} (q \cdot v_i^{(j)})^2.$$

7: **end for**
8: $Sol_i^{(k+1)} = \mathbf{N}_u(S_i)$.
9: $gv_i^{(k+1)} = \mathbf{N}_u(g_i^{(k)})$.
10: **Return:** $Sol_i^{(k+1)}$, $gv_i^{(k+1)}$.

N_τ is the neural networks for τ, k is the iteration time, and $Sol_i^{(k+1)}$ is the output of networks for subdomain Ω_i in $(k+1)$-th iteration. The formal description of D3M is presented in Algorithm 1.

IV. ANALYSIS

While mixed residual formulation is a special case, we consider the basic functional formulation first,

$$J(u) = \int_\Omega \frac{1}{2} \nabla u \cdot \nabla u - fu dS. \qquad (16)$$

Definition 3: Given m closed subspaces $\{V_i\}_{i=1}^m$ and $V = \sum_{i=1}^m V_i \in C^2(\Omega)$, for any $R < \infty$, and a proper, lower semi-continuous, coercive convex functional $J : V \to \Re$, we denote $K_R := \{u \in V | J(u) < R\}$.

Assumption 1: $J \in C^1(K_R)$ and $\exists \alpha_R > 0$ s.t. $\forall v, u \in K_R$

$$J(v) - J(u) - (J'(u), v - u) \geq \alpha_R |v - u|^2, \qquad (17)$$

where J' is uniformly continuous on K_R.

Proposition 2: (P.L.Lions, 1989) *The alternating Schwarz method* (3) *and* (4) *converges to the solution of u of Equation* (14). *The error bound of $\hat{u}_1^{k+1}$ and $\hat{u}_2^{k+1}$ can be estimated via maximum principle* [31], [46], $\exists \rho \in (0, 1)$ *such that for $\forall k \geq 0$*

$$\|u|_{\Omega_i} - \hat{u}_i^{k+1}\|_{L^\infty(\Omega_i)} \leq \rho^k \|u|_{\Omega_i} - \hat{u}_i^0\|_{L^\infty(\Omega_i)}. \qquad (18)$$

where constants ρ is close to one if the overlapping region $\Omega_{i,j}$ is thin.

Lemma 1 (K. Hornik, 1991): On each subdomain Ω_i, neural network $\mathbf{N}_i$ with continuous derivatives up to order K are universal approximators in Sobolev space with an order K, which means $\mathbf{N}_i \in H^1(\Omega_i)$.

Lemma 2 (P. L. Lions, 1988): If the variational formulation (16) satisfies the assumption (1), then it follows that there exists a sequential $u_n \in V_i$ obtained by Schwarz alternating method converges to the minimum u_i^* of $J(u_i)$ on each subdomain Ω_i.

Theorem 1: $J_i(\mathbf{N}_i)$ denotes the objective function on the subdomain Ω_i. Under above assumptions, for $\forall \epsilon > 0$, $\exists M > 0$, while iteration times $k > M$, $\mathbf{N}_i^k$ converges to optimal solution u_i^* of $J(u_i)$ in subdomain Ω_i for a constant $C > 0$

$$|\mathbf{N}_i^k - u_i^*|^2 \leq C\epsilon^{1/2} \text{ in } \Omega_i. \qquad (19)$$

For concision, we use $\mathbf{N}_i$ to represent $\mathbf{N}_i^k$ in the following part.

Proof: $u_n \in V_i$ denotes the sequential in Lemma 2, there exist

$$|u_n - u_i^*| \leq \frac{C_0}{\alpha_R} \omega |u_{n+1} - u_n|, \qquad (20)$$

where $C_0 > 0$ is a constant, $\omega |u_{n+1} - u_n| = |J'(u_{n+1}) - J'(u_n)|$.

Then with Lemma 1 [13], in each subdomain Ω_i, neural network $\mathbf{N}_i \in H^1$. If the training times k in each subdomain is enough, the universal approximation ensures the distance between functions $\mathbf{N}_i$ and u_n is close enough in the Sobolev space, with the constant $C_1 > 0$

$$|\mathbf{N}_i - u_n|^2 \leq C_1^2 \epsilon^2. \qquad (21)$$

While u_n converging to the minimum of u_i^* of $J(u)$, by the optimality conditions it is clear that

$$(J'(u_{n+1}), u_{n+1} - u_n) < \epsilon,$$
$$(J'(u_{n+1}), u_{n+1} - u_i^*) < \epsilon. \qquad (22)$$

Under the assumption, $\exists C > 0$ and the difference between two iterations can be represented as

$$|u_{n+1} - u_n|^2 \leq \frac{1}{\alpha_R} |J(u_{n+1}) - J(u_n)| \leq \frac{1}{\alpha_R} \epsilon. \qquad (23)$$

Consider equation (20), (21) and (23), we have

$$|\mathbf{N}_i - u_i^*| \leq |\mathbf{N}_i - u_n| + |u_n - u_i^*|,$$
$$\leq C_1 \epsilon + \frac{C_0}{\alpha_R} \omega |u_{n+1} - u_n|,$$

$$\leq C\epsilon^{1/2}. \tag{24}$$

□

Theorem 2: For a given boundary function $\mathfrak{g}$ and a fixed q, the optimal solution $\mathbf{N}_u^*$ of Equation (9) and v^* of Equation (14) satisfy $\mathbf{N}_u^* = v^* + \mathfrak{g}$.

Proof: We use τ and ϕ to denote $-\nabla \mathbf{N}_u$ and $-\nabla v$ respectively.

$$\begin{aligned}
L(\phi, v) &= \int_\Omega [(\phi + \nabla v)^2 + (\nabla \cdot \phi - f - \Delta\mathfrak{g})^2] dxdy \\
&\quad + q\int_{\partial\Omega} v dxdy, \quad \text{if } v|_{\partial\Omega} = 0 \\
&= \int_\Omega [(\tau + \nabla\mathfrak{g} + \nabla u - \nabla\mathfrak{g})^2 + (\nabla\cdot\tau - f)^2] dxdy \\
&\quad + q\int_{\partial\Omega}(u - \mathfrak{g})dxdy \\
&= L(\tau, u). \tag{25}
\end{aligned}$$

Function v we optimized is also the optimal solution $\mathbf{N}_u$ with the formula $\mathbf{N}_u^* = v^* + \mathfrak{g}$. □

