16th World Congress in Computational Mechanics (WCCM)
21-26 July 2024, Vancouver, Canada
A. Korobenko, M. Laforest, S. Prudhomme, R. Vaziri (Eds)

A NONOVERLAPPING DOMAIN DECOMPOSITION METHOD FOR EXTREME LEARNING MACHINES SOLVING ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

CHANG-OCK LEE[1*], YOUNGKYU LEE[2] AND BYUNGEUN RYOO[1]

[1] Department of Mathematical Sciences, KAIST
Daejeon 34141, Korea
*colee@kaist.edu

[2] Division of Applied Mathematics, Brown University
Providence, RI 02912, USA

Key words: Extreme learning machine, Nonoverlapping domain decomposition method, Elliptic problem, Parallel computation, Weight initialization.

Abstract. *Extreme learning machine (ELM) is a methodology for solving partial differential equations (PDEs) using a single hidden layer feed-forward neural network. It presets the weight/bias coefficients in the hidden layer with random values, which remain fixed throughout the computation, and uses a linear least squares method for training the parameters of the output layer of the neural network. It is known to be much faster than Physics Informed Neural Networks. However, classical ELM is still computationally expensive when a high level of representation is desired in the solution as this requires solving a large least squares system. In this paper, we propose a nonoverlapping domain decomposition method (DDM) for ELMs that not only reduces the training time of ELMs, but is also suitable for parallel computation. In numerical analysis, DDMs have been widely studied to reduce the time to obtain finite element solutions for elliptic PDEs through parallel computation. Among these approaches, nonoverlapping DDMs are attracting the most attention. Motivated by these methods, we introduce local neural networks, which are valid only at corresponding subdomains, and an auxiliary variable at the interface. We construct a system on the variable and the parameters of local neural networks. A Schur complement system on the interface can be derived by eliminating the parameters of the output layer. The auxiliary variable is then directly obtained by solving the reduced system after which the parameters for each local neural network are solved in parallel. An initialization method suitable for high approximation quality in large systems is also proposed. Numerical results that verify the acceleration performance of the proposed method with respect to the number of subdomains are presented.*

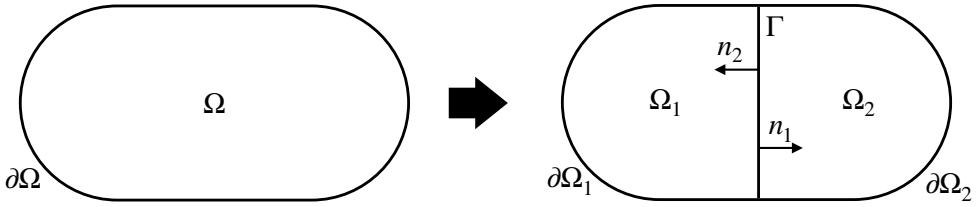


Figure 1: The domain Ω is decomposed into two nonoverlapping subdomains Ω_1 and Ω_2 with the interface Γ. Note that n_1 and n_2 are the outward normal vectors of Ω_1 and Ω_2, respectively, on Γ.

1 INTRODUCTION

Since neural networks can approximate any continuous function according to the Universal Approximation Theorem [1, 18], research on applying them to solve partial differential equations (PDEs) is attracting attention. The most popular method is Physics Informed Neural Networks (PINNs) [17]. A different method, so called extreme learning machines (ELMs) [6, 10, 11] solve the collocation system for the parameters of the output layer by a least squares computation, not by back-propagation of gradients. In this method, the weight/bias coefficients in all the hidden layers of the neural network are preset to random values and fixed throughout computation.

The domain decomposition method (DDM) is a fast solver for PDEs that is suitable for parallel computing of large-scale problems and has been successfully developed for elliptic PDEs [4, 19]. There are two types of DDMs: overlapping and nonoverlapping methods. In the latter, the domain is divided into nonoverlapping subdomains with interfaces. See Figure 1 for a graphical description. Typical nonoverlapping methods are the substructuring methods [2, 7].

Recently, domain decomposition approaches for ELM have been proposed, so called Local ELM (LocELM) [5] and Distributed Physics Informed ELM (DPIELM) [6]. In these methods, C^k continuity conditions are imposed on the subdomain interfaces and the solution on each subdomain is produced by a local feed-forward neural network. However, the parameters of the output layer are not optimized in parallel for each subdomain.

