Neural Network Interface Condition Approximation in a Domain Decomposition Method Applied to Maxwell's Equations

Tobias Knoke, Sebastian Kinnewig, Sven Beuchler, and Thomas Wick

1 Introduction

The time-harmonic Maxwell equations are of great interest in current research fields, e.g., [7, 8, 10, 14, 16]. As their numerical solution is challenging due to their ill-posed nature, e.g., [2], suitable techniques need to be applied. The most prominent technique in literature is based on domain decomposition techniques [5, 17]. The work of Hiptmair [10] can only be applied for the problem in the time domain (i.e., the well-posed problem).

In this work, we design a proof of concept to approximate the interface operator with the help of a feedforward neural network [3, 9, 12]. To this end, a two-domain problem is designed, which is then trained by exchanging data from a modern finite element library deal.II [1] and the well-known PyTorch [15] library. Our main aim is to showcase that our approach is feasible and can be a point of departure for detailed future investigations. An extended version with more technical details and additional computations is [13].

The outline of this work is as follows: In Section 2 we introduce the time-harmonic Maxwell's equations and our notation. Next, in Section 3, domain decomposition and neural network approximations are introduced. Afterward, we address in detail the training process in Section 4. In Section 5, numerical tests demonstrate our proof of concept.

Tobias Knoke, Sebastian Kinnewig, Sven Beuchler, Thomas Wick
Leibniz University Hannover, Institute of Applied Mathematics, Welfengarten 1, 30167 Hannover, Germany, e-mail: tobias.knoke@stud.uni-hannover.de,
{beuchler,kinnewig,thomas.wick}@ifam.uni-hannover.de
and Cluster of Excellence PhoenixD (Photonics, Optics, and Engineering - Innovation Across Disciplines), Leibniz Universität Hannover, Germany

2 Equations

Let $\Omega \subset \mathbb{R}^2$ (here dimension 2, but usually we deal with dimension 3 in Maxwell's equations) be a bounded domain with sufficiently smooth boundary Γ. The latter is partitioned into $\Gamma = \Gamma^\infty \cup \Gamma^{\text{inc}}$. Furthermore, the time-harmonic Maxwell equations are then defined as follows: Find the electric field E such that

$$\begin{cases} \operatorname{curl}\left(\mu^{-1}\operatorname{curl}(E)\right) - \omega^2 E &= 0 \quad \text{in } \Omega \\ \mu^{-1}\gamma^t\left(\operatorname{curl}(E)\right) - i\omega\gamma^T(E) &= 0 \quad \text{on } \Gamma^\infty \\ \gamma^T(E) &= \gamma^T\left(E^{\text{inc}}\right) \text{ on } \Gamma^{\text{inc}}, \end{cases} \tag{1}$$

where $E^{\text{inc}} \colon \mathbb{R}^2 \to \mathbb{C}^2$ is some given incident electric field, $\omega > 0$ is the wave number, μ is the relative permeability and i denotes the imaginary number. For the weak form and corresponding definitions, we seek $E \in H(\operatorname{curl}, \Omega) := \{v \in \mathcal{L}^2(\Omega) \mid \operatorname{curl}(v) \in \mathcal{L}^2(\Omega)\}$. The traces $\gamma^t \colon H(\operatorname{curl}, \Omega) \to H_\times^{-1/2}(\operatorname{div}, \Gamma)$ and $\gamma^T \colon H(\operatorname{curl}, \Omega) \to H_\times^{-1/2}(\operatorname{curl}, \Gamma)$ are defined by

$$\gamma^t(v) = n \times v \quad \text{and} \quad \gamma^T(v) = n \times (v \times n),$$

where $n \in \mathbb{R}^2$ is the normal vector of Ω, $H_\times^{-1/2}(\operatorname{div}, \Gamma) := \{v \in H^{-1/2}(\Gamma) \mid v \cdot n = 0,\ \operatorname{div}_\Gamma v \in H^{-1/2}(\Gamma)\}$ is the space of well-defined surface divergence fields and $H(\operatorname{curl}, \Gamma) := \{v \in H^{-1/2}(\Gamma) \mid v \cdot n = 0,\ \operatorname{curl}_\Gamma(v) \in H^{-1/2}(\Gamma)\}$ is the space of well-defined surface curls. System (1), as well as its weak form (not shown here), is called time-harmonic, because the time dependence can be expressed by $e^{i\omega\tau}$, where $\tau \geq 0$ denotes the time.

