Domain Decomposition Algorithms for Neural Network Approximation of Partial Differential Equations

Hyea Hyun Kim and Hee Jun Yang

1 Introduction

With the success of deep learning technology in many application areas, there have been pioneering approaches to approximate solutions of partial differential equations by neural network functions [2, 10, 12, 13]. Such approaches have advantages over the classical approximation methods in that they can be used without generating meshes adaptive to problem domains or developing equation dependent numerical schemes. However, its accuracy, stability, and efficiency questions have not yet been fully answered. In addition, long training time makes the neural network solution very expensive.

To enhance the neural network solution accuracy, large or deep neural network functions are usually employed. When training parameters in such large or deep neural networks, the optimization error becomes problematic to pollute the resulting computed solution accuracy. To address this issue in the neural network approximation, we approximate the solution by using partitioned local neural network functions. For that we first form an iterative scheme based on domain decomposition methods and we then find local neural network functions that approximate the local problem solutions at each iteration. Contrary to the single large or deep neural network case, the local neural network parameter training can be done more efficiently with less optimization errors.

There have been previous studies that utilize domain decomposition algorithms [14] to enhance the neural network efficiency and accuracy. In [8, 9], alternating Schwarz algorithms were developed to second order elliptic problems and in the author's previous study [6], additive Schwarz algorithms were proposed to

Hyea Hyun Kim
Department of Applied Mathematics and Institute of Natural Sciences, Kyung Hee University, Korea, e-mail: hhkim@khu.ac.kr

Hee Jun Yang
Department of Mathematics, Kyung Hee University, Korea, e-mail: yhjj109@khu.ac.kr

the same model problems, where the neural network functions are formed based on overlapping subdomain partitions. In both approaches, the proposed methods showed promising results but concrete convergence study has not been fully considered. In [4, 5], partitioned neural network functions are formed based on a non-overlapping subdomain partition and the global cost function is formed to train the parameters in the partitioned neural network functions. In their approach, the communication cost between local neural networks becomes enormous, since the number of epochs in the parameter training easily becomes more than several tens of thousands in practice.

In the author's recent work [7], a concrete convergence analysis on one-level and two-level additive Schwarz algorithms was provided with an assumption on the approximation error in the local and coarse neural network solutions. The numerical results on the one-level method are consistent with the convergence analysis. However, those on the two-level methods show that the coarse problem does not help to accelerate the convergence and it even pollutes the solution accuracy. By the NTK (Neural Tangent Kernel) theory [3, 15], when training the parameters in the neural network approximation, the smooth part of solutions is well approximated and the residual loss for the differential equation is well trained than that for the boundary condition. The local neural network solution errors in our proposed method thus showed high contrast errors near the subdomain boundary that resulted a less smooth global error during the iteration. The proposed coarse problem in [7] was not suitable to correct such a non-smooth global error.

In this work, we propose a partitioned neural network by utilizing a partition of unity functions and we then apply the additive Schwarz algorithm to propose an iterative solution procedure on the partitioned neural network functions, where local neural network parameters are trained to approximate local problem solutions at each iteration. When training the local parameters, only the residual loss for the differential equation in each subdomain problem comes in the cost function, and the boundary condition is enforced directly by multiplying the partition of unity function as an ansatz to the local neural network function. With this idea, the optimization error can be reduced when training the local parameters at each iteration compared to the approaches in [7]. As reported in our previous work [7], the coarse problem in the two-level method did not work due to the high contrast optimization errors observed near the boundary of subdomain overlapping region. By utilizing the partition of unity functions in forming the partitioned neural network approximation, we can remove such error problems and the coarse problem in the two-level method is thus expected to work more effectively. Such a partitioned neural network function using ansatz was first proposed in [11] with the aim of obtaining a more accurate neural network approximation to highly oscillatory solutions.

This paper is organized as follows. In Section 2, we introduce neural network approximation methods for solving partial differential equations and in Section 3 we propose one-level and two-level additive Schwarz algorithms for the partitioned neural network functions, where we present the two methods in our previous work [7] and extend those methods to the partitioned neural network functions. In Section 4, numerical results are presented for model elliptic problems and conclusions are given.