Motivated by substructuring methods of the nonoverlapping DDM, we propose a new nonoverlapping DDM for ELM. In the proposed method, a local neural network is introduced for each subdomain, which is valid only in the subdomain, along with auxiliary variable u_Γ, which represents the interface values of the solution. We construct a system for u_Γ and the parameters of the output layers of local neural networks. A Schur complement system is obtained by eliminating the parameters of the local neural networks. After solving the reduced system for u_Γ, the parameters of the local networks are solved in the least squares sense via parallel computation.

It is known that the performance of ELMs depends greatly on the distribution used to sample the hidden layer parameters [5]. A trial and error process to determine a suitable distribution would be expensive for large scale problems. To address this issue, we propose an initialization scheme for one hidden layer networks that scales well with the number of neurons.

2 PRELIMINARIES

We consider a general PDE with boundary conditions:

$$\begin{cases} \mathcal{L}u = f & \text{in } \Omega, \\ \mathcal{B}u = g & \text{on } \partial\Omega, \end{cases} \tag{2.1}$$

where $\Omega \subset \mathbb{R}^d$ is a bounded domain, $\partial\Omega$ denotes the boundary of Ω, f, g are functions, $\mathcal{L}$ is a differential operator defined for a function u, and $\mathcal{B}$ represents boundary conditions, e.g. $\mathcal{B}u = u = g$ gives Dirichlet boundary conditions. We assume that the problem (2.1) is well-posed so that the solution exists uniquely.

2.1 Nonoverlapping domain decomposition methods

DDM is a fast numerical solver for PDEs that can be implemented efficiently in parallel. This paper focuses on nonoverlapping DDM. Assume that the differential operator $\mathcal{L}$ is a second-order differential operator and that the domain Ω is divided into two nonoverlapping subdomains Ω_1 and Ω_2. Then, under suitable assumptions on f and boundaries of the subdomains, the solution of (2.1) is the same as the solution of the following system:

$$\begin{cases} \mathcal{L}u_1 = f & \text{in } \Omega_1, \\ u_1 = g & \text{on } \partial\Omega_1 \backslash \Gamma, \\ \mathcal{L}u_2 = f & \text{in } \Omega_2, \\ u_2 = g & \text{on } \partial\Omega_2 \backslash \Gamma, \\ u_1 = u_2 & \text{on } \Gamma, \\ \dfrac{\partial u_1}{\partial n_1} = -\dfrac{\partial u_2}{\partial n_2} & \text{on } \Gamma, \end{cases} \tag{2.2}$$

where Γ denotes the interface between Ω_1 and Ω_2. In addition, n_1 and n_2 are the outward normal to Ω_1 and Ω_2, respectively.

In substructuring methods, the interior unknowns are eliminated to obtain a Schur complement system defined on the interface. The interface unknowns are found by solving the reduced system using an iterative method. After finding the interface unknowns, the interior unknowns are found through parallel computation.

2.2 Universal approximation theorem for extreme learning machines

ELM was developed for single hidden layer feed-forward neural networks [10, 11]. It presets the weight/bias coefficients in all the hidden layers of the network with random values, which remain fixed throughout the computation, and uses a linear least squares method for training the weight coefficients in the output layer of the neural network. ELM is one example of the so-called randomized neural networks (see e.g. [12, 15]). The applications of ELM to function approximations and linear differential equations have been considered in several recent works [6, 13, 14] based on the following Universal Approximation Theorem.

Theorem 2.1 ([9, 12]). *Given any bounded non-constant piecewise continuous function $g\colon \mathbb{R} \to \mathbb{R}$, for any continuous target function f, there exists a sequence of single-hidden-layer feed-forward neural networks with g as the activation function, with its hidden layer coefficients randomly generated, and with its output layer coefficients appropriately chosen, such that $\lim_{n\to\infty} \|f - f_n\| = 0$, where n is the number of hidden-layer nodes and f_n is the output of the neural network.*

3 SCHUR COMPLEMENT SYSTEM FOR ELM

In this section, we present a new nonoverlapping DDM inspired by Schur complement systems in classical DDMs.

Let n be the number of neurons in the last layer, N be the number of subdomains, and the domain Ω be decomposed into nonoverlapping subdomains $\{\Omega_s\}$. For each subdomain Ω_s, we define a local neural network to produce $u_s(\mathbf{x})$ as

$$u_s(\mathbf{x}) = \sum_{j=1}^{n_N} c_j^s \phi_j^s(\mathbf{x}), \quad \forall \mathbf{x} \in \Omega_s,$$

where $n_N := n/N$, $\phi_j^s(\mathbf{x})$ is the j-th hidden layer output at $\mathbf{x}$, and θ^s is the parameter for $\{\phi_j^s\}$.