Up to now, we prove that D3M can solve steady Poisson's equation with variational formulations. Then, we extend D3M to more general quasilinear parabolic PDEs (26) with physics-constrained approaches.

$$\begin{cases}
\text{div}(\alpha(x, u_i(x), \tau_i(x))) \\
\quad + \gamma(x, u_i(x), \tau_i(t, x)) = 0, & x \in \Omega_i, \\
u_i(t, x) = 0, & x \in \partial\Omega_i,
\end{cases} \tag{26}$$

where τ_i denotes ∇u_i, and $\Omega_i \in \mathbb{R}^d$ are decomposed boundary sets with smooth boundaries $\partial\Omega_i$. We recall the network space of subdomain Ω_i with generated data according to [13]

$$\mathfrak{N}_i^n(\sigma) = \left\{ h(x) : \mathbb{R}^k \to \mathbb{R} | h(x) = \sum_{j=1}^n \beta_j \sigma\left(\alpha_j^T x - \theta_j\right) \right\} \tag{27}$$

where σ is any activation function, $\mathbf{x} \in \mathbb{R}^k$ is one set of generated data, $\beta \in \mathbb{R}^n$, $\alpha \in \mathbb{R}^{k\times n}$ and $\theta \in \mathbb{R}^{k\times n}$ denote coefficients of networks. Set $\mathfrak{N}_i(\sigma) = \bigcup_{i=1}^\infty \mathfrak{N}_i^n(\sigma)$. Under the universal approximation of neural networks Lemma 1, in each subdomain the neural networks f_i^n satisfies

$$\begin{cases}
\text{div}(\alpha(x, f_i^n(x), \tau_i^n(x)) \\
\quad + \gamma(x, f_i^n(x), \tau_i^n(t, x)) = h^n, & x \in \Omega_i, \\
f_i^n(t, x) = b^n, & x \in \partial\Omega_i,
\end{cases} \tag{28}$$

where h^n and b^n satisfy

$$\|h^n\|_{2,\Omega_i}^2 + \|b^n\|_{2,\partial\Omega_i}^2 \to 0, \quad \text{as } n \to \infty. \tag{29}$$

For the following part of analysis, we make some assumptions.

Assumption 2: • There is a constant $\mu > 0$ and positive functions $\kappa(x)$, $\lambda(x)$ such that for all $x \in \Omega_i$ we have

$$\|\alpha(x, u_i, \tau_i)\| \leq \mu(\kappa(x) + \|\tau_i\|),$$

and

$$|\gamma(x, u_i, \tau_i)| \leq \lambda(x)\|\tau_i\|,$$

with $\kappa \in L^2(\Omega_i)$, $\lambda \in L^{d+2+\eta}(\Omega_i)$ for some $\eta > 0$.
- $\alpha(x, u_i, \nabla u_i)$ and $\gamma(x, u_i, \nabla u_i)$ are Lipschitz continuous in $(x, u_i, \nabla u_i) \in \Omega \times \mathbb{R} \times \mathbb{R}^d$.
- In each subdomain, the derivatives of solutions from alternating Schwarz method (3), (4) converge to the derivative of solution u_i. Precisely, there exists a constant $\rho_1 \in (0, 1)$, such that for iteration times $\forall k \geq 0$

$$\|\nabla u_i^* - \nabla u_i^k\|_\infty \leq \rho_1^k \|\nabla u_i^* - \nabla u_i^0\|_\infty.$$

- $\alpha(x, u, \tau)$ is continuously differentiable w.r.t. (t, x).
- There is a positive constant $v > 0$ such that

$$\alpha(x, u, \tau)\tau \geq v|\tau|^2$$

and $\forall \tau_1, \tau_2 \in \mathbb{R}^d, \tau_1 \neq \tau_2$

$$\langle \alpha(x, u, \tau_1) - \alpha(x, u, \tau_2), \tau_1 - \tau_2 \rangle > 0.$$

Theorem 3: Suppose the domain Ω is decomposed into $\{\Omega_i\}_{i=1}^p$, $k > 0$ denotes iteration times (omitted in notations for brief). $\mathfrak{N}_i(\psi)$ denotes networks space in subdomain Ω_i, where subdomains are compact. Assume that target function (26) has unique solution in each subdomain, nonlinear terms $div(x, u, \nabla u)$ and $\gamma(x, u, \nabla u)$ are locally Lipschitz in $(u_i, \nabla u_i)$, and ∇u_i^k uniformly converges to ∇u_i^* with k. For $\forall \epsilon > 0$, there $\exists K > 0$ such that there exists a set of neural networks $\{\mathbf{N}_i \in \mathfrak{N}_i(\psi)\}_{i=1}^p$ satisfies the L^2 error $E_2(\mathbf{N}_i)$ as follow

$$\sum_{i=1}^p \lim_{k\to\infty} E_2(\mathbf{N}_i) \leq K\epsilon. \tag{30}$$

Proof: In each subdomain Ω_i, with iteration times $k > 0$, $E_2^k(\mathbf{N}_i)$ denotes the L^2 loss between $\mathbf{N}_i^k$ and u_i^*.

$$\begin{aligned}
\lim_{k\to\infty} E_2^k(\mathbf{N}_i) &= \| \text{div}(\alpha(x, u_i^*(x), \nabla u_i^*(x))) \\
&\quad + \gamma(x, u_i^*(x), \nabla u_i^*(x)) \\
&\quad - [\text{div}(\alpha(x, \mathbf{N}_i(x), \nabla\mathbf{N}_i(x))) \\
&\quad + \gamma(x, \mathbf{N}_i(x), \nabla\mathbf{N}_i(x))]\|_{\Omega_i}^2 \\
&\quad + \|\mathbf{N}_i\|_{\partial\Omega_i}^2. \tag{31}
\end{aligned}$$

With Lemma 1, it is clear that the sum of last term is smaller than $K_1\epsilon$, where $K_1 > 0$ is a constant. We assumed that ∇u_i^k uniformly converges to ∇u_i^* with k, and this means that

$$\|\nabla u_i^* - \nabla u_i^k\|_\infty \leq \rho_1^k \|\nabla u_i^* - \nabla u_i^0\|_\infty. \tag{32}$$

