

# Iterative Algorithms for Partitioned Neural Network Approximation with the Robin Interface Condition

Young Jae Jeon<sup>[0000-0001-9570-282X]</sup>,  
Hyea Hyun Kim<sup>[0000-0001-7978-5602]</sup>

## 1 Introduction

Neural network approximation to PDEs has become an active research area since its success with physics informed neural networks [9], where the PDE solution is approximated by training neural network parameters to optimize a loss function to the PDE with the prescribed boundary or initial conditions. The new approach can be easily applied to PDEs without meshing the problem domain. It can handle nonlinear, time-dependent, and inverse problems, and can also utilize provided data to improve the accuracy and efficiency of the trained solution. On the other hand, its success highly depends on the hyperparameter settings, such as, network architecture, training samples, the design of the loss function, and the optimizer choice. In addition, the high computational cost for training network parameters is one of the major limitations of the new approach compared to the classical approximation methods.

To address the high computational cost issue, domain decomposition based approaches have been developed in [4, 3, 8, 7, 11, 5]. In [4, 3, 8], partitioned networks based on nonoverlapping or overlapping subdomains are formed to approximate the PDE solution and their parameters are trained for a global loss function, which can take considerable communication cost in a multi-process computation environment, see [10]. In [7, 11, 5], local loss functions are associated with local neural networks and independently trained local neural network solutions are combined with iterative algorithms of domain decomposition methods to give approximate convergent solutions. In the authors' recent work [6], an iterative algorithm was proposed based

---

Young Jae Jeon

Department of Mathematics, Kyung Hee University, Seoul, Republic of Korea, e-mail:  
yjjeon@khu.ac.kr

Hyea Hyun Kim

Department of Applied Mathematics, Kyung Hee University, Yongin, Republic of Korea, e-mail:  
hhkim@khu.ac.kr

on the classical FETI formulation [1] and some preconditioning schemes were also developed for the iterative algorithm in order to speed up the convergence.

In [6], a minimization problem on the interface solution value was solved iteratively by the gradient descent method, where the gradient value is obtained from normal flux values of local solutions sharing the interface. By taking one of the normal flux implicitly, the gradient descent method can be reformulated into a Robin iterative algorithm. In this work, as a continuation of the authors' previous study [6], taking one of the gradient terms implicitly, a Robin iterative algorithm is proposed and its two-level extension is developed to improve the parallel scalability. Numerical results will be provided to show the efficiency of the Robin iterative algorithm.

## 2 Neural network approximation on non-overlapping subdomain partitions

In this section, we include a brief overview of neural network approximation based on non-overlapping subdomain partitions. For the description of the method, we consider the following model elliptic problem on a convex polygonal domain  $\Omega$  in  $\mathbb{R}^2$ ,

$$-\Delta u = f \text{ in } \Omega, \quad u = g \text{ on } \partial\Omega, \quad (1)$$

where  $f$  is a square integrable function in  $\Omega$ .

By dividing the domain  $\Omega$  into nonoverlapping subdomains  $\{\Omega_i\}_{i=1}^N$ , we can rewrite the above model problem into an equivalent partitioned problem, find  $u_i$  in each  $H_1(\Omega_i)$  such that

$$\begin{aligned} -\Delta u_i &= f \text{ in } \Omega_i, \\ u_i &= u_j, \quad \frac{\partial u_i}{\partial n_i} = -\frac{\partial u_j}{\partial n_j} \text{ on } \Gamma_{ij}, \quad \forall j \in s(i), \\ u_i &= g \text{ on } \Gamma_{i0} = \partial\Omega_i \cap \partial\Omega, \end{aligned} \quad (2)$$

where  $s(i)$  denotes the set of subdomain indices  $j$  such that  $\Omega_j$  is the neighboring subdomain that shares the interface  $\Gamma_{ij}$  with  $\Omega_i$ , and  $\frac{\partial u_i}{\partial n_i}$  denotes the outward normal flux for the solution  $u_i$  of the subdomain  $\Omega_i$ . Throughout the paper, we will use the notation  $i$  and  $j$  to denote the subdomain indices.