For simplicity, we assume $N = 2$ and $\mathcal{L}$ is a second-order differential operator. Then the solution of (2.1) is the same as the solution of (2.2).

Let $\{\mathbf{x}_i^s, \mathbf{x}_\Gamma^s, \mathbf{x}_b^s\}$ be the interior, interface, and boundary points, respectively, where $\mathbf{x}_i^s \in \mathbb{R}^{m_i^s \times d}$, $\mathbf{x}_\Gamma^s \in \mathbb{R}^{m_\Gamma^s \times d}$, and $\mathbf{x}_b^s \in \mathbb{R}^{m_b^s \times d}$. Note that m_i^s, m_Γ^s, and m_b^s denote the number of interior, interface, and boundary points, respectively. We introduce an auxiliary variable u_Γ satisfying

$$u_1 = u_2 = u_\Gamma \text{ on } \Gamma.$$

Then, we discretize (2.2) as follows:

$$\begin{bmatrix} K^1 & 0 & B^1 \\ 0 & K^2 & B^2 \\ A^1 & A^2 & 0 \end{bmatrix} \begin{bmatrix} c^1 \\ c^2 \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} F^1 \\ F^2 \\ 0 \end{bmatrix}, \tag{3.1}$$

where

$$K^s = \begin{bmatrix} \mathcal{L}\phi_1^s(\mathbf{x}_i^s) & \cdots & \mathcal{L}\phi_{n_N}^s(\mathbf{x}_i^s) \\ \phi_1^s(\mathbf{x}_b^s) & \cdots & \phi_{n_N}^s(\mathbf{x}_b^s) \\ \phi_1^s(\mathbf{x}_\Gamma^s) & \cdots & \phi_{n_N}^s(\mathbf{x}_\Gamma^s) \end{bmatrix}, \quad A^s = \begin{bmatrix} \frac{\partial \phi_1^s}{\partial n_s}(\mathbf{x}_\Gamma^s) & \cdots & \frac{\partial \phi_{n_N}^s}{\partial n_s}(\mathbf{x}_\Gamma^s) \end{bmatrix},$$

$$c^s = \begin{bmatrix} c_1^s \\ \vdots \\ c_{n_N}^s \end{bmatrix}, \quad B^s = \begin{bmatrix} 0 \\ 0 \\ -I \end{bmatrix}, \quad F^s = \begin{bmatrix} f(\mathbf{x}_i^s) \\ g(\mathbf{x}_b^s) \\ 0 \end{bmatrix}, \quad \text{for } s = 1, 2.$$

Remark 3.1. When there are more than two subdomains, $\frac{\partial \phi_j^s}{\partial n_s}(x_{\Gamma,j}^s)$ can become ambiguous when $x_{\Gamma,j}^s$ is a common corner point of the subdomains. In this case, we add a row to A^s for each edge to which $x_{\Gamma,j}^s$ belongs, where n_s is the normal direction to that edge.

Writing

$$K = \begin{bmatrix} K^1 & 0 \\ 0 & K^2 \end{bmatrix}, \quad A = \begin{bmatrix} A^1 & A^2 \end{bmatrix}, \quad B = \begin{bmatrix} B^1 \\ B^2 \end{bmatrix}, \quad F = \begin{bmatrix} F^1 \\ F^2 \end{bmatrix}, \quad \text{and } c = \begin{bmatrix} c^1 \\ c^2 \end{bmatrix},$$

we multiply

$$\begin{bmatrix} I & 0 \\ -AK^+ & I \end{bmatrix} \quad (K^+ = (K^T K)^{-1} K^T)$$

to (3.1), to obtain

$$\begin{bmatrix} K & B \\ 0 & -AK^+B \end{bmatrix} \begin{bmatrix} c \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} F \\ -AK^+F \end{bmatrix}. \tag{3.2}$$

Instead of solving the least squares system of (3.1), we solve the least squares system of (3.2). That is, (c, u_Γ) is the solution of

$$\begin{cases} K^T K c + K^T B u_\Gamma = K^T F, \\ B^T K c + B^T (I + K^{+T} A^T A K^+) B u_\Gamma = B^T (I + K^{+T} A^T A K^+) F. \end{cases} \tag{3.3}$$

By eliminating c from (3.3), we obtain the Schur complement system

$$B^T (I + K^{+T} A^T A K^+ - K K^+) B u_\Gamma = B^T (I + K^{+T} A^T A K^+ - K K^+) F. \tag{3.4}$$

Since the system (3.4) is SPD, we use the conjugate gradient method (CGM) [8] to solve the system. After finding u_Γ, in each subdomain Ω_s, we optimize the parameters of the output layer of each local neural network by solving ${K^s}^T K^s c^s = {K^s}^T (F^s - B^s u_\Gamma)$. The whole process is described in Algorithm 1.