For the implementation with the help of a Galerkin finite element method (FEM), we need the discrete weak form. Based on the De-Rham cohomology, we need to choose our basis functions out of the Nédélec space $\mathcal{N}_h^p(\Omega)$. For the description of the Nédélec space we refer to [4]. The discrete form is given by, find $E_h \in \mathcal{N}_h^p(\Omega)$ such that

$$\begin{aligned} \int_\Omega \mu^{-1}\operatorname{curl}(E_h)\operatorname{curl}(\Phi_h) - \omega^2 E_h\Phi_h \,\mathrm{d}x + \int_{\Gamma^\infty} i\omega\gamma^T(E_h)\,\gamma^T(\Phi_h)\,\mathrm{d}s \\ = \int_{\Gamma^\infty} \gamma^T(E_h^{inc})\gamma^T(\Phi_h)\,\mathrm{d}s\ \forall\Phi_h \in \mathcal{N}_h^p(\Omega). \end{aligned} \tag{2}$$

For a more in-depth derivation of equations (1) and their discretization see [14].

3 Numerical approach

3.1 Domain decomposition

Since the solution of the Maxwell equation system (1) is challenging, we apply a non-overlapping domain composition method (DDM) in which the domain is divided into subdomains as follows

$$\Omega = \bigcup_{i=0}^{n_{\mathrm{dom}}} \Omega_i \quad \text{with}$$
$$\Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j,$$

in such a way, that every subdomain Ω_i becomes small enough, so it can be handled with a direct solver. The global solution of the electric field E is obtained via an iterative method, where we solve the time-harmonic Maxwell's equations on each subdomain with suitable interface conditions between the different subdomains. So we obtain a solution E_i^k for every subdomain Ω_i, where k denotes the k-th iteration step. The initial interface condition is given by

$$g_{ji}^{k=0} := -\mu^{-1}\gamma_i^t \left(\mathrm{curl}\left(E_i^{k=0}\right)\right) - ikS\left(\gamma_i^T\left(E_i^{k=0}\right)\right) = 0, \tag{3}$$

where S describes the surface operator [6]. Please note that in ikS, i denotes the imaginary number, while as subscript, i is an index. Afterwards, the electric-field E_i^{k+1} is computed at each step by solving the following system

$$\begin{cases} \mathrm{curl}\left(\mu^{-1}\mathrm{curl}\left(E_i^{k+1}\right)\right) - \omega^2 E_i^{k+1} & = 0 & \text{in } \Omega_i \\ \mu^{-1}\gamma_i^t\left(\mathrm{curl}\left(E_i^{k+1}\right)\right) - i\omega\gamma_i^T\left(E_i^{k+1}\right) & = 0 & \text{on } \Gamma_i^{\infty} \\ \gamma_i^T\left(E_i^{k+1}\right) & = \gamma_i^T\left(E_i^{\mathrm{inc}}\right) & \text{on } \Gamma_i^{\mathrm{inc}} \\ \mu^{-1}S\left(\gamma_i^t\left(\mathrm{curl}\left(E_i^{k+1}\right)\right)\right) - i\omega\gamma_i^T\left(E_i^{k+1}\right) & = g_{ji}^k & \text{on } \Sigma_{ij}, \end{cases} \tag{4}$$

where $\Sigma_{ij} = \Sigma_{ji} := \partial\Omega_i \cap \partial\Omega_j$ denotes the interface of two neighbouring elements and the interface condition is updated by

$$g_{ji}^{k+1} = -\mu^{-1}\gamma_i^t\left(\mathrm{curl}\left(E_i^{k+1}\right)\right) - ikS\left(\gamma_i^T\left(E_i^{k+1}\right)\right) = -g_{ij}^k - 2ikS\left(\gamma_i^T\left(E_i^{k+1}\right)\right). \tag{5}$$

In case of success we obtain $\lim_{k\to\infty} E_i^k = E|_{\Omega_i}$, but this convergence depends strongly on the chosen surface operator S (see [5, 6]).

