

# Domain Decomposition Architectures and Gauss–Newton Training for Physics-Informed Neural Networks

Alexander Heinlein<sup>[0000–0003–1578–8104]</sup>,  
Taniya Kapoor<sup>[0000–0002–6361–446X]</sup>

## 1 Introduction

In recent years, the use of neural networks (NNs) for solving boundary value problems governed by partial differential equations, as an alternative to classical discretizations such as finite differences or finite elements, has been explored. The approach dates back to the 1990s [3, 12], with popular recent methods including physics-informed NNs (PINNs) [17] and the deep Ritz method [5]. Their implementation is rather simple based on state-of-the-art deep learning frameworks such as TensorFlow, PyTorch, or JAX. Commonly cited additional advantages include their mesh-free, nonlinear nature, improved scaling to higher-dimensional problems, and applicability to parametrized and inverse settings. However, despite these benefits, hyperparameter selection and training remain sensitive, and achieving high accuracy reliably is challenging, especially for complex physical problems.

Difficulties in NN training are also attributed to the spectral bias (a.k.a. frequency principle), meaning that networks learn low-frequency modes faster than high-frequency modes; cf. [16]. This phenomenon can be linked to the use of activation functions with global support; cf. [8]. Consequently, high-frequency errors are reduced slowly, which is particularly problematic for problems with high-frequency solutions. Among others approaches, localizing the network may help alleviate this issue. For example, finite-basis PINNs (FBPINNs) [14] employ window functions based on an overlapping domain decomposition (DD) to localize NNs to subdomains. This strategy improves convergence and accuracy. A multilevel extension [4] further enhances the performance on multi-frequency problems, and extensions to

---

Alexander Heinlein  
Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, Delft, 2628 CD, The Netherlands, e-mail: a.heinlein@tudelft.nl

Taniya Kapoor  
Artificial Intelligence Group, Wageningen University & Research, Wageningen, The Netherlands, e-mail: taniya.kapoor@wur.nl

time-dependent problems and operator learning [7], randomized NNs [1, 18], and Kolmogorov–Arnold networks [9] have also been explored.

Another contributing factor is the widespread use of first-order optimizers, such as stochastic gradient descent (SGD) or Adam [11]. An alternative is based on the Gauss–Newton (GN) method; cf. [2]. For partial differential equation-based loss functions, this is also called the energy natural gradient (ENG) method [15]. This approach can improve convergence, at the cost of solving a linear system involving the Gramian matrix, which may be (nearly) singular and generally ill-conditioned.

In this work, we combine the one-level FBPINN approach with GN training, achieving high accuracy with significantly reduced iteration counts. Because of localization in the FBPINN architecture, in each GN step we obtain a block-sparse system, with off-diagonal blocks between non-overlapping subdomains equal to zero. Related previous work implicitly applied GN optimization to randomized FBPINNs, in which only the last layer is trained, yielding a quadratic optimization problem whose linearization is linear. In [1, 18], efficient linear solvers for these systems have been developed. There are also other prior works on solvers for the random feature method (RFM), which is closely related to randomized FBPINNs. Since these systems share the same structure as our linear systems in each GN step, those techniques could be employed in a more efficient future implementation. Here, we do not yet focus on implementation efficiency, but present a proof of concept and, for the first time, consider GN training for fully trained, not randomized, FBPINNs.

## 2 Finite-basis physics-informed neural networks

For simplicity, we introduce the NN-based approach for a Laplace problem on a Lipschitz domain  $\Omega \subset \mathbb{R}^d$  with boundary  $\partial\Omega$ :