Algorithm 1: Nonoverlapping DDM for ELM

Set θ^s randomly.
for $s = 1, \cdots, N$ *in parallel* **do**
  Construct K^s, A^s, B^s, and F^s
Solve $B^T (I + K^{+T} A^T A K^+ - K K^+) B u_\Gamma = B^T (I + K^{+T} A^T A K^+ - K K^+) F$
for $s = 1, \cdots, N$ *in parallel* **do**
  Solve ${K^s}^T K^s c^s = {K^s}^T (F^s - B^s u_\Gamma)$

4 NUMERICAL EXPERIMENTS

We present numerical results of the proposed nonoverlapping DDM for ELM. Experiments on various PDEs are conducted to demonstrate the performance of the proposed method.

A 1-hidden layer network with $n = 2^{16} = 65{,}536$ neurons with $\tanh$ activation is used. The number of neurons in each local neural network is n/N where N is the number of subdomains.

The total number of training points is $m = (2^8 + 1)^2 = 66{,}049$, which are distributed to each subdomain. CGM is used to solve the reduced system for u_Γ, stopping when the relative residual is smaller than 10^{-9}.

All algorithms are implemented using Pytorch [16] with mpi4py [3]. Except for the 16×16 case, all experiments were conducted on a single node equipped with two Intel Xeon Gold 6448H (2.4GHz 32c) processors, 256GB RAM, and each process was allocated $64/N$ cores. For the case of 16×16, two nodes with two Intel Xeon Gold 6448H and three nodes with two Intel Xeon Gold 6348 (2.6GHz 28c) with 256GB RAM each were used, and each process was allocated one core. The cluster is connected by a 100Gbps Infiniband network and runs on the CentOS 7 operating system.

4.1 Initialization of parameters of the hidden layers

In Algorithm 1, we solve the system (3.4) using CGM which requires matrix-vector multiplications involving $B^T(I + K^{+T}A^TAK^+ - KK^+)B$. When a 1-hidden layer network is applied to the local neural network, the output of the j-th neuron for a given $\mathbf{x}$ can be described as

$$\phi_j^s(\mathbf{x}) = \sigma\left(w_j^s \cdot \mathbf{x} + b_j^s\right) \text{ where } w_j^s \in \mathbb{R}^d \text{ and } b_j^s \in \mathbb{R}.$$

To satisfy the basic concept of ELMs, the initialization of $\{w_j^s\}$ and $\{b_j^s\}$ must meet two conditions. First, the span of $\{\phi_j^s\}_{j=1}^{n_N}$ should cover the solution space on each subdomain. Second, $\{\phi_j^s\}_{j=1}^{n_N}$ should be sufficiently independent. A certain way to achieve the first condition would be that for given $w \in \mathbb{R}^d$, $b \in \mathbb{R}$ and $\epsilon > 0$, the probability that there is some ϕ_j^s for which $\left\|w - w_j^s\right\| < \epsilon$ and $|b - b_j^s| < \epsilon$ tends to 1 as the number of neurons increases. However, in such a situation, many neuron outputs become redundant, violating the second condition. As a compromise, we require that the probability merely be a number close to 1 for some fixed ϵ.

Let $\bar{B}_r(\Omega_s) = \{\xi \mid \operatorname{dist}(\xi, \Omega_s) \le r(n_N)\}$. Suppose that for some functions $l, r \colon \mathbb{N} \to \mathbb{R}_+$ we choose w_j^s and b_j^s in the following way:

$$\begin{aligned} w_j^s &\sim U([-l(n_N), l(n_N)]^d), \\ b_j^s &= -w_j^s \cdot \xi_j^s \quad \text{for } \xi_j^s \sim U(\bar{B}_{r(n_N)}(\Omega_s)), \end{aligned}$$

where U denotes the uniform distribution.