3.2 Neural network approximation

Since the computation of a good approximation of S is challenging, we examine a new approach in which we attempt to approximate this operator with the help of a neural network (NN). For a first proof of concept, we choose a prototype example and explore whether at all an NN can approximate the values on the interfaces. As it is not feasible to compute the exact surface operator S, we aim to compute g_{ij}^{k+l}, $l > 0$ with an NN, where we use g_{ij}^{k} and E_i^{k+1} as input. Another benefit of this approach is that we can generate easily a training data set from a classical domain decomposition method, as described in section 4.3. For simplicity, we choose $S = \mathbb{1}$ inside of our classical domain decomposition method. Hence, the advantage of this approach is that the interface condition can be updated without recomputing the system (4) at each step, rising hope to reducing the computational cost.

4 Neural network training

In this section, we describe the training process. Besides the mathematical realization, we also need to choose the software libraries. For computing the time-harmonic Maxwell equations with the finite element method (FEM), we utilize deal.II [1]. The neural network is trained with PyTorch [15].

4.1 Decomposing the domain

Before we construct the NN, we choose the domain, the decomposition and the grid on which the system (4) is solved to obtain the training values, because they will influence the size of the network. The domain in our chosen example, given by $\Omega = (0,1) \times (0,1)$ is divided into two subdomains $\Omega_0 = (0,1) \times (0,0.5)$ and $\Omega_1 = (0,1) \times (0.5,1)$, see Figure 1 and the grid (obtained from two times uniform refinement) on which the FEM is applied is a mesh of 32×32 elements with quadratic Nédélec elements.

Hence, 32 elements with each 4 degrees of freedom (dofs) are located on the interface in both subdomains. We evaluate the interface condition and the solution on each dof and use the values as the input and the target of the NN. Therefore the input contains $4 \cdot \dim(g_{ij}) + 4 \cdot \dim(E_i) = 16$ values and the output consists of $4 \cdot \dim(g_{ji}) = 8$ values and we obtain 32 input-target pairs with one computation.

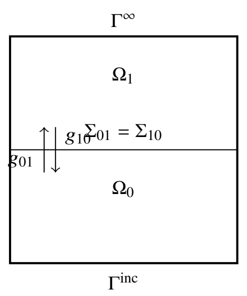

Fig. 1 Visualization of the domain Ω with the chosen decomposition

4.2 Neural network construction

Regarding the previous considerations, we need an input layer with 16 neurons and an output layer with 8 neurons. Furthermore, we use one hidden layer with 500 neurons. Our tests revealed, that this is a sufficent and reasonable size for our purpose, since it leads to more effective networks in terms of error minimization and training duration than other sizes we tested (e.g. 50, 100 and 300 neurons in the hidden layer). The activation functions used are the sigmoid function given by $f(x) = (1 + e^{-x})^{-1}$ in the hidden layer, which turned out to be the most effective among those we tested (e.g. $\tanh(x)$, $\log\left((1 + e^{-x})^{-1}\right)$ and $\max(0, x) + \min(0, e^x - 1)$) and the identity in the other layers. Moreover, we apply separate networks N_{01} and N_{10} of the same shape for both interface conditions g_{01} and g_{10}, since it turned out that they are approximated differently fast and accurately.

4.3 Training

To obtain enough training data, we vary the boundary condition E^{inc} and create a set of training values and a set of test values to control the network during the training and avoid overfitting. The training set and the test set are generated by the boundary values that are displayed in Table 1.