2 Neural network approximation for partial differential equations

Among several neural network approaches to solutions of partial differential equations, we will consider the PINN (Physics Informed Neural Network) method by [12]. Our domain decomposition approach can be applied to other neural network approximation methods by [2, 10, 13] as well. In the PINN methods, the solution is approximated with a neural network function $U(x;\theta)$ and the parameters θ in the neural network function are trained to solve supervised learning tasks in order to satisfy any given laws of physics described by partial differential equations,

$$\mathcal{L}(u) = f, \quad \text{in } \Omega, \quad \mathcal{B}(u) = g, \quad \text{on } \partial\Omega, \tag{1}$$

where $\mathcal{L}$ denotes a differential operator defined for a function u and $\mathcal{B}$ describes a given boundary condition on u, and f, g are given functions.

We assume that the model problem in (1) is well-posed and the solution u exists. We then approximate the solution u in (1) by a neural network, $U(x;\theta)$, where the parameters θ are trained to minimize the cost function

$$\mathcal{J}(\theta) = \mathcal{J}_{X_\Omega}(\theta) + \mathcal{J}_{X_{\partial\Omega}}(\theta),$$

where

$$\begin{aligned}\mathcal{J}_{X_\Omega}(\theta) &:= \frac{1}{|X_\Omega|}\sum_{x\in X_\Omega} |\mathcal{L}(U(x;\theta)) - f(x)|^2,\\ \mathcal{J}_{X_{\partial\Omega}}(\theta) &:= \frac{1}{|X_{\partial\Omega}|}\sum_{x\in X_{\partial\Omega}} |\mathcal{B}(U(x;\theta)) - g(x)|^2.\end{aligned}$$

In the above, X_D denotes the collection of points chosen from the region D and $|X_D|$ denotes the number of points in the set X_D. The cost function $\mathcal{J}_{X_\Omega}(\theta)$ and $\mathcal{J}_{X_{\partial\Omega}}(\theta)$ are designed so that the optimized neural network $U(x;\theta)$ satisfies the equations in (1) derived from physics laws. When training the parameters θ, the following gradient based method is used,

$$\theta^{(n+1)} = \theta^{(n)} - \epsilon\nabla_\theta\mathcal{J}(\theta^{(n)})$$

for a given initial $\theta^{(0)}$ and with a suitable learning rate ϵ. Each gradient update step is called an epoch and usually more than several hundreds of thousand epochs are needed in such neural network approximation methods. Overall computation cost in the PINN is thus very expensive compared to the classical approximation methods.

The error between the exact solution $u(x)$ and the computed neural network solution $U(x;\widetilde{\theta})$ can be analyzed as follows. Letting $U(x;\theta^*)$ be the optimal approximate solution, we obtain

$$u(x) - U(x;\widetilde{\theta}) = (u(x) - U(x;\theta^*)) + (U(x;\theta^*) - U(x;\widetilde{\theta})),$$

where the first term in the right hand side is called the approximation error and the second is the optimization error. The approximation error can be controlled by enlarging the network size, while the optimization error is difficult to deal with. The optimization error depends on how to choose the training data set, how to form the loss functions, and how to perform the gradient based method.

In [11], it was numerically verified that for highly oscillatory model solutions PINN requires larger neural network functions and larger training epochs to increase the approximation solution accuracy. Such approximation property in the PINN was also analyzed by the NTK (Neural Tangent Kernel) theory, see [15]. To enhance the training efficiency and accuracy, in [11], the approximate solution is formed by using partitioned neural network functions with a much lesser number of parameters in each local neural network function than those in the single large neural network function. When a highly oscillatory solution is localized to a small subdomain, it becomes less oscillatory and thus it can be well approximated with a smaller neural network function. The parameter training cost in a smaller neural network function also becomes much smaller than that in the single larger neural network function. By utilizing the partitioned neural network functions, we thus expect that for difficult model problems we can reduce both the approximation error and the optimization error more effectively than just using a single large neural network function. In addition, utilizing the parallel computing resources, we can even make our one-level and two-level methods much more efficient than the single neural network case.