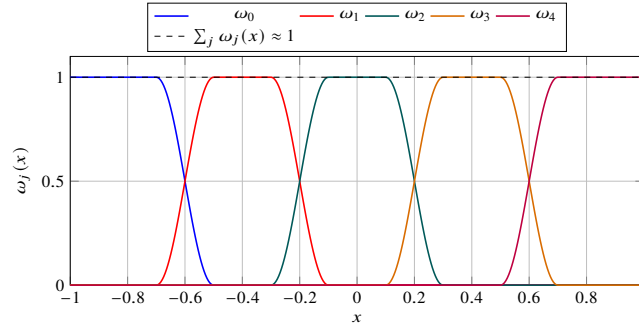
$$-\Delta u = f \quad \text{in } \Omega, \quad u = g \quad \text{on } \partial\Omega. \quad (1)$$

Here,  $f$  and  $g$  are the given source term and boundary data, respectively. Since we employ PINNs, which rely on the strong form of the PDE, we assume that  $u \in C^2(\Omega)$ . Other types of PDEs and boundary conditions can be treated analogously.

### 2.1 Physics-informed neural networks

In order to approximate the solution of eq. (1) using NNs, we introduce the equivalent minimization problem

$$\arg \min_{u \in C^2(\Omega)} \|\Delta u + f\|_{L^2(\Omega)}^2 + \omega \|u - g\|_{L^2(\partial\Omega)}^2,$$



**Fig. 1** Five overlapping partition of unity window functions  $\omega_k$  on  $[-1, 1]$  built from cosine (Hann) windows on overlapping subintervals.

with weighting parameter  $\omega > 0$ . Then, we approximate the integrals in the  $L^2$  norms using sampling and approximate the solution using a NN  $u_\theta$ , where  $\theta \in \mathbb{R}^P$  are the trainable parameters of the NN. We consider simple  $k$ -layer dense feedforward NNs of the form

$$u_\theta(x) = (L^{(k)} \circ \sigma \circ L^{(k-1)} \circ \sigma \circ \dots \circ \sigma \circ L^{(1)})(x),$$

where the  $L^{(i)}(x) = W^{(i)}x + b^{(i)}$  are affine linear transformations with weight matrices  $W^{(i)} \in \mathbb{R}^{d_i \times d_{i-1}}$  and bias vectors  $b^{(i)} \in \mathbb{R}^{d_i}$ , and  $\sigma$  is a nonlinear activation function applied component-wise. In order to enforce boundary conditions, we introduce constraint operator  $C$ , which enforces them explicitly; as a result, the boundary condition loss vanishes. This results in the discrete minimization problem

$$\arg \min_{\theta} \mathcal{L} \quad \text{with} \quad \mathcal{L} =: \frac{1}{N} \sum_{i=1}^N (\Delta C(u_\theta(x_i)) + f(x_i))^2 \quad (2)$$

The points  $\{x_i\}_{i=1}^N \subset \Omega$  are sampled collocation points in the domain. There are various choices for the sampling; see [19] for more details. To solve eq. (2), standard gradient-based optimizers, such as Adam [11], are often employed.

As mentioned before, training is particularly challenging when the solution exhibits high-frequency components, due to the spectral bias [16]; this is further amplified by the spectral properties of the differential operator.

## 2.2 Domain decomposition-based network architecture

In order to localize NNs and improve the approximation of high-frequency components, we employ the one-level FBPINN approach [14]. Specifically, we introduce a set of  $K$  overlapping subdomains  $\{\Omega_k\}_{k=1}^K$  such that  $\bigcup_{k=1}^K \Omega_k = \Omega$ , and a corresponding set of window functions  $\{\omega_k\}_{k=1}^K$  forming a partition of unity, that is,

**Table 1** *Test case 1*: hyperparameters for the domain-decomposed PINN (FBPINN)

optimizer	Adam	Gauss-Newton
learning rate	$lr = 10^{-2}$	constant step size $\alpha = 10^{-2}$
stopping criterion	2000 its. or $\mathcal{L} < 10^{-6}$	5000 its. or $\mathcal{L} < 10^{-6}$
collocation points	$N_f = 1000$ (uniform)	
subdomains	$K = 24$ ; overlap $\delta = 0.5$	
constraint operator	$C(x) = \tanh(k\pi x)$	
subdomain network	MLP [1, 20, 1], $\sigma = \tanh$ ; init.: $W \sim \mathcal{U}[-1, 1]$ , $b = 0$	