Table 1 Boundary values for generating the training set and the test set

E^{inc} for the training set		E^{inc} for the test set
$\begin{pmatrix} e^{\frac{-(x-0.7)^2}{0.008}} \\ 0 \end{pmatrix}$	$\begin{pmatrix} \cos(\pi^2 y) + \sin(\pi^2 x)i \\ \sin(\pi^2 y) + 0.5\cos(\pi^2 x)i \end{pmatrix}$	$\begin{pmatrix} e^{\frac{-(x-0.5)^2}{0.003}} \\ 0 \end{pmatrix}$
$\begin{pmatrix} e^{\frac{-(x-0.2)^2}{0.002}} \\ 1 \end{pmatrix}$	$\begin{pmatrix} \sin(\pi^2 x) + \sin(\pi^2 x)i \\ \sin(\pi^2 y) + 0.5\cos(\pi^2 x)i \end{pmatrix}$	$\begin{pmatrix} \cos(\pi^2 y) + \sin(\pi^2 x)i \\ \cos(\pi^2 y) + 0.5\cos(\pi^2 x)i \end{pmatrix}$
$\begin{pmatrix} e^{\frac{-(x-0.7)^2}{0.003}} \\ 1 \end{pmatrix}$	$\begin{pmatrix} \sin(\pi^2 x) + \sin(\pi^2 x)i \\ \sin(\pi^2 x) + 0.5\cos(\pi^2 x)i \end{pmatrix}$	
$\begin{pmatrix} e^{\frac{-(x-0.8)^2}{0.003}} \\ \sin(\pi^2 x) \end{pmatrix}$	$\begin{pmatrix} \cos(\pi^2 y) + \sin(\pi^2 x)i \\ \cos(\pi^2 x) + 0.5\cos(\pi^2 x)i \end{pmatrix}$	
$\begin{pmatrix} e^{\frac{-(x-0.5)^2}{0.003}} \\ \cos(\pi^2 x) \end{pmatrix}$	$\begin{pmatrix} \cos(\pi^2 x) + \sin(\pi^2 x)i \\ \cos(\pi^2 y) + 0.5\cos(\pi^2 x)i \end{pmatrix}$	

Since we choose 10 different boundary values for the training set and 2 for the test set and each of them generates a set of 32 training/test values (one per element on the interface), we obtain all in all a set of $32 \cdot 10 = 320$ input-target pairs (each with with a total of $16 + 8 = 24$ values) for the training and a test set of $32 \cdot 2 = 64$ input-target pairs for both networks. To keep the computations simple, we choose a small wave number $\omega = \epsilon\frac{2\pi}{3}$, where ϵ denotes the relative permittivity, and compute the sets with the iterative DDM in 4 steps. Afterwards we use the results $\left(g_{ij}^1, E_i^2\right)$

and g_{ji}^3 as the input and the targets to train our NNs with the application of the mean squared error as the loss function and the Adam algorithm [11] as the optimizer. The network N_{01} is trained with the learning rate 10^{-5}. The initial training error 3.12 and the test error 5.87 are reduced to $1.7 \cdot 10^{-4}$ and $3 \cdot 10^{-3}$ after 29 843 steps of the optimization method. At N_{10} the initial training error 0.72 and the test error 1.28 are reduced to $3 \cdot 10^{-4}$ and $4 \cdot 10^{-3}$ after 20 326 steps with learning rate 10^{-5} and after further training with learning rate 10^{-6} in 3706 steps, we finally achieve the training error $2.9 \cdot 10^{-4}$ and the test error $3 \cdot 10^{-3}$.

5 Numerical tests

In this section, we apply the implemented and trained NNs for different numerical examples. For the first example, we choose the following boundary condition

$$E^{\mathrm{inc}}(x,y) = \begin{pmatrix} \cos\left(\pi^2\,(y-0.5)\right) + \sin\left(\pi^2 x\right) i \\ \cos\left(\pi^2 y\right) + 0.5\sin\left(\pi^2 x\right) i \end{pmatrix},$$

and compute the first interface conditions g_{10}^1 and g_{01}^1 and the solutions E_1^1 and E_0^1 by solving (4) and (5) once. Afterwards, these values are passed on to the networks N_{01}