3 Additive Schwarz algorithms for neural network approximation

In this section, we first review the one-level and two-level additive Schwarz algorithms that were proposed in our previous work [7] and their convergence results under the approximation error assumption on each local and coarse neural network solutions. We then introduce a partitioned neural network function to approximate the solution and propose iterative methods on the partitioned neural network function to find the convergent iterates to the solution. The iteration methods can be analyzed as the same way in our previous study [7] to give the same convergence result. As we will see in the numerical results later, the partitioned neural network function gives less optimization errors and thus it gives faster convergence than in the previous work [7].

Our method is developed for the following model elliptic problem in a bounded domain Ω, i.e., to find u in the Hilbert space $H^1(\Omega)$ satisfying

$$-\triangle u = f \text{ in } \Omega, \quad u = g \text{ on } \Omega, \tag{2}$$

where $H^1(\Omega)$ denotes the space of square integrable functions up to the first derivatives. In the one-level additive Schwarz method, for a given overlapping subdomain partition,$\{\Omega_i\}_i$ of the domain Ω, with an overlapping width δ, the following iterative scheme is proposed to find its solution u. For a given $u^{(n)}$, the following problem in each subdomain Ω_i is solved to find $u_i^{(n+1)}$,

$$-\triangle u_i^{(n+1)} = f \text{ in } \Omega_i, \quad u_i^{(n+1)} = u^{(n)} \text{ in } \overline{\Omega} \setminus \Omega_i. \tag{3}$$

Using $u_i^{(n+1)}$, the next iterate is then formed to give

$$u^{(n+1)} = (1 - N\tau)u^{(n)} + \tau \sum_{i=1}^{N} u_i^{(n+1)},$$

where N is the number of subdomains in the partition and τ is a relaxation parameter. Let N_c be the maximum number of subdomains sharing the same geometric position in Ω. With $\tau \leq 1/N_c$, $u^{(n)}$ converges to the solution u of (2) under a suitably chosen space of functions, see [14, 16]. The algorithm can be further extended into a two-level method by introducing the coarse problem,

$$-\triangle w_0^{(n+1)} = f + \triangle u^{(n)} \text{ in } \Omega, \quad w_0^{(n+1)} = 0 \text{ on } \partial\Omega,$$

and by including the coarse problem solution to the iteration formula,

$$u^{(n+1)} = (1 - N\tau)u^{(n)} + \tau \left(\left(\sum_{i=1}^{N} u_i^{(n+1)} \right) + w_0^{(n+1)} \right).$$

In [7], following similarly as in the analysis for the variational inequalities [1], under the assumptions on the stable decomposition property and the strengthened Cauchy-Schwarz inequality, see [14, Section 2.3], the iterates $u^{(n)}$ converge to the exact solution u with the convergence rate $R(\tau)$,

$$a(u - u^{(n+1)}, u - u^{(n+1)}) \leq R(\tau) a(u - u^{(n)}, u - u^{(n)}), \tag{4}$$

where $a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, dx$, and $R(\tau)$ is

$$R(\tau) = 1 - \frac{2}{2 + C_0}\tau + N_c^2 \tau^2 \text{ and } R(\tau) = 1 - \frac{2}{2 + C_0}\tau + 2(N_c^2 + 1)\tau^2$$

in the one-level case and two-level case, respectively. In the above, the constant C_0 is that appears in the stable decomposition property. In a more detail, in the one-level case, the constant C_0 follows the growth of $N_c N H/\delta$ and in the two-level case, the constant C_0 follows the growth of $N_c H/\delta$, under the approximation property assumption on the coarse Hilbert subspace, see [14, Sections 3.5 and 3.6]. Combining our convergence analysis in (4) with the bound for C_0, we can thus conclude that for a suitable choice of τ, the iterates $u^{(n)}$ converge to u in the Hilbert space $H_0^1(\Omega)$,

$$|u^{(n+1)} - u|_1 \leq C |u^{(n)} - u|_1,$$

with the constant $C < 1$ increasing to 1 as N increasing in the one-level case, while with the constant C being robust as N increasing in the two-level case. The convergent rate C in the one-level method deteriorates as the more subdomains in the partition while it is robust to the increase of the number of subdomains in the two-level method, that have been also observed in additive Schwarz preconditioners to algebraic systems in classical numerical methods.