So that we have

$$\begin{aligned}
&\lim_{k\to\infty} E_2^k(\mathbf{N}_i) \\
&\leq \int_{\Omega_i} | \text{div}(\alpha(x, u_i^*(x), \nabla u_i^*(x))) - \text{div}(\alpha(x, u_i^k(x), \nabla u_i^k(x))) \\
&\quad + \text{div}(\alpha(x, u_i^k(x), \nabla u_i^k(x))) - \text{div}(\alpha(x, \mathbf{N}_i(x), \nabla\mathbf{N}_i(x)))| dx \\
&\quad + \int_{\Omega_i} |\gamma(x, u_i^*(x), \nabla u_i^*(x)) - \gamma(x, u_i^k(x), \nabla u_i^k(x))
\end{aligned}$$

$$+\gamma(x, u_i^k(x), \nabla u_i^k(x)) - \gamma(x, \mathbf{N}_i(x), \nabla \mathbf{N}_i(x))|dx + K_1\epsilon$$

$$\leq \int_{\Omega_i} |\operatorname{div}(\alpha(x, u_i^*(x), \nabla u_i^*(x))) - \operatorname{div}(\alpha(x, u_i^k(x), \nabla u_i^k(x)))|dx$$

$$+ \int_{\Omega_i} |\operatorname{div}(\alpha(x, u_i^k(x), \nabla u_i^k(x))) - \operatorname{div}(\alpha(x, \mathbf{N}_i(x), \nabla \mathbf{N}_i(x)))|dx$$

$$+ \int_{\Omega_i} |\gamma(x, u_i^*(x), \nabla u_i^*(x)) - \gamma(x, u_i^k(x), \nabla u_i^k(x))|dx$$

$$+ \int_{\Omega_i} |\gamma(x, u_i^k(x), \nabla u_i^k(x)) - \gamma(x, \mathbf{N}_i(x), \nabla \mathbf{N}_i(x))|dx + K_1\epsilon$$

$$\leq \int_{\Omega_i} |\operatorname{div}(\alpha(x, u_i^*(x), \nabla u_i^*(x))) - \operatorname{div}(\alpha(x, u_i^k(x), \nabla u_i^k(x)))|dx$$

$$+ \int_{\Omega_i} |\gamma(x, u_i^*(x), \nabla u_i^*(x)) - \gamma(x, u_i^k(x), \nabla u_i^k(x))|dx$$

$$+ K_2\epsilon + K_1\epsilon, \tag{33}$$

where $K_2 > 0$ is a constant, and the error bound between u_i^k and $\mathbf{N}_i^k$ is proved from Theorem 7.1 in [21].

$$\lim_{k\to\infty} \int_{\Omega_i} |\operatorname{div}(\alpha(x, u_i^*(x), \nabla u_i^*(x)))$$
$$- \operatorname{div}(\alpha(x, u_i^k(x), \nabla u_i^k(x)))|^2 dx$$

$$\leq \int_{\Omega_i} (|u_i^k|^{q_1} + |\nabla u_i^k|)|^{q_2} + |u_i^*|^{q_3} + |\nabla u_i^*|)|^{q_4})$$
$$\times (|u_i^* - u_i^k| + |\nabla u_i^* - \nabla u_i^k|)dx$$

$$\leq (\int_{\Omega_i} (|u_i^k - u_i^*|^{q_1} + |\nabla u_i^k - \nabla u_i^*|^{q_2} + |u_i^*|^{\max\{q_1, q_3\}}$$
$$+ |\nabla u_i^*|^{\max\{q_2, q_4\}})^r dx)^{1/r}$$

$$\times (\int_{\Omega_i} \rho^k \|u_i^* - u_i^0\|_\infty + \rho_1^k \|\nabla u_i^* - \nabla u_i^0\|_\infty dx)$$

$$\leq (\int_{\Omega_i} \|u_i^k - u_i^*\|_\infty^{q_1} + \|\nabla u_i^k - \nabla u_i^*\|_\infty^{q_2} + \sup_{\Omega_i} |u_i^*|^{\max\{q_1, q_3\}}$$
$$+ \sup_{\Omega_i} |\nabla u_i^*|^{\max\{q_2, q_4\}} dx)\epsilon$$

$$\leq K_3\epsilon, \tag{34}$$

where $K_3 > 0$ is a constant depends on ϵ. While $\gamma(\cdot)$ is also Lipschitz continuous, we can prove the upper bound of $\int_{\Omega_i} |\gamma(x, u_i^*(x), \nabla u_i^*(x)) - \gamma(x, u_i^k(x), \nabla u_i^k(x))|dx$ in the same formula with Equation (34), denoted as $K_4\epsilon$. Hence we can obtain

$$\sum_{i=1}^p \lim_{k\to\infty} E_2(\mathbf{N}_i) \leq \sum_{i=1}^p (K_1\epsilon + K_2\epsilon + K_3\epsilon + K_4\epsilon)$$
$$\leq K\epsilon. \tag{35}$$

$\square$

Theorem 4: Under Assumption 2 and Equation (29), *with iteration times* $k \to \infty$, *the set of neural networks* $\mathbf{N}_i$ *converge to the unique solutions to* (26), *strongly in* $\mathcal{L}^\rho(\Omega_i)$ *for every* $\rho < 2$. *In addition, in each subdomain the sequence* $\{\mathbf{N}_i^n(x)\}_{n\in\mathbb{N}}$ *is bounded in n under the constraint of* Proposition 2 *and converges to* u_i.

Proof: In each subdomain, the convergence can be obtained from the Theorems 7.1 and 7.3 in [21]. With the

Proposition 2, the sequence $\{\mathbf{N}_i^n\}_{n\in\mathbb{N}}$ is uniform bounded in n, and the rates of convergence to the solution u^* are related to overlapping areas. $\square$

Specially, the n in $\{\mathbf{N}_i^n(x)\}_{n\in\mathbb{N}}$ means training times instead of iteration times. We leave the proof for time-dependent data-free variational formulations for future work.

V. D3M SUMMARY

To summarize the strategies in Section III and IV, the full procedure of D3M comprises the following steps:

- pre-step: Set the architecture of neural networks in each subdomain;
- offline step: Construct functions for boundary conditions, train networks and generate local solutions;
- online step: Estimate target input data using neural networks. If solutions don't converge, transfer information on interfaces and go back to the offline step.