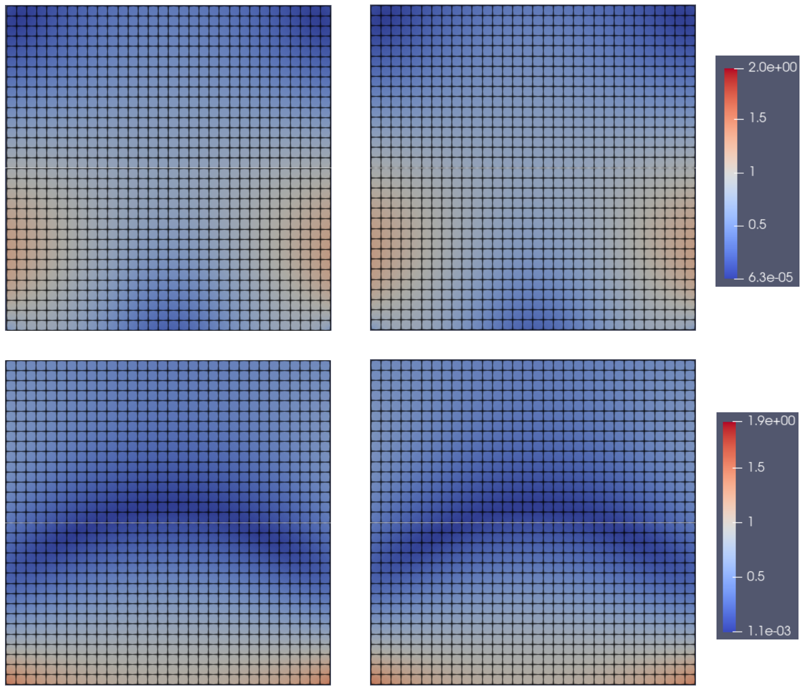

Fig. 2 First example: Real part (above) and imaginary part (below) of the NN solution (left) and the DDM solution (right)

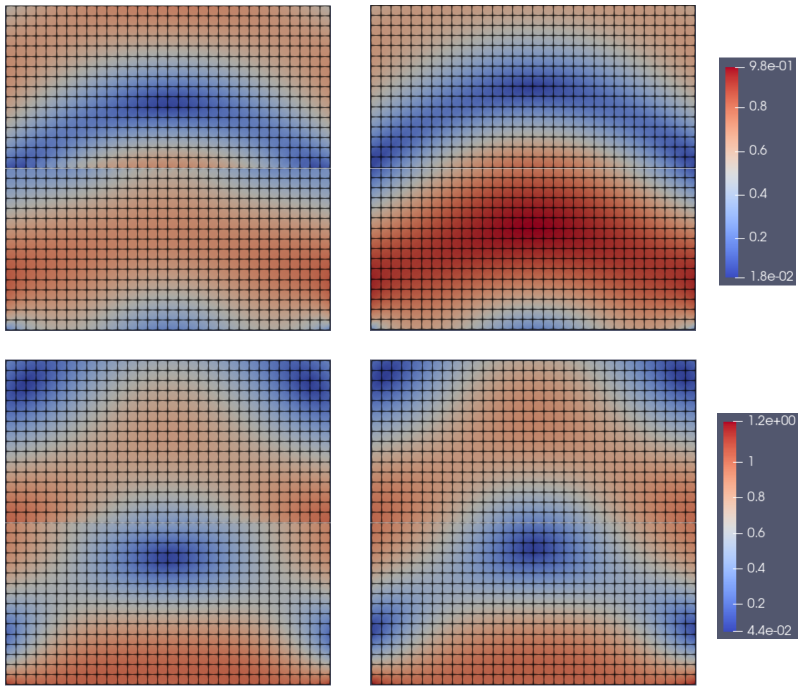

Fig. 3 Second example: Real part (above) and imaginary part (below) of the NN solution (left) and the DDM solution (right)

and N_{10}. The output they return is then handled as our new interface condition which we use to solve system (4) one more time. With that, we obtain the final solution. Moreover, we compute the same example with the DDM in 4 steps. The results that are displayed in Figure 2 show excellent agreement.