Since the bias b_j^s determines the geometric position of ϕ_j^s, we assume that r is fixed. Then, the given initialization satisfies the conditions laid out above, if and only if $l = O(n_N^{1/d})$. To determine the proportionality constant, we first scale the subdomain Ω_s to the unit square. Combining $n_N = n/N$ with the fact that scaling performs a $N^{1/d}$ multiple, the initialization for 2D problems is given by

$$\begin{aligned} w_j^s &\sim U([-l, l]^2) \quad \text{for } l = c\sqrt{n}, \\ b_j^s &= -w_j^s \cdot \xi_j^s \quad \text{for } \xi_j^s \sim U(\bar{B}_r(\Omega_s)). \end{aligned} \tag{4.1}$$

The values of c and r are obtained empirically: we used $c = 1/8$ for the Poisson equation and $c = 1/16$ for the plate bending problem, respectively; we set $r = \operatorname{diam}(\Omega_s)/2$ for both problems.

Table 1: Relative L^2 and H^1 errors of the solutions of 2-D Poisson equation with the exact solution $u(x,y) = \sin(2\pi x)e^y$. N denotes the number of subdomains and the solutions of the case of $N = 1 \times 1$ is the ELM solution. For $N > 1$, the solutions are computed by Algorithm 1.

N	L^2 rel. error	H^1 rel. error	Wall-clock time (sec)
1×1	8.1550e-11	1.6095e-09	8617.13
2×2	1.0748e-10	1.0590e-09	315.83
4×4	1.6192e-08	3.2441e-08	19.46
8×8	2.6392e-08	8.1948e-08	3.48
16×16	1.0212e-07	1.5199e-07	10.43

Note that although r and c must be determined empirically, this can be done on a small scale problem before application to large scale problems.

4.2 Application to PDEs

We now look at the performance of the proposed non-overlapping DDM for ELM. All reported computation times for solving the model equations are averages of 10 runs. The finite element method (FEM) solution on a 4096×4096 grid is employed in cases where the exact solution is unknown. Note that the solution space is discretized by conforming P1 elements. To solve the system derived from FEM, we used CGM with stop condition 10^{-9} on the relative norm of the residual.

4.2.1 2-D Poisson equation

We first consider the following 2-D Poisson problem:

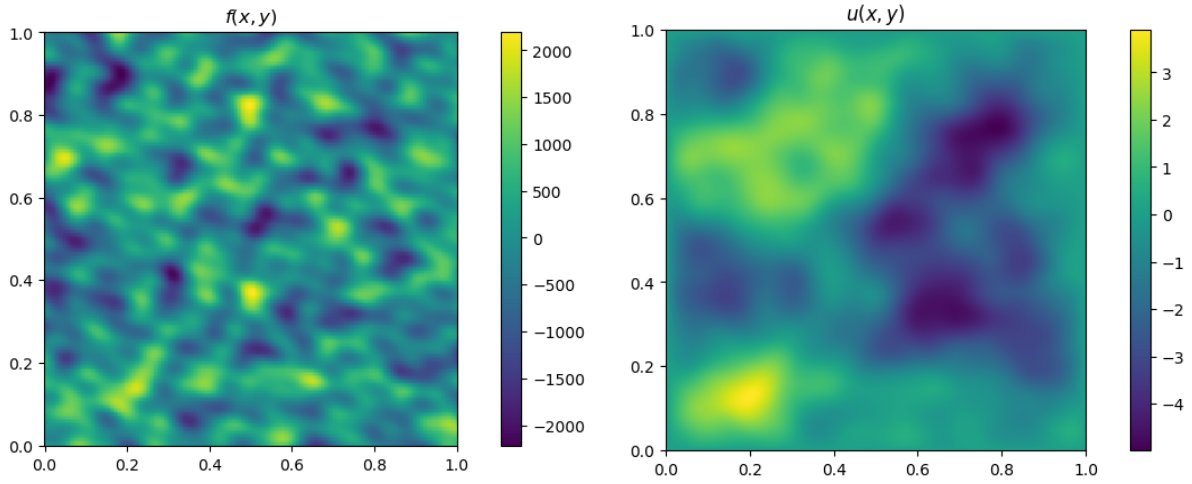
$$\begin{cases} -\Delta u = f & \text{in } \Omega = (0,1)^2, \\ u = g & \text{on } \partial\Omega \end{cases} \tag{4.2}$$

with the exact solution $u(x,y) = \sin(2\pi x)e^y$. Table 1 shows the relative L^2 and H^1 errors of the proposed DDM solution. Note that the result of $N = 1 \times 1$ denotes the ELM solution trained on the whole domain Ω. The relative errors of the solutions generated by Algorithm 1 are slightly larger than those of the ELM solution, but the wall-clock time is significantly smaller.