**Table 2** *Test case 2*: hyperparameters for the 2D domain-decomposed PINN (FBPINN)

optimizer	Adam	Gauss-Newton
learning rate	$10^{-3}$	constant step size $\alpha = 10^{-2}$
stopping criterion	30 000 its. or $\mathcal{L} < 10^{-5}$	1000 its. or $\mathcal{L} < 10^{-5}$
collocation points	$100 \times 100$ (uniform grid) $\Rightarrow N_f = 10\,000$	
subdomains	$(k_x, k_y) = (2, 2)$ ; overlap $(\delta_x, \delta_y) = (0.5, 0.5)$	
constraint operator	$C(x, y) = \phi(x)\phi(y)$ , where $\phi(s) = 1 - s^2$	
subdomain network	MLP [2, 20, 1], $\sigma = \tanh$ ; init.: $W \sim \mathcal{U}[-1, 1]$ , $b = 0$	

$$\sum_{k=1}^K \omega_k(x) = 1 \quad \text{for all } x \in \Omega, \quad \omega_k(x) = 0 \quad \text{for } x \in \Omega \setminus \Omega_k.$$

We define the overlap between neighboring subdomains by a parameter  $\delta > 0$ . For simplicity, let us consider the one-dimensional case; square and cubic domains in two and three dimensions follow via tensor product. With  $r = \frac{2\delta}{K}$ , we define the window function  $\omega_k(x)$  for the  $k$ -th subdomain  $\Omega_k = (a_k, b_k)$  as

$$\omega_k(x) = \begin{cases} 1/2[1 - \cos(\pi(x - a_k)/r)], & a_k \leq x < a_k + r, \\ 1, & a_k + r \leq x < b_k - r, \\ 1/2[1 - \cos(\pi(x - b_k)/r)], & b_k - r \leq x < b_k, \\ 0, & \text{otherwise,} \end{cases}$$

with

$$a_k = (2k - \delta)/K - 1, \quad b_k = (2(k + 1) + \delta)/K - 1, \quad k = 0, 1, \dots, K - 1.$$

For each overlapping subdomain, we introduce an NN  $u_{\theta_k}$  and construct the global solution as

$$u_{\theta}(x) = \sum_{k=1}^K \omega_k(x) u_{\theta_k}(n_k(x)), \quad (3)$$

where  $\theta = (\theta_1, \dots, \theta_K)$  are the trainable parameters of the local networks, and  $n_k(x)$  is a normalization function mapping  $\Omega_k$  to a reference domain; see [14] for details. Finally, we replace  $u_{\theta}$  in eq. (2) with the FBPINN ansatz eq. (3).

The use of window functions and normalization allows each subdomain network to capture locally high-frequency components efficiently. For problems exhibiting

multiple frequency scales, a multilevel extension of the FBPINN framework [4] can further enhance performance.

### 3 Gauss–Newton training

Usually, the minimization problem in eq. (2) is optimized using methods based on gradient descent (GD),

$$\theta^{(m+1)} = \theta^{(m)} - \alpha \nabla_{\theta} \mathcal{L}(\theta^{(m)}),$$

where  $\theta^{(m)}$  are the parameters at iteration  $m$  and  $\alpha$  is the learning rate. The most popular extension of gradient descent is the Adam optimizer [11], which employs adaptive learning rates and momentum terms.