The pre-step is cheap, because the setting of neural networks is an easy task. In the offline stage, the complex system is fully decomposed into component parts, which means that there is no data exchange between subdomains. Since we only approximate the data on interface with normal simple approach such as Fourier series and polynomial chaos [36], [47], the approximation is also low costly. After decomposition, the requirement for number of samples for the Monte Carlo integration of the residual loss function (15) is significantly reduced, while the density of samples does not change. Since the number of samples decreasing and the domain becoming simpler, we can use neural networks with few layers to achieve a relatively high accuracy. If we keep the same number of samples and layers as the global setting, D3M should obtain a better accuracy. The cost of the online step is low, since no PDE solve is required. This full approach is also summarized in Figure 4, where transformations of data happen between adjacent subdomains.

For the problems we focus on (systems governed by PDEs), the cost of the D3M is dominated by the local training procedures in the offline step. Here we present a rough order of magnitude analysis of the computational costs where C_{solve} denotes the cost of one block in each training epoch (i.e., the cost of any block with the same number of neurons in each training iteration is taken to be equal for simplicity). The dominant cost of D3M is the total number of blocks of neural network and training times, $\sum_{i=1}^p N_i B_i T_i C_{\text{solve}}$, where N_i is sample size, B_i is number of blocks and T_i is training times. If we consider equal offline sample sizes, number of blocks and training epochs, $N_i = N_{off}, B_i = B_{off}, T_i = T_{off}$ for all subdomains $\{D_i\}_{i=1}^p$, then total cost can be written as $pN_{off} B_{off} T_{off} C_{\text{solve}}$. The total cost is decreased by employing the idea of hierarchical neural networks [48], [49].

VI. NUMERICAL TESTS

Here we consider two classical problems, the Poisson's equation and the time-independent Schrödinger equation, to verify the performance of our D3M. All timings conduct on an Intel
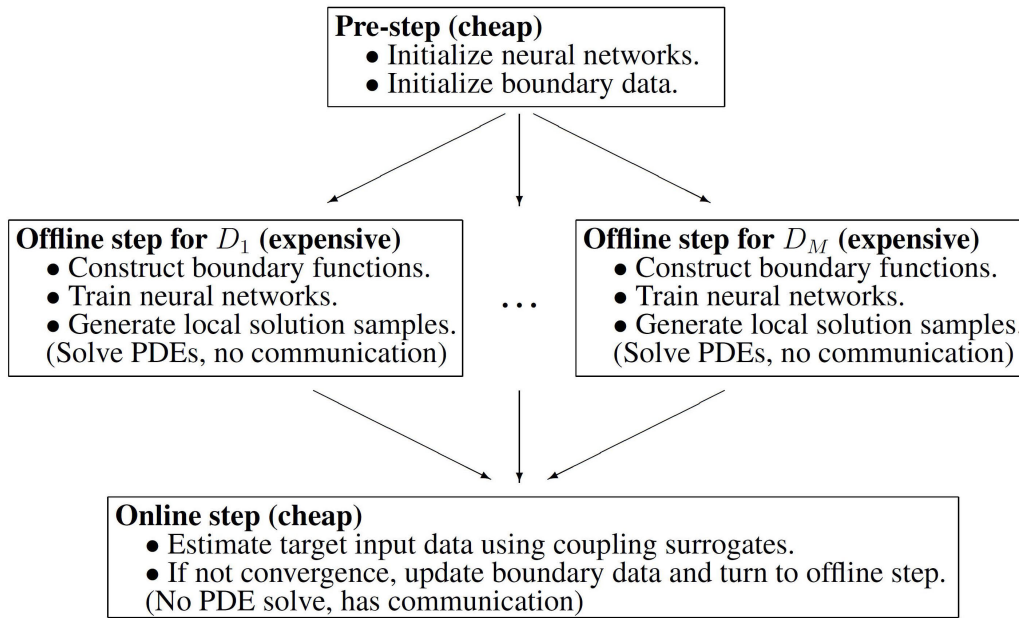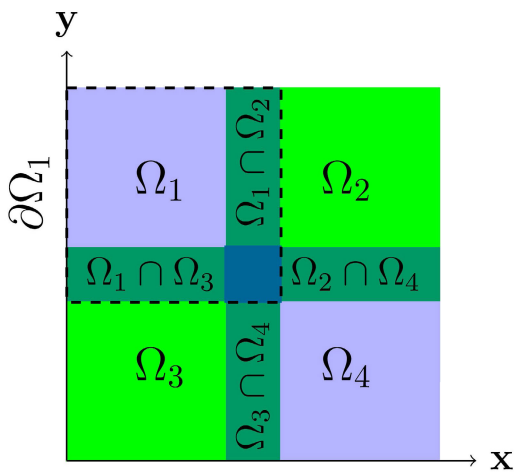
FIGURE 4. D3M summary.

FIGURE 5. Illustrations of the physical domain with four overlapping components.

Core i5-7500, 16GB RAM, Nvidia GTX 1080Ti processor with PyTorch 1.0.1 [43] under Python 3.6.5. We train the networks only 30 epoch using L-BFGS in numerical tests (cost within two minutes). The choices of the neuron and the layer numbers are empirical, which may not be optimal.

A. POISSON'S EQUATION

$$\begin{cases} -\Delta u(x, y) = 1, & \text{in } \Omega, \\ u(x, y) = 0, & \text{on } \partial\Omega, \end{cases} \tag{36}$$

where the physical domain is $\Omega = (-1, 1) \times (-1, 1)$. The domain decomposition setting is illustrated in Figure 5. To further improve the efficiency of D3M, we propose a new type of sampling methods. We randomly sample

in each subdomain, and the number of samples increases with iteration times increase. The sample size on interfaces remains the same to provide an accurate solution for data exchange. An illustration of our D3M sampling method is represented in Figure 6

Remark 2: According to the research on overfitting, the hypothesis set (w.r.t. the complexity of neural networks) should match the quantity and quality of data instead of target functions [50]. So in the initial several iterations, the number of residual blocks is small. The number increases while the sample size in Figure 6 increases.

After decomposition, with designed $\mathfrak{g}_i$ satisfying Definition 1 for each subdomains, the function $v_i := u_i - \mathfrak{g}_i$ satisfies the homogeneous Poisson's equation and follows (3) and (4).

The results of D3M is shown in Figure 7(a). For comparison, we plot the result of normal deep Ritz method (DRM) with the same type of network and the finite element method (FEM) in Figure 7(b) and (c). We set the result of FEM (degrees of freedom $N_h = 1089$) as the groundtruth and define the relative error $e_r = \frac{\|sol - fem_{sol}\|_2}{\|fem_{sol}\|_2}$. We compare results using residual network (ResNet), and the comparison including relative errors are shown in Table 1. We can see that with the same setting for networks, our D3M offers a higher accuracy than normal DRM in this experiment. With the number of residual block increasing, the parameter of network is increasing so that the performance of approximation becomes better.