In conservative physics-informed neural network (cPINN) methods [4], local neural network functions  $U_i(x; \theta_i)$  are employed to approximate the solution  $u_i$  in each subdomain  $\Omega_i$  and the following global loss function is formed to train the local network parameters  $\theta_i$ ,

$$\begin{aligned}
L(\theta_1, \dots, \theta_N) &:= w_I \sum_{i=1}^N \sum_{x \in X(\Omega_i)} |\Delta U_i(x; \theta_i) + f(x)|^2 \\
&+ w_N \sum_{\Gamma_{ij}} \sum_{x \in X(\Gamma_{ij})} \left| \frac{\partial U_i}{\partial n_i}(x; \theta_i) + \frac{\partial U_j}{\partial n_j}(x; \theta_j) \right|^2 \\
&+ w_D \sum_{i=1}^N \sum_{j \in s(i)} \sum_{x \in X(\Gamma_{ij})} |U_i(x; \theta_i) - U_j(x; \theta_j)|^2 + w_E \sum_{i=1}^N \sum_{x \in X(\Gamma_{i0})} |U_i(x; \theta_i) - g(x)|^2,
\end{aligned}$$

where  $X(A)$  denotes the set of sample points chosen from the region  $A$ . In the above,  $w_I$ ,  $w_N$ ,  $w_D$ , and  $w_E$  are weight factors introduced to balance the different terms in the loss function.

In order to utilize the partitioned neural network structure for a multi-processor parallel computing environment, a localized loss function was proposed in the second author's work [5],

$$\begin{aligned}
L_i(\theta_i) &:= w_{i,I} \sum_{x \in X(\Omega_i)} |\Delta U_i(x; \theta_i) + f(x)|^2 \\
&+ w_{i,N} \sum_{j \in s(i)} \sum_{x \in X(\Gamma_{ij})} \left| \frac{\partial U_i}{\partial n}(x; \theta_i) - \bar{U}_{ij,n}(x) \right|^2 \\
&+ w_{i,D} \sum_{j \in s(i)} \sum_{x \in X(\Gamma_{ij})} |U_i(x; \theta_i) - \bar{U}_{ij}(x)|^2 + w_{i,E} \sum_{x \in X(\Gamma_{i0})} |U_i(x; \theta_i) - g(x)|^2,
\end{aligned}$$

where the interface solution value  $\bar{U}_{ij}$  is defined as, with  $0 \leq \alpha \leq 1$ ,

$$\bar{U}_{ij}(x) := (1 - \alpha) \tilde{U}_{ij}(x) + \alpha \frac{1}{2} (U_i(x; \tilde{\theta}_i) + U_j(x; \tilde{\theta}_j))$$

by using the trained parameters  $\tilde{\theta}_i$  and  $\tilde{\theta}_j$  of the previous outer iteration, and the interface value  $\tilde{U}_{ij}$  of the previous outer iteration. The interface normal flux value  $\bar{U}_{ij,n}$  is similarly defined by using the local solution normal flux values. Inside each outer iteration, the local network parameters  $\theta_i$  are independently trained for the loss  $L_i(\theta_i)$  with a certain number of training epochs and the trained local solutions are then used to update the interface values  $\bar{U}_{ij}$  and  $\bar{U}_{ij,n}$  for the next outer iteration.

In [5], data communication between local neural networks is thus required at every outer iteration but not every training epoch in contrast to the cPINN. Such an approach could save the data communication cost greatly and it becomes more suitable for parallel computation, i.e., training local neural networks in parallel by using multiprocessors. On the other hand, the numerical experiments in [5] showed a slower convergence for the iterative solutions, which is related to the fact that the update formula for  $\bar{U}_{ij}$  and  $\bar{U}_{ij,n}$  are not optimized for the model problem. In the authors' previous study [6], an optimized update formula for  $\bar{U}_{ij}$  was proposed based on the classical FETI formulation [1] for the model problem in (1). For the given

interface solution value  $\bar{u}^{(n)}$ , the next iterate  $\bar{u}^{(n+1)}$  is obtained as

$$\bar{u}^{(n+1)} = \bar{u}^{(n)} - \epsilon \left( \frac{\partial u_i^{(n)}}{\partial n_i} + \frac{\partial u_j^{(n)}}{\partial n_j} \right) \text{ on } \Gamma_{ij}, \quad (3)$$

where  $u_i^{(n)}$  and  $u_j^{(n)}$  denote the local problem solutions of  $\Omega_i$  and  $\Omega_j$  for the provided interface value on their boundaries taken from the current interface solution value  $\bar{u}^{(n)}$ . For a sufficiently small  $\epsilon (> 0)$ , the iterates  $\bar{u}^{(n)}$  are shown to converge to the problem solution on the interface  $\Gamma$ . The details of the proposed scheme are listed in Algorithm 1. Due to the small step size assumption on  $\epsilon$ , various preconditioning schemes were also proposed and tested in the authors' previous study [6] in order to speed up the iteration convergence.