As discussed in [2, 15], the training performance can be significantly improved by employing the GN method, which yields the following update:

$$\theta^{(m+1)} = \theta^{(m)} - \alpha G^+(\theta^{(m)}) \nabla_{\theta} \mathcal{L}(\theta^{(m)}), \quad (4)$$

where  $G^+(\theta)$  is the pseudoinverse of the energy Gram matrix. For the PINN loss eq. (2), the entries  $G_{ij}$  of  $G(\theta)$  read

$$G_{ij} = \sum_{k=1}^N \Delta(\partial_{\theta_i}(Cu_{\theta}(x_k))) \Delta(\partial_{\theta_j}(Cu_{\theta}(x_k))) \approx \int_{\Omega} \Delta(\partial_{\theta_i}(Cu_{\theta})) \Delta(\partial_{\theta_j}(Cu_{\theta})) dx.$$

GN can be understood as approximating the Hessian of Newton’s method by the energy Gram matrix. Note that  $G(\theta)$  can equivalently be expressed as  $J(\theta)^T J(\theta)$ , where  $J(\theta)$  denotes the Jacobian of the PINN loss. The term  $G^+(\theta) \nabla_{\theta} \mathcal{L}(\theta)$ , appearing in the update eq. (4), is also called the energy natural gradient (ENG) [15].

Because  $G(\theta)$  can be nearly singular, we regularize it as  $G(\theta) + \mu I$  with a small  $\mu > 0$ . For this work,  $\mu = 10^{-5}$  is chosen for all numerical experiments. This regularization approach, which interpolates between the GN and GD algorithms, is also called the Levenberg–Marquardt algorithm; cf. [13].

### 4 Numerical results

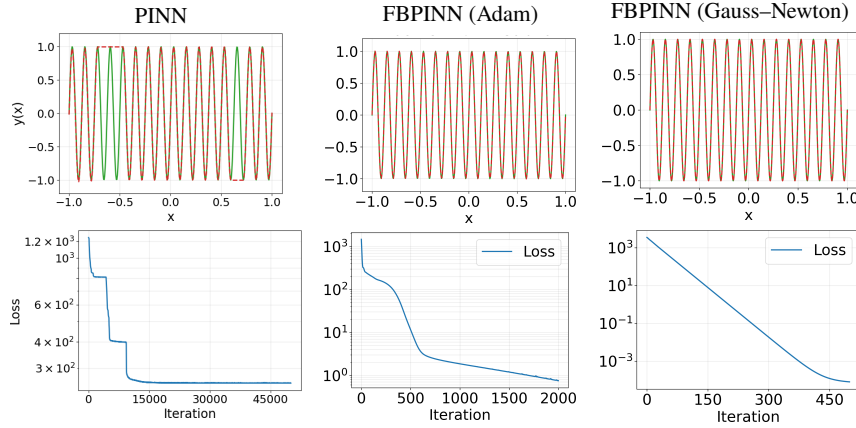
In this section, we validate the performance of the proposed approach using two canonical problems. The first test case is

$$y' - 16\pi \cos(16\pi x) = 0, \quad x \in [-1, 1],$$

with  $y(0) = 0$ . As the second case, we consider the two-dimensional Helmholtz equation

**Table 3** Relative  $\ell_2$  test errors of FBPINN with Adam and Gauss–Newton training for the 1D and 2D problems.

Problem	FBPINN (Adam)	FBPINN (Gauss–Newton)
Test case 1: 1D ODE	$7.8 \cdot 10^{-3}$	$8.0 \cdot 10^{-4}$
Test case 2: 2D Helmholtz	$2.0 \cdot 10^{-4}$	$1.5 \cdot 10^{-4}$



**Fig. 2** Comparison of prediction in orange against the analytical solution in green (top) and training loss (bottom) of three methods for the ordinary differential equation problem: vanilla PINN (left), FBPINN with Adam (middle), and FBPINN with Gauss–Newton (right).