B. SCHRÖDINGER EQUATION

In the area of domain decomposition methods, the steady-state Schrödinger equation is one of the classical
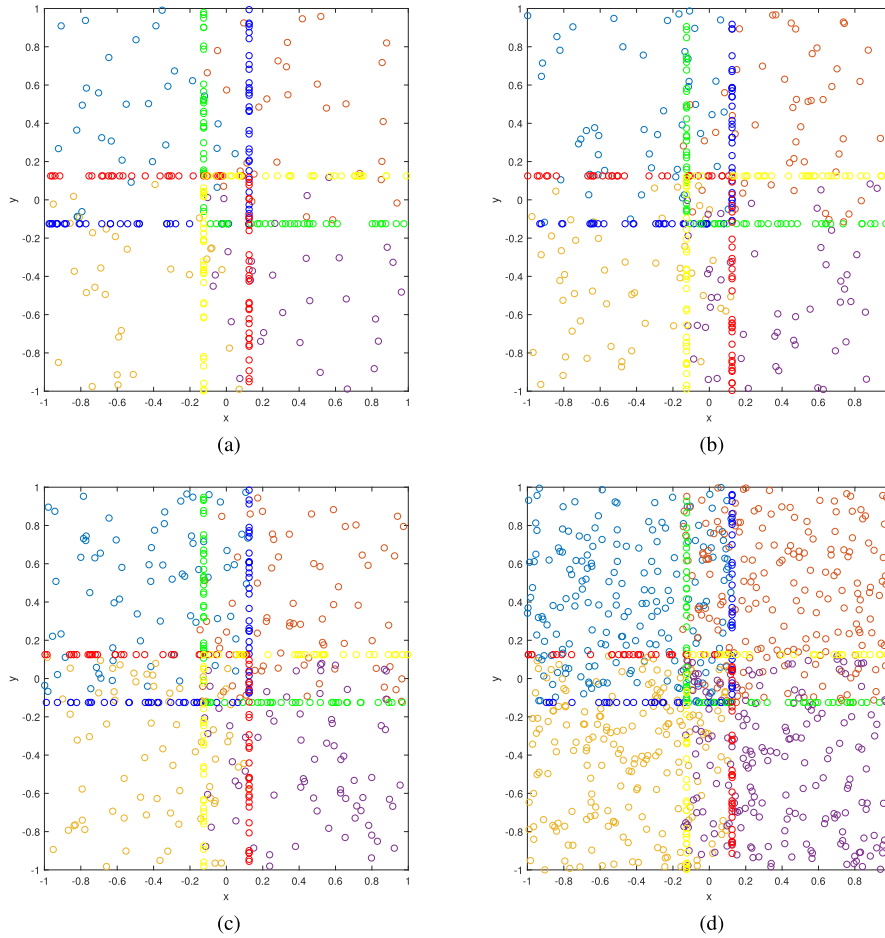
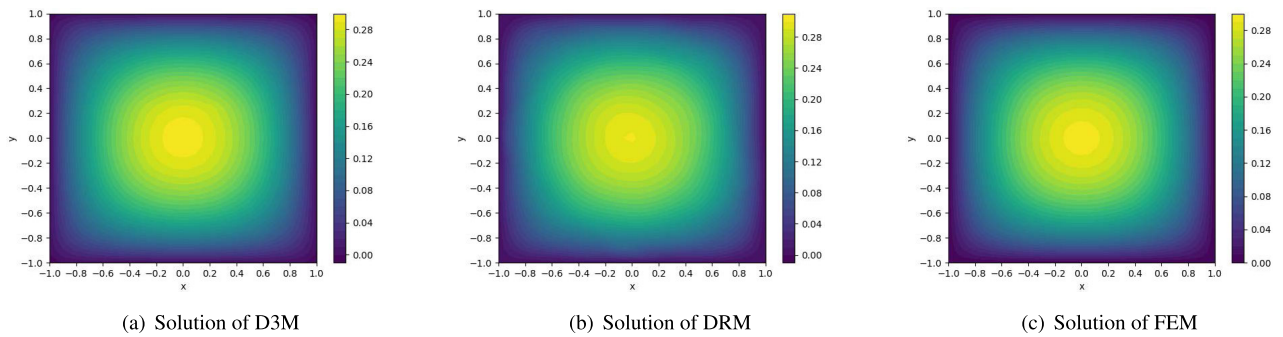**FIGURE 6.** D3M sampling: A new type of mesh-free sampling.

(a) Solution of D3M   (b) Solution of DRM   (c) Solution of FEM

FIGURE 7. Solutions computed by three different methods.

TABLE 1. The relative error for Poisson's equation.

Method	Net type	Blocks	Number of neurons	Relative error
DRM	ResNet	4	2048	0.0271
DRM	ResNet	8	4096	0.0157
D3M	ResNet	4	2048	0.0065
D3M	ResNet	8	4096	0.0045

problems [51], [52].

$$\left[\frac{-\bar{h}^2}{2\,m}\nabla^2 + V(\mathbf{r})\right]\Psi(\mathbf{r}) = E\Psi(\mathbf{r}). \qquad (37)$$

where $\bar{h} = \frac{h}{2\pi}$ is the reduced Planck constant, m is the particle's mass and E is a known constant related to the energy level. This equation occurs often in quantum mechanics where $V(\mathbf{r})$ is the function for potential energy. Here we consider an infinite potential well