In addition, we test a more oscillatory case where $g \equiv 0$ and

$$\begin{gathered} f(\mathbf{x}) = \sum_{i=1}^{256} a_i \sin(w_i \cdot \mathbf{x} + b_i), \\ a_i \sim \mathcal{N}(0, \alpha^4/256), \quad w_i \sim \mathcal{N}(0, \alpha^2 I), \quad b_i \sim U((0, 2\pi)). \end{gathered} \tag{4.3}$$

In this formulation, f approximates a Gaussian random field of varying oscillation depending on α. In this case, since the exact solution is unknown, the FEM solution is used as a reference. A



(a) (b)

Figure 2: Graphical description of $f(x, y)$ and corresponding FEM solution $u(x, y)$ of 2-D Poisson equation; u is computed via FEM on a $4{,}096 \times 4{,}096$ mesh. (a) f for $\alpha = 32$ (b) Solution u for $\alpha = 32$

graphical depiction of the right hand sides and solutions for $\alpha = 32$ is shown in Figure 2. The results of Algorithm 1 are summarized in Table 2. Note that the wall-clock times decrease as the number of subdomains increases until the 16×16 case. At this point the interface problem becomes large enough to overtake the cost of solving a least squares problem on a subdomain.

4.2.2 2-D Poisson equation with a variable coefficient

Next, we test the proposed DDM with an equation with variable coefficient

$$\begin{cases} -\nabla \cdot (\rho \nabla u) = f & \text{in } \Omega = (0,1)^2, \\ u = g & \text{on } \partial\Omega, \end{cases} \tag{4.4}$$

where $\rho(\mathbf{x}) = \tanh(\mathrm{grf}(\mathbf{x})) + 1.1$, $f \equiv 1$, and $g \equiv 0$. Here, grf is an approximate Gaussian random field defined in (4.3). We again test the cases $\alpha = 32$. Note that ρ varies from 0.1 to 2.1. Since there is no closed form solution of (4.4), the FEM solution is used as the reference solution. Figure 3 gives a graphical depiction of ρ and the reference solution u.

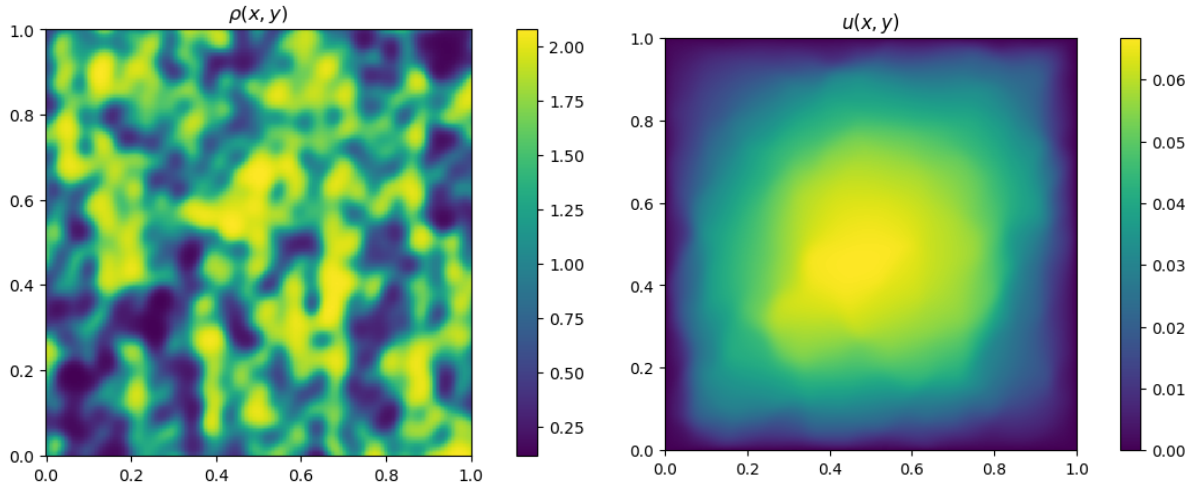
Table 3 shows the relative L^2 and H^1 errors. Best accuracy is achieved when $N = 16 \times 16$, suggesting the proposed DDM is more effective in handling oscillatory problems.