---

**Algorithm 1** Gradient descent algorithm (GD)

---

**Step 1:** Train the network parameter  $\theta_i$  of  $U_i(x; \theta_i)$  for the Dirichlet problem,

$$\begin{aligned} -\Delta u_i &= f_i \text{ in } \Omega_i, \\ u_i &= \bar{u}^{(n)}|_{\Gamma_{ij}} \text{ on } \Gamma_{ij}, \quad j \in s(i), \quad u_i = g \text{ on } \Gamma_{i0}. \end{aligned} \quad (4)$$

**Step 2:** Update  $\bar{u}^{(n+1)}$  by using gradient descent method, with a sufficiently small  $\epsilon > 0$ ,

$$\bar{u}^{(n+1)} = \bar{u}^{(n)} - \epsilon r_{ij} \text{ on each interface } \Gamma_{ij},$$

where

$$r_{ij} := \frac{\partial U_i(x; \theta_i)}{\partial n_i} + \frac{\partial U_j(x; \theta_j)}{\partial n_j} \text{ on each interface } \Gamma_{ij}.$$

**Step 3:** Go to Step 1 to proceed the iteration if the stop condition is not met.

---

### 3 Robin iterative algorithms for partitioned neural network approximation

We can improve the iterative scheme in (3) by taking one of the normal flux values, i.e.  $\partial u_i / \partial n_i$ , implicitly to obtain

$$\bar{u}^{(n+1)} + \epsilon \frac{\partial u_i^{(n+1)}}{\partial n_i} = \bar{u}^{(n)} - \epsilon \frac{\partial u_j^{(n)}}{\partial n_j} \text{ on } \Gamma_{ij}. \quad (5)$$

The resulting formula in the above is identical to the Robin interface condition. We then propose Algorithm 2 of the Robin iterative method.

For the Robin iterative method, its convergence is affected by the choice of the parameter  $\epsilon$ . In [2], for a symmetric two-subdomain partition case with local

**Algorithm 2** Robin iterative method (RB)

**Step 1:** Train the network parameter  $\theta_i$  of  $U_i(x; \theta_i)$  for the Robin problem,

$$\begin{aligned} -\Delta u_i &= f_i \text{ in } \Omega_i, \\ u_i + \epsilon \frac{\partial u_i}{\partial n_i} &= \bar{r}_i^{(n)}|_{\Gamma_{ij}} \text{ on } \Gamma_{ij}, \quad j \in s(i), \quad u_i = g \text{ on } \Gamma_{i0}. \end{aligned}$$

**Step 2:** Update  $\bar{r}_i^{(n+1)} = U_j - \epsilon \frac{\partial U_j}{\partial n_j}$  on each interface  $\Gamma_{ij}$ ,  $j \in s(i)$ .

**Step 3:** Go to Step 1 to proceed the iteration if the stop condition is not met.

problems discretized by conforming linear finite elements of the uniform mesh size  $h$ , the optimal parameter is shown to be  $\epsilon = \sqrt{h}$ . In the neural network approximation, we can also train the parameter  $\epsilon$  to minimize the cost function in order to accelerate the Robin iteration scheme adaptively in contrast to the standard finite element approximation. In our computation, we simply choose  $\epsilon = 1/\sqrt{m}$ , where  $m$  is the number of sample points on each  $\Gamma_{ij}$ , following the finite element recipes.

In order to improve the parallel scalability of the Robin iterative algorithm, we now include a coarse neural network  $U_0(x; \theta_0)$  to the neural network approximation  $U(x; \theta_0, \theta_1, \dots, \theta_N)$ ,

$$U|_{\Omega_i} = U_i(x; \theta_i) + \alpha U_0(x; \theta_0)$$

with an adaptively chosen parameter  $\alpha > 0$ . For the choice of  $\alpha$ , we propose the following adaptive strategy based on the residual error:  $\alpha$  is the average value of  $\|f + \Delta U_i(x; \theta_i)\|$  over the training sample points for the coarse network. Such a choice can approximate the magnitude of the coarse solution of the residual equation and can help to effectively train the coarse neural network solution. The resulting two-level Robin iterative method is listed in Algorithm 3.