To find a neural network approximate solution, at each iteration in the additive Schwarz methods, we approximate the local problem solution and the coarse problem solution with neural network functions $U_i(x;\theta_i^{(n+1)})$ and $W_0(x;\theta_0^{(n+1)})$ and train the parameters $\theta_i^{(n+1)}$ and $\theta_0^{(n+1)}$ to minimize the cost functions related to each local problem and the coarse problem, respectively. The neural network iterates $U^{(n+1)}$ are then defined as

$$U^{(n+1)} = (1 - N\tau)U^{(n)} + \tau\left(\sum_{i=1}^{N} U_i^{(n+1)} + W_0^{(n+1)}(x, \theta_0^{(n+1)})\right),$$

where $U_i^{(n+1)}(x)$ are $U_i(x;\theta_i^{(n+1)})$ in Ω_i and $U^{(n)}(x)$ in the rest part, i.e., $\overline{\Omega} \setminus \Omega_i$. In the iteration method, we should store all the previous step parameters to obtain the resulting final step solution as a function of x, which is not desirable in the practical calculation.

To obtain a more practical method, we rewrite the above iteration formula as follows: for any x in Ω

$$U^{(n+1)}(x) = (1-|s(x)|\tau)U^{(n)}(x)+\tau\left(\sum_{i\in s(x)} U_i(x,\theta_i^{(n+1)}) + W_0^{(n+1)}(x,\theta_0^{(n+1)})\right), \quad (5)$$

where $s(x)$ denotes the set of subdomain indices sharing x and $|s(x)|$ denotes the number of elements in the set $s(x)$. We introduce

$$\widehat{U}^{(n+1)}(x) := \frac{1}{|s(x)|}\left(\sum_{i\in s(x)} U_i(x,\theta_i^{(n+1)}) + W_0^{(n+1)}(x,\theta_0^{(n+1)})\right) \quad (6)$$

and rewrite the above iteration formula into

$$U^{(n+1)}(x) = (1 - |s(x)|\tau)U^{(n)}(x) + |s(x)|\tau\widehat{U}^{(n+1)}(x).$$

For the iterates $\widehat{U}^{(n+1)}(x)$, they also converge to $u(x)$ in the L^2-norm, see [7], and the following practical one-level (without the term $W_0^{(n+1)}$ in the iteration formula in (5) and (6)) and two-level additive Schwarz algorithms are finally obtained:

Algorithm 1: One-level method (input: $U^{(0)}$, output: $\widehat{U}^{(n+1)}$)
Step 0: Let $U^{(0)}(x)$ be given and $n = 0$.

Step 1: Find $\theta_i^{(n+1)}$ in $U_i(x;\theta_i^{(n+1)})$ for

$$-\triangle u = f \text{ in } \Omega_i, \quad u = U^{(n)} \text{ on } \partial\Omega_i.$$

Step 2: Update $U^{(n+1)}$ at each data set $X_{\partial\Omega_i}$ as, see (6),

$$U^{(n+1)}(x) = (1-\tau|s(x)|)U^{(n)}(x) + \tau|s(x)|\widehat{U}^{(n+1)}.$$

Step 3: Go to **Step 1** with $n = n+1$ or set the output as $\widehat{U}^{(n+1)}$ if the stopping condition is met.