$$V(\mathbf{r}) = \begin{cases} 0, & \mathbf{r} \in [0, 1]^d, \\ \infty, & \mathbf{r} \notin [0, 1]^d. \end{cases} \qquad (38)$$
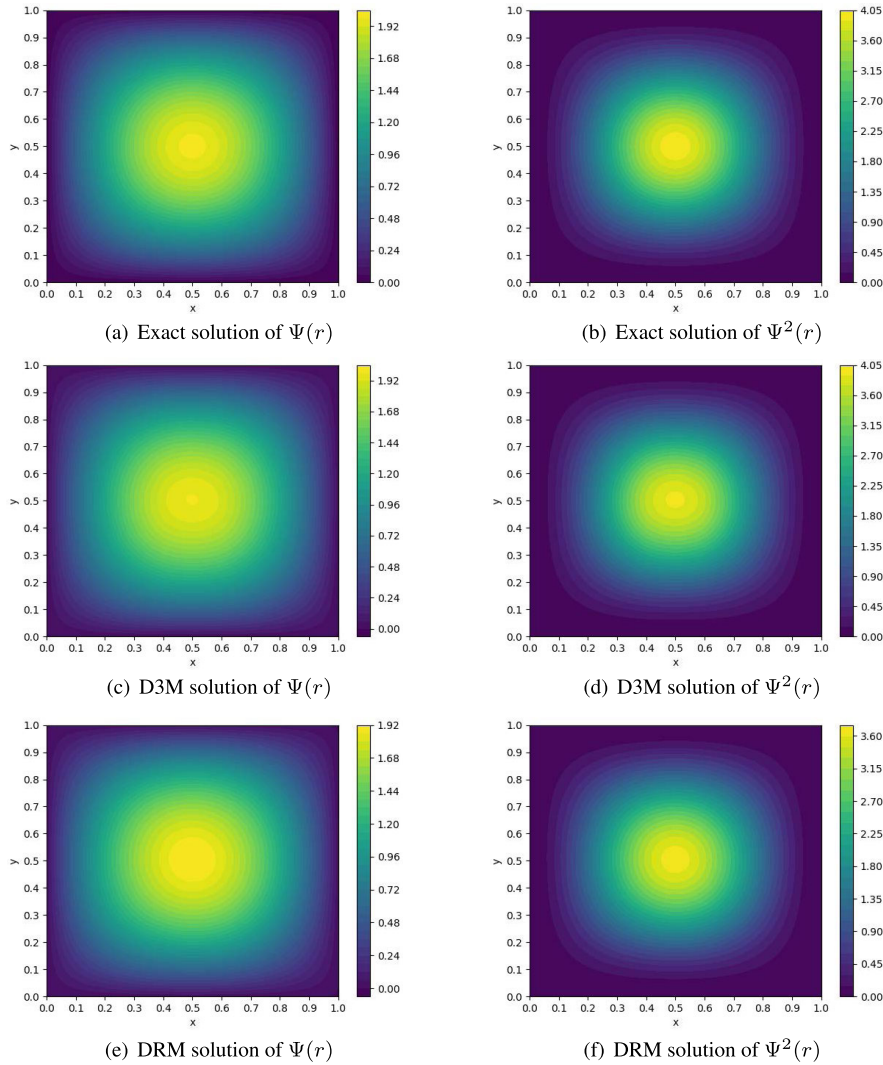
FIGURE 8. The solutions of wave function and probability density for a time-independent Schrödinger equation.

The variational loss is

$$L(\tau_i, \mathbf{N}_i, q) = \int_{\Omega_i} [\Delta \mathbf{N}_i(r)) + \Delta \mathfrak{g}_i - E \mathbf{N}_i(r)) + E \mathfrak{g}_i] dr$$
$$+ q \int_{\partial \Omega_i} (\mathbf{N}_i(r))^2 dr + \gamma (\int_{\Omega_i} |\mathbf{N}_i(r)|^2 dr - 1 + P_i)^2,$$
(39)

where $\int_{\Omega} |\Psi(r)|^2 dr = 1$, because $\Psi(r)$ is the wave function and $\Psi(r)^2$ means the probability density of particle appearing. P_i denotes the probability of particle appearing in $\Omega \backslash \Omega_i$. It should be noted that $\mathbf{N}_i$ is an approximation of $\Psi(\mathbf{r})_i - g_i$, where $\Psi(\mathbf{r})_i$ is the global solution $\Psi(\mathbf{r})$ restricted on the subdomain Ω_i and g_i is the boundary function for the interface of Ω_i.

For this two dimensional time-independent Schrödinger equation, we can calculate the analytical solution $\Psi(r) = A \sin(\frac{m\pi r_1}{b}) C \sin(\frac{n\pi r_2}{a})$, where $A = C = \sqrt{2}$, $a = b = 1$ and $n = m = 1$ in this infinite potential well case with domain

TABLE 2. Relative errors for the wave function and the probability density.

Target	Method	Net type	Blocks	Number of neurons	Relative error
Wave	DRM	ResNet	4	2048	0.0209
Wave	DRM	ResNet	8	4096	0.0169
Wave	D3M	ResNet	4	2048	0.0095
Wave	D3M	ResNet	8	4096	0.0045
Prob.	DRM	ResNet	4	2048	0.0357
Prob.	DRM	ResNet	8	4096	0.0334
Prob.	D3M	ResNet	4	2048	0.0143
Prob.	D3M	ResNet	8	4096	0.0091

$[0, 1] \times [0, 1]$. As shown in Figure 8, excellent agreement can be achieved between the exact solutions and predictions from our D3M. Compared with the solutions of DRM, D3M shows a better performance especially in peak values. The comparison of accuracy for both the wave equation and the probability density is shown in Table 2 and Figure 9. Under the same conditions including network structure, Lagrangian multiplier, learning rate, number of samples and training
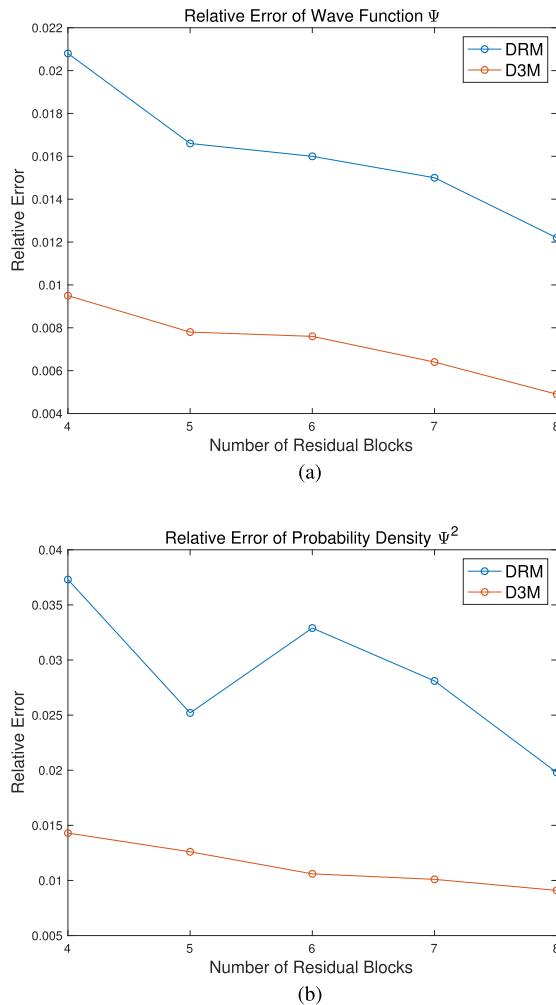
Relative Error of Wave Function Ψ

Relative Error of Probability Density Ψ²

FIGURE 9. Comparison of relative errors corresponding to different number of residual blocks.

epoch, our D3M shows smaller errors in both wave equation and probability density.