4.2.3 Plate Bending

Finally, we test a plate bending problem, assuming the Kirchhoff-Love plate bending theory for isotropic and homogeneous plates. For Young's modulus E, plate thickness H, and Poisson's

Table 2: Relative L^2 and H^1 errors of the solutions of 2-D Poisson equation in the more oscillatory case when $\alpha = 32$. The FEM solution on a $4{,}096 \times 4{,}096$ mesh is used as the reference solution. N denotes the number of subdomains and the case of $N = 1 \times 1$ is the ELM solution. For $N > 1$, the solutions are computed by Algorithm 1.

N	L^2 rel. error	H^1 rel. error	Wall-clock time (sec)
1×1	3.4758e-06	2.8890e-05	10992.84
2×2	2.7517e-06	2.2974e-05	314.09
4×4	2.4976e-06	1.9726e-05	19.54
8×8	2.4946e-06	2.1812e-05	3.28
16×16	2.2749e-06	1.9084e-05	8.11



(a) (b)

Figure 3: Graphical description of $\rho(x, y)$ and corresponding FEM solution $u(x, y)$ of 2-D Poisson equation with variable coefficient; u is computed via FEM on a $4{,}096 \times 4{,}096$ mesh. (a) ρ for $\alpha = 32$ (b) Solution u for $\alpha = 32$

Table 3: Relative L^2 and H^1 errors of the solutions of 2-D Poisson equation with variable coefficient when $\alpha = 32$. The FEM solution on a $4{,}096 \times 4{,}096$ mesh is used as the reference solution. N denotes the number of subdomains and the case of $N = 1 \times 1$ is the ELM solution. For $N > 1$, the solutions are computed by Algorithm 1.

N	L^2 rel. error	H^1 rel. error	Wall-clock time (sec)
1×1	7.6186e-03	3.3764e-02	11020.72
2×2	7.4366e-03	5.7018e-02	315.94
4×4	5.8671e-03	5.6910e-02	20.61
8×8	4.6123e-03	2.2875e-02	4.56
16×16	6.0091e-04	4.8583e-03	14.10

Table 4: Relative L^2 and H^1 errors of the solutions of a 2D plate bending problem. N denotes the number of subdomains and the case of $N = 1 \times 1$ is the ELM solution. For $N > 1$, the solutions are computed by Algorithm 1.

N	L^2 rel. error	H^1 rel. error	Wall-clock time (sec)
1×1	8.7574e-09	1.2857e-07	11105.85
2×2	1.1559e-08	1.6362e-07	322.29
4×4	6.9948e-09	1.1603e-07	21.93
8×8	3.6260e-08	7.3627e-07	3.74
16×16	4.7774e-07	5.5870e-06	5.55

ratio ν, the flexural rigidity is given by $D = EH^3/12(1-\nu^2)$. The equation for the simply supported plate is given by

$$\begin{cases} \Delta^2 u = \dfrac{q}{D} & \text{in } \Omega = (0,1)^2, \\ u = 0 & \text{on } \partial\Omega, \\ \Delta u = 0 & \text{on } \partial\Omega, \end{cases}$$

where q is the transversal load. The transmission conditions are continuity in u and Δu and their normal derivative across the interface. We solve the equation with $q = \sin(\pi x)\sin(\pi y)$, $E = 10^7$, $H = 10^{-3}$, and $\nu = 0.3$, and the exact solution is $u = \sin(\pi x)\sin(\pi y)/4\pi^4 D$. Table 4 presents the relative L^2 and H^1 errors of the results of Algorithm 1.

5 Conclusion

In this paper, we proposed a novel nonoverlapping domain decomposition method for ELM that is suitable for distributed memory computing. Motivated by substructuring methods in numerical analysis, a local neural network was defined in each nonoverlapping subdomain and

an auxiliary variable u_Γ at the interface was introduced. The coefficients of the last layer of local neural networks were eliminated using Schur complements, which gave a reduced system for u_Γ. After that, u_Γ was obtained by solving the system, and the coefficients of the last layer of local neural networks were obtained using the auxiliary variable u_Γ in parallel. We also presented an initialization technique when the local neural network is equipped with a shallow neural network. Numerical experiments demonstrated the practical efficacy of the proposed nonoverlapping method when large number of neurons are used. We expect that the proposed nonoverlapping DDM can be efficiently utilized for training ELMs with a large number of neurons through distributed parallel computation. As a future direction, it would be interesting to achieve scalability by either introducing a coarse problem or developing a preconditioner.