Algorithm 2: Two-level method (input: $U^{(0)}$, output: $\widehat{U}^{(n+1)}$)
Step 0: Let $U^{(0)}(x)$ be given and $n = 0$.
Step 1-1: Find $\theta_i^{(n+1)}$ in $U_i(x;\theta_i^{(n+1)})$ for

$$-\triangle u = f \text{ in } \Omega_i, \quad u = U^{(n)} \text{ on } \partial\Omega_i.$$

Step 1-2: Find $\theta_0^{(n+1)}$ in $W_0(x;\theta_0^{(n+1)})$ for

$$-\triangle w = f + \triangle U^{(n)} \text{ in } \Omega, \quad w = 0 \text{ on } \partial\Omega.$$

Step 2: Update $U^{(n+1)}$ at each data set $X_{\partial\Omega_i}$ as, see (6),

$$U^{(n+1)}(x) = (1-\tau|s(x)|)U^{(n)}(x) + \tau|s(x)|\widehat{U}^{(n+1)}.$$

Step 3: Go to **Step 1-1** with $n = n+1$ or set the output as $\widehat{U}^{(n+1)}$ if the stopping condition is met.

For the neural network iterates $U^{(n)}$ and $\widehat{U}^{(n)}$, the following convergence results are shown

$$|U^{(n+1)} - u|_1 \le |u^{(n+1)} - u|_1 + \frac{1}{1-C}\epsilon,$$

$$\|\widehat{U}^{(n+1)} - u\|_0 \le \frac{C_p}{\tau}\left(|u^{(n+1)} - u|_1 + |u^{(n)} - u|_1 + \frac{2}{1-C}\epsilon\right),$$

where ϵ denotes the approximation error in the local and coarse neural network solutions, $u^{(n)}$ are the iterates in the Hilbert space, C denotes the convergence rate in the Hilbert space iterates $u^{(n)}$, $\|\cdot\|_0$ denotes the L^2-norm, and C_p is the constant in the Poincare inequality, see [7].

As reported in numerical results in [7], the optimization errors also appear in the computed neural network solutions and they resulted in less accurate approximate solutions at each iteration. The resulting errors are observed to have high contrast near the boundary of the overlapping region, that is harder to be approximated by the coarse neural network function. Such optimization error behaviors in the neural network approximation have been analyzed by NTK theory [3, 15]. Regarding the local problems in our iteration method, the parameters in the local neural network function are trained to minimize the cost function, consisting of the residual loss to the differential equation, and the residual loss to the boundary condition. The residual loss to the boundary condition is harder to optimize and such optimization behavior remains as the high contrast error near the overlapping region boundary.

To address such a drawback in our previous method, we form a partitioned neural network function to approximate the solution $u(x)$,

$$U(x;\theta_1,\cdots,\theta_N)=\sum_{i=1}^{N}\phi_i(x)U_i(x;\theta_i),$$

where $\phi_i(x)$ are a partition of unity functions for the given overlapping subdomain partition,

$$\sum_{i=1}^{N}\phi_i(x)=1,\ 0\le\phi_i(x)\le 1,\quad \phi_i(x)=0,\forall x\in\Omega\setminus\Omega_i.$$

We note that in [11] the parameters θ_i are trained to minimize the following global cost function without utilizing the partitioned neural network structure for parallel computing algorithms,

$$\begin{aligned}L(\theta_1,\cdots,\theta_N)=&\frac{1}{|X_\Omega|}\sum_{x\in X_\Omega}|\triangle U(x;\theta_1,\cdots,\theta_N)+f(x)|^2\\&+\frac{1}{|X_{\partial\Omega}|}\sum_{x\in X_{\partial\Omega}}|U(x;\theta_1,\cdots,\theta_N)-g(x)|^2.\end{aligned}$$

In our work, we propose an iteration method where each local parameters θ_i can be trained in parallel for a localized problem at each iteration. Such an iterative solution procedure is more desirable for the partitioned neural networks.