VII. CONCLUSION

This paper proposes a new deep domain decomposition method (D3M). The most significant contribution of the proposed approach is parallel computation, which lays a foundation for employing physics-constrained deep learning framework in large-scale engineering simulations or designs. This is accomplished by incorporating domain decomposition method into the loss function. And our framework absorbs the idea of mixed finite element method, so that the boundary condition can be satisfied more accurately. We have demonstrated that the deep domain decomposition method can solve general parabolic PDEs with high accuracy.

In theory, our approach is feasible to solve complex systems with many subdomains and corresponding neural networks. In practice, the approach suffers from three main bottlenecks. One is that a bad initialization of neural networks can lead to superfluous cost for following iterations. Second is the choice of functions for approximating interfaces.

And another is that the choice of Lagrangian multiplier is important but lacks of prior. We leave these questions for future work.

REFERENCES

[1] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics.* New York, NY, USA: Oxford Univ. Press, 2014.

[2] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.

[3] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Proc. Int. Conf. Artif. Neural Netw.* Springer, 1997, pp. 583–588.

[4] G. Berkooz, P. Holmes, and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," *Annu. Rev. Fluid Mech.*, vol. 25, no. 1, pp. 539–575, Jan. 1993.

[5] K. Willcox and J. Peraire, "Balanced model reduction via the proper orthogonal decomposition," *AIAA J.*, vol. 40, no. 11, pp. 2323–2330, 2002.

[6] K. Veroy, D. Rovas, and A. Patera, "A posteriori error estimation for reduced-basis approximation of parametrized elliptic coercive partial differential equations: 'Convex inverse' bound conditioners," *ESAIM: Control, Optim. Calculus Variat.*, vol. 8, pp. 1007–1028, Aug. 2002.

[7] A. Quarteroni and G. Rozza, "Numerical solution of parametrized Navier–Stokes equations by reduced basis methods," *Numer. Methods Partial Differ. Equ.*, vol. 23, no. 4, pp. 923–948, 2007.

[8] H. C. Elman and Q. Liao, "Reduced basis collocation methods for partial differential equations with random coefficients," *SIAM/ASA J. Uncertainty Quantification*, vol. 1, no. 1, pp. 192–217, Jan. 2013.

[9] P. Chen, A. Quarteroni, and G. Rozza, "Comparison between reduced basis and stochastic collocation methods for elliptic problems," *J. Sci. Comput.*, vol. 59, no. 1, pp. 187–216, Apr. 2014.

[10] Q. Liao and G. Lin, "Reduced basis ANOVA methods for partial differential equations with high-dimensional random inputs," *J. Comput. Phys.*, vol. 317, pp. 148–164, Jul. 2016.

[11] J. Jiang, Y. Chen, and A. Narayan, "A goal–oriented reduced basis methods–accelerated generalized polynomial chaos algorithm," *SIAM/ASA J. Uncertainty Quantification*, vol. 4, no. 1, pp. 1398–1420, Jan. 2016.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[13] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[14] E. Weinan, "A proposal on machine learning via dynamical systems," *Commun. Math. Statist.*, vol. 5, no. 1, pp. 1–11, Mar. 2017.

[15] W. E and B. Yu, "The deep Ritz method: A deep learning–based numerical algorithm for solving variational problems," *Commun. Math. Stat.*, vol. 6, no. 1, pp. 1–12, Mar. 2018.

[16] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (Part I): Data-driven solutions of nonlinear partial differential equations," 2017, *arXiv:1711.10561*. [Online]. Available: https://arxiv.org/abs/1711.10561

[17] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (Part II): Data-driven discovery of nonlinear partial differential equations," 2017, *arXiv:1711.10566*. [Online]. Available: https://arxiv.org/abs/1711.10566

[18] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019.

[19] F. Song, G. Pange, C. Meneveau, and G. Karniadakis, "Fractional physical-inform neural networks (fPINNs) for turbulent flows," *Bull. Amer. Phys. Soc.*, to be published.

[20] Z. Long, Y. Lu, X. Ma, and B. Dong, "PDE-Net: Learning PDEs from data," 2017, *arXiv:1710.09668*. [Online]. Available: https://arxiv.org/abs/1710.09668

[21] J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," *J. Comput. Phys.*, vol. 375, pp. 1339–1364, Dec. 2018.

[22] Y. Zhu and N. Zabaras, "Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification," *J. Comput. Phys.*, vol. 366, pp. 415–447, Aug. 2018.

[23] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data," *J. Comput. Phys.*, vol. 394, pp. 56–81, Oct. 2019.

[24] K. Wu and D. Xiu, "Numerical aspects for approximating governing equations using data," *J. Comput. Phys.*, vol. 384, pp. 200–221, May 2019.

[25] K. Wu, T. Qin, and D. Xiu, "Structure-preserving method for reconstructing unknown Hamiltonian systems from trajectory data," 2019, *arXiv:1905.10396*. [Online]. Available: https://arxiv.org/abs/1905.10396

[26] S. Zhang and G. Lin, "Robust data-driven discovery of governing physical laws with error bars," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 474, no. 2217, 2018, Art. no. 20180305.

[27] H. A. Schwarz, *Ueber Einen Grenzübergang Durch Alternirendes Verfahren* (Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich). Zürich, Switzerland: Zürcher u. Furrer, 1870.

[28] S. L. Sobolev, "The Schwarz algorithm in the theory of elasticity," *Dokl. Acad. Nauk USSR*, vol. 2, pp. 235–238, 1936.

[29] S. G. Mikhlin, "On the Schwarz algorithm," *Dokl. Akad. Nauk SSSR*, vol. 77, no. 4, pp. 569–571, 1951.

[30] P.-L. Lions, "On the Schwarz alternating method. I," in *Proc. 1st Int. Symp. Domain Decomposition Methods Partial Differ. Equ.*, vol. 1. Paris, France, 1988, p. 42.