$$\Delta u(x, y) + k^2 u(x, y) = f(x, y), \quad (x, y) \in [-1, 1]^2,$$

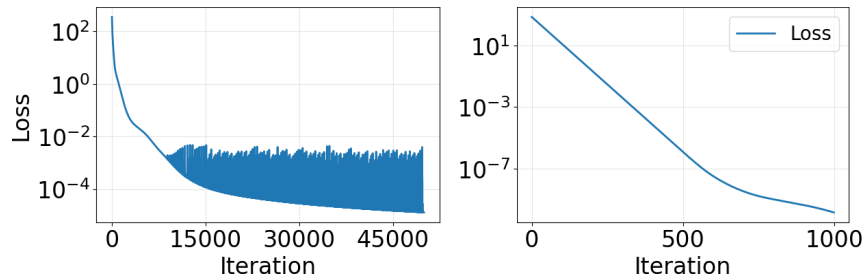
subject to homogeneous Dirichlet boundary conditions. We consider a low wave number case with  $k = 1$  and the source term

$$f(x, y) = -4 + 2(x^2 + y^2) + (1 - x^2)(1 - y^2),$$

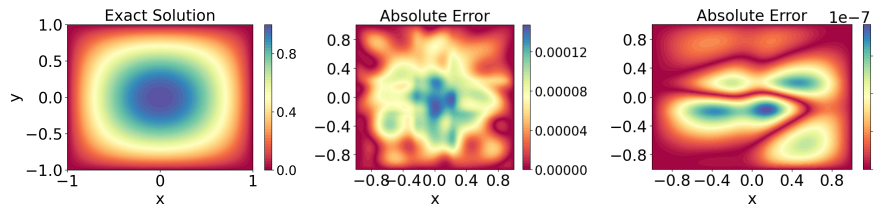
which yields the exact solution  $u_{\text{exact}}(x, y) = (1 - x^2)(1 - y^2)$ . In this work, we do not vary the number of subdomains or the overlap; a detailed study is left for future work. Throughout this section, the test error denotes the absolute or relative  $\ell_2$  error computed on the training collocation points.

As the baseline for the 1D problem, we employ a fully connected NN with four hidden layers of 30 neurons each, a tanh activation, and Glorot initialization [6]; this gives a total of 2881 trainable parameters. We sample  $N_f = 1000$  collocation points uniformly in  $[-1, 1]$  and train with the Adam optimizer [11] using a learning rate of  $10^{-3}$  and full batches for 50 000 steps. This vanilla PINN attains a relative error of  $7.5 \cdot 10^{-1}$  on test case 1. We compare this baseline with FBPINNs trained using Adam and Gauss–Newton, with hyperparameters summarized in Table 1.

Table 3 reports the relative  $\ell_2$  errors for the 1D ODE. FBPINN trained with Adam attains a test error of  $7.8 \cdot 10^{-3}$ , improving over the baseline PINN, and Gauss–Newton reduces the error by an order of magnitude to  $8.0 \cdot 10^{-4}$ . The loss and



**Fig. 3** FBPINN training loss for the 2D Helmholtz problem: Adam (left) and Gauss–Newton (right).



**Fig. 4** Exact solution (left) and absolute errors for Adam (middle) and Gauss–Newton (right).

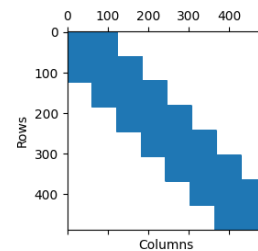
prediction curves in Figure 2 show that the vanilla PINN converges slowly, the FBPINN improves accuracy, and Gauss–Newton further accelerates convergence while aligning the prediction with the reference solution.

We did not conduct a detailed study on the NN architectures. Nonetheless, even PINN models with more parameters continue to suffer from spectral bias and difficulties in learning highly oscillatory functions; see, for example, [14, 4, 10].

While we do not report computing times, Figure 5 illustrates the sparsity structure of the Gramian  $G$  induced by the FBPINN architecture. A fully connected network would yield a dense Gramian, but the domain decomposition creates pronounced sparsity, allowing the use of efficient iterative solvers such as those in [18]; here, for stability, we use `jax.numpy.linalg.lstsq`, which solves via singular value decomposition.