Our new iteration method is as follows:

Algorithm 3: PNN One-level method (input: $U^{(0)}$, output: $\widehat{U}^{(n+1)}$)
Step 0: Set the initial iterate $U^{(0)}=U(x;\theta_1^{(0)},\cdots,\theta_N^{(0)})$.
Step 1: Find $\theta_i^{(n+1)}$ in $\phi_i(x)U_i(x;\theta_i)$ to approximate the local problem solution;

$$-\triangle u=f+\triangle((1-\phi_i(x))U^{(n)})\text{ in }\Omega_i,$$

$$u=0\text{ on }\partial\Omega_i\cap\Omega,\quad u=g\text{ on }\partial\Omega_i\cap\partial\Omega.$$

Step 2: Set the next iterate;

$$U^{(n+1)}=(1-\alpha)U^{(n)}+\alpha\sum_{i=1}^{N}\phi_i(x)U_i(x;\theta_i^{(n+1)})$$

Step 3: If the stoping condition is met then set the output $\widehat{U}(x)=\sum_{i=1}^{N}\phi_i(x)U_i(x;\theta_i^{(n+1)})$, otherwise continue the iteration to go back to Step 1.

In Algorithm 3, for the case of floating subdomains, the zero boundary condition is already enforced in $\phi_i(x)U_i(x;\theta_i^{(n+1)})$ by the partition of unity function $\phi_i(x)$ and only the differential equation comes in the loss function. We can thus expect that the parameter optimization for such a local problem has less optimization errors than those in Algorithms 1 and 2. Its two-level version can be derived by adding the coarse correction term $W_0^{(n+1)}$ to the iterates

$$U^{(n+1)} = (1-\alpha)U^{(n)} + \alpha \left(\sum_{i=1}^{N} \phi_i(x)U_i(x;\theta_i^{(n+1)}) + W_0(x;\theta_0^{(n+1)}) \right),$$

where $W_0(x;\theta_0^{(n+1)})$ is the neural network approximation to the global coarse problem, i.e., to find w in a coarse subspace V_0 of $H_1(\Omega)$ such that

$$-\triangle w = f + \triangle U^{(n)}, \text{ in } \Omega, \quad w = 0, \text{ on } \partial\Omega.$$

For the Algorithm 3 and its two-level version, their convergence can be shown by following similarly as in our previous work [7]. More rigorous convergence analysis will be provided in a complete version of the proceeding paper. We note that at each iteration the local parameters are fully trained for the given differential equation. Even the case, local parameter training cost per each iteration is much smaller than the training cost in the single large neural network. The number of training epochs is much smaller and the gradient update per epoch is also much cheaper for the smaller local neural network functions. The trade-off is that the total training cost in our proposed method also depends on the number of outer iterations. As the more subdomains in the partition, the more outer iterations are needed. It is thus important to include the coarse component to speed up the outer iterations.

4 Numerical results

For the proposed iterative methods, we consider the following simple one-dimensional model problem to compare their convergence behavior,

$$-u'' = f(x) \text{ in } (-1\ 1), \quad u(x) = 0 \text{ at } x = -1,\ 1,$$

where $f(x)$ is chosen to give the exact solution $u(x) = \sin(\pi x)$.

For the domain $(-1\ 1)$ we introduce an overlapping subdomain partition with 10 subdomains. We then consider a partitioned neural network with 10 local neural network functions, that are defined on each subdomains in the overlapping subdomain partition. For all the local neural network functions, the number of parameters is set as 106. We also use the same size of the coarse neural network function with 106 parameters in the two-level method. When training the parameters in the local and coarse neural network functions, we use 10000 training epochs in the gradient method, where we use the Adam optimizer with the learning rate 0.001.

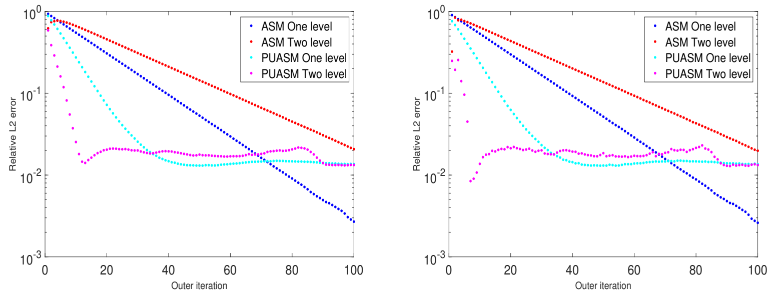

Fig. 1 Error decay history: Left figure (Error plots for $U^{(n)}$), Right figure (Error plots for $\widehat{U}^{(n)}$), ASM One level (Algorithm 1), ASM Two level (Algorithm 2), PUASM One level (Algorithm 3), PUASM Two level (Algorithm 3 with the coarse correction term).