[31] P.-L. Lions, "On the Schwarz alternating method. II," in *Proc. 2nd Int. Symp. Domain Decomposition Methods Partial Differ. Equ.*, vol. 4. Philadelphia, PA, USA: SIAM, 1989, pp. 47–70.

[32] H. Antil, M. Heinkenschloss, R. H. W. Hoppe, and D. C. Sorensen, "Domain decomposition and model reduction for the numerical solution of PDE constrained optimization problems with localized optimization variables," *Comput. Vis. Sci.*, vol. 13, no. 6, pp. 249–264, Aug. 2010.

[33] Z.-J. Liao, R. Chen, Z. Yan, and X.-C. Cai, "A parallel implicit domain decomposition algorithm for the large eddy simulation of incompressible turbulent flows on 3D unstructured meshes," *Int. J. Numer. Methods Fluids*, vol. 89, no. 9, pp. 343–361, Mar. 2019.

[34] F. Kong, V. Kheyfets, E. Finol, and X.-C. Cai, "An efficient parallel simulation of unsteady blood flows in patient-specific pulmonary artery," *Int. J. Numer. Methods Biomed. Eng.*, vol. 34, no. 4, p. e2952, Apr. 2018.

[35] Q. Liao and K. Willcox, "A domain decomposition approach for uncertainty analysis," *SIAM J. Sci. Comput.*, vol. 37, no. 1, pp. A103–A133, Jan. 2015.

[36] Y. Chen, J. Jakeman, C. Gittelson, and D. Xiu, "Local polynomial chaos expansion for linear differential equations with high dimensional random inputs," *SIAM J. Sci. Comput.*, vol. 37, no. 1, pp. A79–A102, Jan. 2015.

[37] S. Li, X. Shao, and X.-C. Cai, "Multilevel space–time additive Schwarz methods for parabolic equations," *SIAM J. Sci. Comput.*, vol. 40, no. 5, pp. A3012–A3037, Jan. 2018.

[38] S. Zampini and X. Tu, "Multilevel balancing domain decomposition by constraints deluxe algorithms with adaptive coarse spaces for flow in porous media," *SIAM J. Sci. Comput.*, vol. 39, no. 4, pp. A1389–A1415, Jan. 2017.

[39] D. Zhang, L. Yang, and G. E. Karniadakis, "Bi-directional coupling between a PDE-domain and an adjacent Data-domain equipped with multi-fidelity sensors," *J. Comput. Phys.*, vol. 374, pp. 121–134, Dec. 2018.

[40] D. N. Arnold, "Mixed finite element methods for elliptic problems," *Comput. Methods Appl. Mech. Eng.*, vol. 82, nos. 1–3, pp. 281–300, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/004578259090168L

[41] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[42] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013, vol. 30, no. 1, p. 3.

[43] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. (2017). *Automatic Differentiation in Pytorch*. [Online]. Available: https://openreview.net/pdf?id=BJJsrmfCZ

[44] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, nos. 1–3, pp. 503–528, Aug. 1989.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[46] L. Kantorovich and V. Krylov, "Approximate methods of higher analysis," *Bull. Amer. Math. Soc.*, vol. 66, no. 3, pp. 146–147, 1960.

[47] M. Arnst, R. Ghanem, E. Phipps, and J. Red-Horse, "Dimension reduction in stochastic modeling of coupled problems," *Int. J. Numer. Methods Eng.*, vol. 92, no. 11, pp. 940–968, Dec. 2012.

[48] L. W. T. Ng and K. E. Willcox, "Multifidelity approaches for optimization under uncertainty," *Int. J. Numer. Meth. Eng.*, vol. 100, no. 10, pp. 746–772, Dec. 2014.

[49] K. Li, K. Tang, J. Li, T. Wu, and Q. Liao, "A hierarchical neural hybrid method for failure probability estimation," *IEEE Access*, vol. 7, pp. 112087–112096, 2019.

[50] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*, vol. 4. New York, NY, USA: AMLBook, 2012.

[51] A. Toselli and O. Widlund, *Domain Decomposition Methods-Algorithms and Theory*. Berlin, Germany: Springer-Verlag, 2006, vol. 34.

[52] T. Hagstrom, R. Tewarson, and A. Jazcilevich, "Numerical experiments on a domain decomposition algorithm for nonlinear elliptic boundary value problems," *Appl. Math. Lett.*, vol. 1, no. 3, pp. 299–302, 1988.
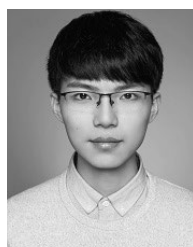
KE LI received the B.S. degree in information and computational science from Chongqing University. He is currently pursuing the master's degree with the School of Information Science and Technology, ShanghaiTech University, the Shanghai Institute of Microsystem and Information Technology, the Chinese Academy of Sciences, and the University of Chinese Academy of Sciences. His research interests include uncertainty quantification, domain decomposition, and deep learning.

KEJUN TANG received the B.S. degree in computational mathematics from Yantai University. He is currently pursuing the Ph.D. degree with the School of Information Science and Technology, ShanghaiTech University, the Shanghai Institute of Microsystem and Information Technology, the Chinese Academy of Sciences, and the University of Chinese Academy of Sciences. His research interests include tensor methods, machine learning, and scientific computing.
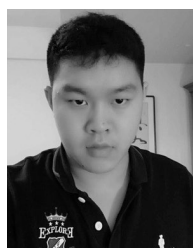
TIANFAN WU received the B.S. degree in mathematics and statistics from Miami University, Oxford. He is currently pursuing the master's degree with the Viterbi School of Engineering, University of Southern California. His research interests include data science and computational statistics.
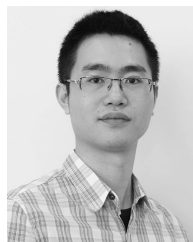
QIFENG LIAO received the Ph.D. degree in applied numerical computing from the School of Mathematics, The University of Manchester, in December 2010. From January 2011 to June 2012, he held a postdoctoral position with the Department of Computer Science, University of Maryland, College Park. From July 2012 to February 2015, he held a postdoctoral position with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology. He is currently an Assistant Professor with the School of Information Science and Technology, ShanghaiTech University. His work was supported by the National Natural Science of China and Shanghai Young East Scholar. His research interest includes efficient numerical methods for PDEs with high-dimensional random inputs.

● ● ●