As a second example, we increase the wave number, which leads to a more complicated problem. Therefore we repeat the same computation with $\omega = \epsilon\pi$ and leave the other parameters (especially the networks) unchanged. In contrast to the previous example, the results that are displayed in Figure 3 show differences. While the imaginary part is still well approximated, the real part of the NN solution differs significantly from the DDM solution and shows a discontinuity on the interface.

6 Conclusion

In this contribution, we provided a proof of concept and feasibility study for a neural network approximation of the interface conditions in domain decomposition. Analyzing our numerical tests, it can be inferred that the approach works for two subdomains. Ongoing work is the extension to more subdomains.

Acknowledgements This work is funded by the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122, Project ID 390833453).

References

1. Arndt, D., Bangerth, W., Feder, M., Fehling, M., Gassmöller, R., Heister, T., Heltai, L., Kronbichler, M., Maier, M., Munch, P., Pelteret, J.-P., Sticko, S., Turcksin, B., and Wells, D. The `deal.II` library, version 9.4. *Journal of Numerical Mathematics* **30**(3), 231–246 (2022).
2. Beuchler, S., Kinnewig, S., and Wick, T. *Parallel domain decomposition solvers for the time harmonic Maxwell equations*, *Lecture Notes in Computational Science and Engineering*, vol. 145, 615–622. Springer (2023).
3. Bishop, C. M. *Pattern recognition and machine learning*. Springer (2006).
4. Demkowicz, L., Kurtz, J., Pardo, D., Paszynski, M., Rachowicz, W., and Zdunek, A. *Computing with HP-Adaptive Finite Elements, Vol. 2: Frontiers Three Dimensional Elliptic and Maxwell Problems with Applications*. Chapman & Hall/CRC, 1st ed. (2007).
5. Dolean, V., Gander, M. J., and Gerardo-Giorda, L. Optimized Schwarz methods for Maxwell's equations. *SIAM J. Sci. Comput.* **31**(3), 2193–2213 (2009).
6. El Bouajaji, M., Thierry, B., Antoine, X., and Geuzaine, C. A quasi-optimal domain decomposition algorithm for the time-harmonic maxwell's equations. *Journal of Computational Physics* **294**, 38–57 (2015).
7. Faustmann, M., Melenk, J. M., and Parvizi, M. $\mathcal{H}$-matrix approximability of inverses of FEM matrices for the time-harmonic Maxwell equations. *Advances in Computational Mathematics* **48**(5) (2022).
8. Henneking, S. and Demkowicz, L. A numerical study of the pollution error and dpg adaptivity for long waveguide simulations. *Computers & Mathematics with Applications* **95**, 85–100 (2021).
9. Higham, C. F. and Higham, D. J. Deep learning: An introduction for applied mathematicians. *SIAM review* **61**(4), 860–891 (2019).
10. Hiptmair, R. Multigrid method for maxwell's equations. *SIAM Journal on Numerical Analysis* **36**(1), 204–225 (1998).
11. Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization (2017). 1412.6980.
12. Kinnewig, S., Kolditz, L., Roth, J., and Wick, T. *Numerical Methods for Algorithmic Systems and Neural Networks*. Hannover : Institutionelles Repositorium der Leibniz Universität Hannover, Lecture Notes. Institut für Angewandte Mathematik, Leibniz Universität Hannover (2022, https://doi.org/10.15488/11897).
13. Knoke, T., Kinnewig, S., Beuchler, S., Demircan, A., Morgner, U., and Wick, T. Domain decomposition with neural network interface approximations for time-harmonic maxwell's equations with different wave numbers (2023). ArXiv:2303.02590.
14. Monk, P. *Finite Element Methods for Maxwell's Equations*. Oxford Science Publications (2003).
15. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In: *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc. (2019). URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
16. Schöberl, J. A posteriori error estimates for maxwell equations. *Mathematics of Computation* **77**(262), 633–649 (2008).
17. Toselli, A. and Widlund, O. *Domain decomposition methods - algorithms and theory*. Volume 34 of Springer Series in Computational Mathematics. Springer, Berlin, Heidelberg (2005).