We use randomly selected 100 training data points in each local and coarse problem parameter training.

In Fig. 1, the convergence history in the proposed methods is presented up to 100 outer iterations. The relative L^2-errors in the neural network approximate solutions at each outer iteration are plotted and compared. In the left figure, the errors for the neural network iterates $U^{(n)}$ to the exact solution are plotted for the four proposed methods. In the right figure, the errors for the practical neural network iterates $\widehat{U}^{(n)}$ are plotted, see (6) for the ASM Two level (Algorithm 2) and $\widehat{U}^{(n)} = (\sum_{i=1}^{N} \phi_i(x)U_i(x;\theta_i^{(n)})) + W_0(x;\theta_0^{(n)})$ for PUASM Two level (Algorithm 3 with the coarse correction term). The error plots in both figures show that the coarse correction term in the ASM Two level method does not help to speed up the convergence of the ASM One level. The convergence rate is even larger than the ASM One level method. As discussed earlier, this is related to the optimization error behaviors in the local neural network parameter training, that produce high contrast errors near the boundary of the overlapping region.

In the case of the PUASM Two level method, the coarse problem accelerates the convergence greatly at the early outer iterations. The local problem in the PUASM case has only the residual loss for the differential equation and the high contrast optimization error problems are alleviated in this case with the help of the partition of unity functions. However, at the later outer iterations, the errors can not be further reduced due to the practical implementation issue in the partition of unity functions. The practical implementation issue with the partition of unity functions needs a further investigation and our future research will be focused on proposing some new idea in forming and implementing the partition of unity functions that are suitable for neural network approximation.

Acknowledgements The first author is supported by NRF-2022R1A2C100388511.

References

1. Badea, L. and Wang, J. An additive Schwarz method for variational inequalities. *Mathematics of Computation* **69**(232), 1341–1354 (2000).
2. E, W., Han, J., and Jentzen, A. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Commun. Math. Stat.* **5**(4), 349–380 (2017).
3. Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems* **31** (2018).
4. Jagtap, A. D. and Karniadakis, G. E. 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).
5. Jagtap, A. D., Kharazmi, E., and Karniadakis, G. E. 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**, 113028 (2020).
6. Kim, H. H. and Yang, H. J. Domain decomposition algorithms for physics-informed neural networks. In: *Proceedings of the 26th International Conference on Domain Decomposition Methods* (2021).
7. Kim, H. H. and Yang, H. J. Additive Schwarz algorithms for neural network approximate solutions. *arXiv preprint arXiv:2211.00225* (2022).
8. Li, K., Tang, K., Wu, T., and Liao, Q. D3M: A deep domain decomposition method for partial differential equations. *IEEE Access* **8**, 5283–5294 (2019).
9. Li, W., Xiang, X., and Xu, Y. Deep domain decomposition method: Elliptic problems. In: *Mathematical and Scientific Machine Learning*, 269–286. PMLR (2020).
10. Long, Z., Lu, Y., and Dong, B. PDE-Net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network. *J. Comput. Phys.* **399**, 108925, 17 (2019).
11. Moseley, B., Markham, A., and Nissen-Meyer, T. Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations. *arXiv preprint arXiv:2107.07871* (2021).
12. Raissi, M., 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. *J. Comput. Phys.* **378**, 686–707 (2019).
13. Sirignano, J. and Spiliopoulos, K. DGM: a deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **375**, 1339–1364 (2018).
14. Toselli, A. and Widlund, O. *Domain decomposition methods—algorithms and theory*, *Springer Series in Computational Mathematics*, vol. 34. Springer-Verlag, Berlin (2005).
15. Wang, S., Yu, X., and Perdikaris, P. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics* **449**, 110768 (2022).
16. Xu, J. and Zikatanov, L. The method of alternating projections and the method of subspace corrections in Hilbert space. *J. Amer. Math. Soc.* **15**(3), 573–597 (2002).