**Algorithm 3** Two-level Robin iterative method (Two-level RB)

**Step 1:** Train the network parameter  $\theta_0$  for  $U_0(x; \theta_0)$  for the model problem, with  $\alpha^{(0)} = 1$ ,

$$-\Delta(U_i + \alpha^{(n)} u_0) = f \text{ in } \Omega, \quad u_0 = 0 \text{ on } \partial\Omega.$$

**Step 2:** Update  $\bar{r}_i^{(n+1)}$ , using  $\tilde{U}_j := U_j + \alpha^{(n)} U_0$ ,

$$\bar{r}_i^{(n+1)} = \tilde{U}_j - \epsilon \frac{\partial \tilde{U}_j}{\partial n_j} \text{ on each interface } \Gamma_{ij}, \quad j \in s(i).$$

**Step 3:** Train the network parameter  $\theta_i$  of  $U_i(x; \theta_i)$  for the Robin problem,

$$\begin{aligned} -\Delta u_i &= f_i \text{ in } \Omega_i, \\ u_i + \epsilon \frac{\partial u_i}{\partial n_i} &= \bar{r}_i^{(n+1)}|_{\Gamma_{ij}} \text{ on } \Gamma_{ij}, \quad j \in s(i), \quad u_i = g \text{ on } \Gamma_{i0}. \end{aligned}$$

**Step 4:** Update  $\alpha^{(n+1)}$ , using average of  $\|f + \Delta U_i(x; \theta_i)\|$  and go to Step 1 to proceed the iteration if the stop condition is not met.

## 4 Numerical results

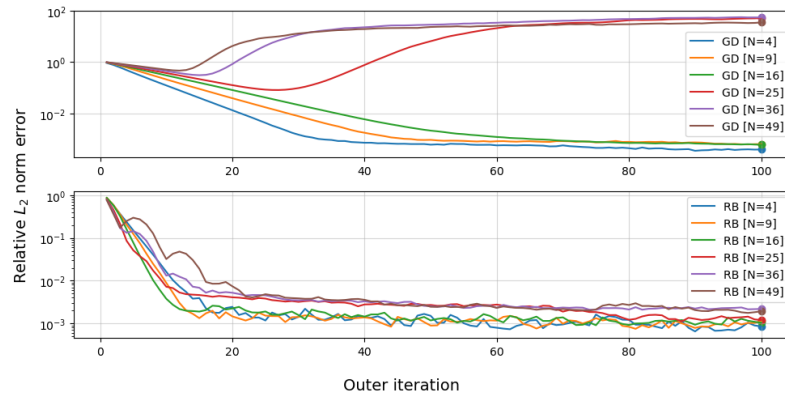
In this section, we evaluate the performance of the proposed algorithms for the Poisson equation on the unit square domain,  $\Omega = (0, 1)^2$ . We consider two model solutions:

$$u_1(x, y) = \sin(\pi x) \sin(\pi y), \quad (6)$$

$$u_2(x, y) = \sin(\pi x) \sin(\pi y) + 0.05 \sin(8\pi x) \sin(8\pi y), \quad (7)$$

where the first model is smooth, and the second model contains high-frequency oscillations. The  $L^2$  errors are computed on  $100 \times 100$  uniform test sample points and the local and coarse problems are trained with 5000 epochs using the Adam optimizer of the learning rate 0.001 inside each outer iteration.

We first compare the GD and RB methods for the smooth model problem (6). We maintain the total number of parameters (approximately 5,500) and training sample points (approximately 5,500) for all configurations and only increase the number of subdomains in the partition. Fig. 1 shows the relative  $L^2$  errors over the outer iterations as increasing the number of subdomains. The GD method converges slowly and becomes significantly less effective as the number of subdomains increases. In contrast, the RB method remains robust and converges rapidly across all subdomain partitions, demonstrating its advantage over the GD method in the one-level setting.



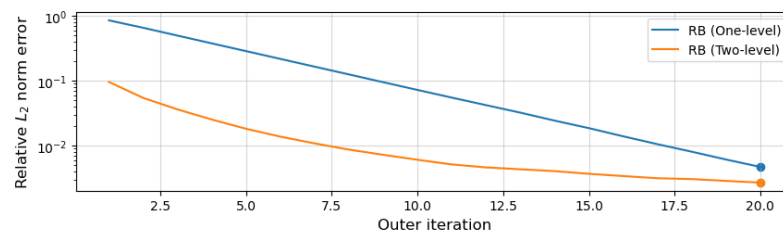
**Fig. 1** Plots of relative  $L^2$  errors over the outer iterations for GD (up) and RB (down) methods as increasing the number of subdomains.

We next consider the model problem (7). For this experiment, we take a  $4 \times 4$  subdomain partition and compare the performance of the RB and the two-level RB methods. Each local network consists of four hidden layers with 13 neurons per layer (599 trainable parameters per subdomain) and is trained with 620 training sample points, resulting in 9,920 total sample points and 9,584 total parameters. Coarse problem training sample points are constructed by taking 59 interior points

randomly per subdomain (total 944 sample points). The coarse network employs the same architecture as the local networks. Both the one-level and two-level methods use the same network and training sample points for the local problems.