REFERENCES

[1] Barron, A.R., "Universal approximation bounds for superpositions of a sigmoidal function", *IEEE Transactions on Information Theory*, Vol. 39, 1993, pp. 930-945.

[2] Bramble, J.H. and Pasciak, J.E. and Schatz, A.H., "The Construction of Preconditioners for Elliptic Problems by Substructuring. I", *Mathematics of Computation*, Vol. 47, 1986, pp. 103-134.

[3] Dalcin, L. and Fang, Y.L.L., "mpi4py: Status Update After 12 Years of Development", *Computing in Science & Engineering*, Vol. 23, 2021, pp. 47-54.

[4] Dolean, V. and Jolivet, P. and Nataf, F., *An introduction to domain decomposition methods: algorithms, theory, and parallel implementation*, SIAM, Philadelphia, 2015.

[5] Dong, S. and Li, Z., "Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations", *Computer Methods in Applied Mechanics and Engineering*, Vol. 387, 2021, pp. 114129.

[6] Dwivedi, V. and Srinivasan, B., "Physics informed extreme learning machine (PIELM)–a rapid method for the numerical solution of partial differential equations", *Neurocomputing*, Vol. 391, 2020, pp. 96-118.

[7] Farhat, C. and Roux, F.X., "A method of finite element tearing and interconnecting and its parallel solution algorithm", *International Journal for Numerical methods in Engineering*, Vol. 32, 1991, pp. 1205-1227.

[8] Hestenes, M.R. and Stiefel, E., "Methods of Conjugate Gradients for Solving Linear Systems", *Journal of Research of the National Bureau of Standards*, Vol. 49, 1952, pp. 409-436.

[9] Huang, G.B. and Chen, L. and Siew, C.K., "Universal approximation using incremental constructive feedforward networks with random hidden nodes", *IEEE Transactions on Neural Networks*, Vol. 17, 2006, pp. 879-892.

[10] Huang, G.B. and Wang, D.H. and Lan, Y., “Extreme learning machines: a survey", *International Journal of Machine Learning and Cybernetics*, Vol. 2, 2011, pp. 107-122.

[11] Huang, G.B. and Zhu, Q.Y. and Siew, C.K., “Extreme learning machine: theory and applications", *Neurocomputing*, Vol. 70, 2006, pp. 489-501.

[12] Igelnik, B. and Pao, Y.H., “Stochastic choice of basis functions in adaptive function approximation and the functional-link net", *IEEE Transactions on Neural Networks*, Vol. 6, 1995, pp. 1320-1329.

[13] Liu, H. and Xing, B. and Wang, Z. and Li, L., “Legendre neural network method for several classes of singularly perturbed differential equations based on mapping and piecewise optimization technology", *Neural Processing Letters*, Vol. 51, 2020, pp. 2891-2913.

[14] Panghal, S. and Kumar, M., “Optimization free neural network approach for solving ordinary and partial differential equations", *Engineering with Computers*, Vol. 37, 2021, pp. 2989-3002.

[15] Pao, Y.H. and Park, G.H. and Sobajic, D.J., “Learning and generalization characteristics of the random vector functional-link net", *Neurocomputing*, Vol. 6, 1994, pp. 163-180.

[16] Paszke, A. and Gross, S. and Massa, F. and Lerer, A. and Bradbury, J. and Chanan, G. and Killeen, T. and Lin, Z. and Gimelshein, N. and Antiga, L. and Desmaison, A. and Kopf, A. and Yang, E. and DeVito, Z. and Raison, M. and Tejani, A. and Chilamkurthy, S. and Steiner, B. and Fang, L. and Bai, J. and Chintala, S., “PyTorch: An Imperative Style, High-Performance Deep Learning Library", *Advances in Neural Information Processing Systems*, 2019.

[17] Raissi, M. and Perdikaris, P. and Karniadakis, G.E., “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations", *Journal of Computational Physics*, Vol. 378, 2019, pp. 686-707.

[18] Siegel, J.W. and Xu, J., “Approximation rates for neural networks with general activation functions", *Neural Networks*, Vol. 128, 2020, pp. 313-321.

[19] Toselli, A. and Widlund, O., *Domain Decomposition Methods – Algorithms and Theory*, Springer, Berlin, 2005.