The 2D Helmholtz problem shows the same pattern. We compare FBPINNs trained with Adam and Gauss–Newton using the hyperparameters in Table 2. The relative errors reduce from  $1.1 \cdot 10^{-4}$  with Adam to  $4.8 \cdot 10^{-7}$  with GN, confirming a consistent improvement; see the error plots in Figure 4. The loss curves in Figure 3 likewise show slower convergence for Adam and faster decay with closer agreement for Gauss–Newton.

Overall, training with the Gauss–Newton method significantly accelerates convergence and also improves accuracy across both model problems.



**Fig. 5** Exemplary sparsity pattern of the Gramian  $G$  for the FBPINN architecture for the one dimensional ODE problem with 8 subdomains. Its condition number is approximately  $10^{20}$ .

**Acknowledgements** We would like to thank Marius Zeinhofer and Rami Masri for the helpful discussions about the presented approach.

## References

1. Anderson, S., Dolean, V., Moseley, B., Pestana, J.: ELM-FBPINN: Efficient finite-basis physics-informed neural networks (2024). ArXiv:2409.01949
2. Cai, Z., Ding, T., Liu, M., Liu, X., Xia, J.: A structure-guided Gauss–Newton method for shallow ReLU neural networks (2024). ArXiv:2404.05064
3. Dissanayake, M.W.M.G., Phan-Thien, N.: Neural-network-based approximations for solving partial differential equations. *Commun. Numer. Methods Eng.* **10**(3), 195–201 (1994)
4. Dolean, V., Heinlein, A., Mishra, S., Moseley, B.: Multilevel domain decomposition-based architectures for physics-informed neural networks. *Comput. Methods Appl. Mech. Eng.* **429**, 117116 (2024)
5. E, W., Yu, B.: The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.* **6**(1), 1–12 (2018)
6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: *Proc. Int. Conf. Artif. Intell. Stat.*, pp. 249–256 (2010)
7. Heinlein, A., Howard, A.A., Stinis, P., Beecroft, D.: Multifidelity domain decomposition-based physics-informed neural networks and operators for time-dependent problems. In: *Math. Optim. Mach. Learn.*, pp. 79–92. De Gruyter (2025)
8. Hong, Q., Siegel, J.W., Tan, Q., Xu, J.: On the activation function dependence of the spectral bias of neural networks (2022). ArXiv:2208.04924
9. Howard, A.A., Jacob, B., Murphy, S.H., Heinlein, A., Stinis, P.: Finite basis Kolmogorov–Arnold networks: domain decomposition for data-driven and physics-informed problems (2024). ArXiv:2406.19662
10. Kapoor, T., Wang, H., Núñez, A., Dollevoet, R.: Transfer learning for improved generalizability in causal physics-informed neural networks for beam simulations. *Eng. Appl. Artif. Intell.* **133**, 108085 (2024)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017). ArXiv:1412.6980
12. Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **9**(5), 987–1000 (1998)
13. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. *Q. Appl. Math.* **2**(2), 164–168 (1944)
14. 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)
15. Müller, J., Zeinhofer, M.: Achieving high accuracy with PINNs via energy natural gradient descent. In: *Proc. Int. Conf. Mach. Learn.*, pp. 25471–25485 (2023)
16. Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., Courville, A.: On the spectral bias of neural networks. In: K. Chaudhuri, R. Salakhutdinov (eds.) *Proceedings of the 36th International Conference on Machine Learning, Proceedings of Machine Learning Research*, vol. 97, pp. 5301–5310 (2019)
17. 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)
18. Shang, Y., Heinlein, A., Mishra, S., Wang, F.: Overlapping Schwarz preconditioners for randomized neural networks with domain decomposition. *Comput. Methods Appl. Mech. Eng.* **442**, 118011 (2025)
19. Wu, C., Zhu, M., Tan, Q., Kartha, Y., Lu, L.: A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Comput. Methods Appl. Mech. Eng.* **403**, 115671 (2023)