Fig. 2 compares the error decay behaviors of the one-level and two-level Robin methods, showing that the two-level method achieves significantly faster convergence than the one-level method. Fig. 3 shows the solution and error plots of the two-level method over the first four outer iterations. The coarse correction can capture the dominant global errors and thus provide effective updates to the Robin interface values for the local problems, leading to faster convergence than the one-level Robin method.

The proposed Robin iterative algorithms show promising results for the test examples. Concrete convergence analysis with a suitable Robin parameter choice, more extensive numerical experiments, and applications to more general PDEs will be studied in our future work.

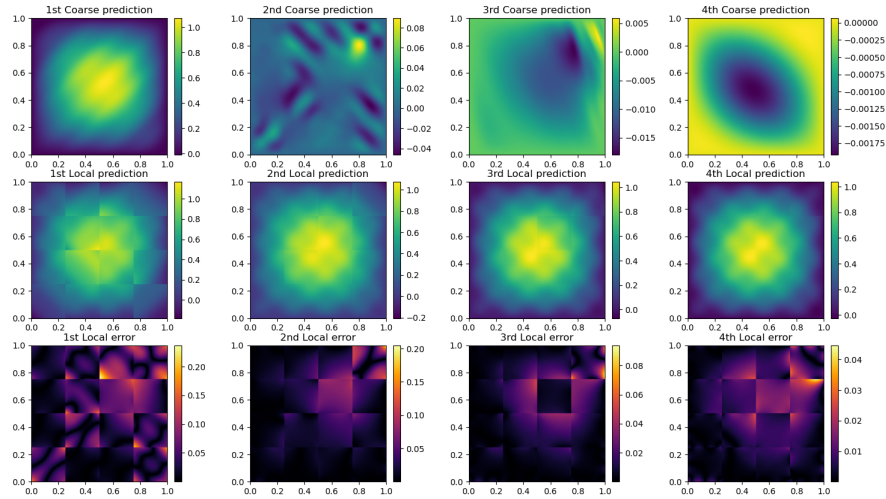


**Fig. 2** Comparison of the relative  $L_2$  error decay for the one-level and two-level Robin iterative methods on a  $4 \times 4$  subdomain partition.

**Acknowledgements** The second author was supported by the National Research Foundation of Korea (NRF) grant RS-2025-00516964.

## References

1. Farhat, C., Roux, F.X.: A method of finite element tearing and interconnecting and its parallel solution algorithm. *Int. J. Numer. Methods Eng.* **32**(6), 1205–1227 (1991)
2. Gander, M.J., Kwok, F.: Best Robin parameters for optimized Schwarz methods at cross points. *SIAM J. Sci. Comput.* **34**(4), A1849–A1879 (2012)
3. Jagtap, A.D., Karniadakis, G.E.: Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Commun. Comput. Phys.* **28**(5) (2020)
4. Jagtap, A.D., Kharazmi, E., Karniadakis, G.E.: Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* **365**, 113028 (2020)
5. Jang, D.K., Kim, K., Kim, H.H.: Partitioned neural network approximation for partial differential equations enhanced with Lagrange multipliers and localized loss functions. *Comput. Methods Appl. Mech. Eng.* **429**, 117168 (2024)



**Fig. 3** Solution and error plots over the first four outer iterations: coarse predictions (top row), updated local solutions (middle row), and errors in the updated local solutions (bottom row)

6. Jeon, Y.J., Kim, H.H.: Neural network tearing and interconnecting methods for partial differential equations. Under review
7. Li, K., Tang, K., Wu, T., Liao, Q.: D3M: A deep domain decomposition method for partial differential equations. *IEEE Access* **8**, 5283–5294 (2019)
8. Moseley, B., Markham, A., Nissen-Meyer, T.: Finite basis physics-informed neural networks (FBPINNs): A scalable domain decomposition approach for solving differential equations. *Adv. Comput. Math.* **49**(4), 62 (2023)
9. Raissi, M., Perdikaris, P., 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)
10. Shukla, K., Xu, M., Trask, N., Karniadakis, G.E.: Scalable algorithms for physics-informed neural and graph networks. *Data-Centric Eng.* **3**, e24 (2022)
11. Yang, H.J., Kim, H.H.: Iterative algorithms for partitioned neural network approximation to partial differential equations. *Comput. Math. Appl.* **170**, 237–259 (2024)