Hard-constraining Neumann boundary conditions in physics-informed neural networks via Fourier feature embeddings

Christopher Straub, Philipp Brendel, Vlad Medvedev & Andreas Rosskopf
Department Modeling and Artificial Intelligence
Fraunhofer Institute for Integrated Systems and Device Technology IISB
Schottkystrasse 10, 91058 Erlangen, Germany
{christopher.straub,philipp.brendel,vlad.medvedev,
andreas.rosskopf}@iisb.fraunhofer.de

Abstract

We present a novel approach to hard-constrain Neumann boundary conditions in physics-informed neural networks (PINNs) using Fourier feature embeddings. Neumann boundary conditions are used to described critical processes in various application, yet they are more challenging to hard-constrain in PINNs than Dirichlet conditions. Our method employs specific Fourier feature embeddings to directly incorporate Neumann boundary conditions into the neural network's architecture instead of learning them. The embedding can be naturally extended by high frequency modes to better capture high frequency phenomena. We demonstrate the efficacy of our approach through experiments on a diffusion problem, for which our method outperforms existing hard-constraining methods and classical PINNs, particularly in multiscale and high frequency scenarios.

1 Introduction

Diffusion equations are essential in the study of multiscale processes such as semiconductor modeling (Aleksandrov & Kozlovski, 2009), battery modeling (Doyle et al., 1993), biological systems (Mereghetti et al., 2011), and larger scale phenomena like atmospheric and oceanic transport processes (Sharan & Gopalakrishnan, 2003; Holmes et al., 2021). These problems often contain Neumann boundary conditions, specifying the gradient or flux at a boundary, to represent symmetries or essential physical processes like insulation, recharge rates, or surface charge densities. Physics-informed neural networks (PINNs) offer a promising alternative to classical numerical solvers to simulate such systems (Raissi et al., 2019). The main idea of PINNs is to represent the unknown solution of a physical system as a neural network and train it using the governing physical laws. Despite PINNs being successfully applied in a variety of different settings (Toscano et al., 2025; Ghalambaz et al., 2024), they face some difficulties. As argued in Wang et al. (2021; 2022), two major challenges faced by PINNs are balancing multiple learning objectives (learning one or more differential equations and boundary conditions) and accurately approximating high frequency or multiscale phenomena.

One way to address the former challenge is to incorporate certain requirements on the solution, e.g., boundary conditions, directly into the neural network's structure instead of learning them. This is known as *hard-constraining* and is typically achieved by adding carefully chosen, non-trainable layers to the start or the end of a neural network to transform its input or output, respectively. Hard-constraining techniques are commonly used for Dirichlet or periodic boundary conditions (Toscano et al., 2025). Due to the simplification of the learning task, they typically improve the performances of PINNs (Lu et al., 2021b; Sukumar & Srivastava, 2022; Zeinhofer et al., 2024). Neumann boundary conditions are more challenging to hard-constrain and consequently less frequently hard-constrained, see section 2.2 for a review of existing methods in this direction.

To allow neural networks to approximate high frequency or multiscale data, it has been proposed in Tancik et al. (2020) to expand the input of the neural network into Fourier modes of different frequencies. In the seminal work on the application of PINNs to multiscale problems Wang et al. (2021), it has been shown theoretically and experimentally that such *Fourier feature embeddings* help PINNs to capture high frequency and multiscale phenomena.

In this paper, we present a new way of hard-constraining Neumann boundary conditions based on carefully chosen Fourier feature mappings. Concretely, our contributions are:

- We present a new method to hard-constrain Neumann boundary conditions into PINNs by suitably transforming their input using a single, scalar Fourier embedding. This embedding can further be extended by higher frequencies to better capture high frequency phenomena.
- We demonstrate the efficacy of the new method for a forward diffusion problem with Neumann boundary conditions. The new method yields more accurate results than existing PINN methods, with the most significant improvements observed for high frequencies and for differing frequencies on different scales.

Despite the fact that the numerical experiments conducted in the present workshop paper are limited to a one-dimensional forward problem, it should be emphasized that the method is highly versatile and can also be applied in various other contexts. A selection of these is outlined in appendix B, including the case of a higher dimensional spatial domain.

2 THEORETICAL BACKGROUND

In this section we provide the necessary backgrounds on which the new method relies. As an explicit example to introduce the methods, we consider the one-dimensional heat equation with spatial and temporal domains both normalized to unity, i.e.,

$$\partial_t u = D\,\partial_x^2 u, \qquad (x,t) \in [0,1]^2, \tag{1}$$

where $D > 0$ is a diffusivity parameter. This equation is coupled with vanishing Neumann boundary conditions

$$\partial_x u(0,t) = 0 = \partial_x u(1,t), \qquad t \in [0,1], \tag{2}$$

and initial condition

$$u(x,0) = g(x), \qquad x \in [0,1], \tag{3}$$

for prescribed $g\colon [0,1] \to \mathbb{R}$. Theoretical background on this system is given in appendix A.

2.1 PHYSICS-INFORMED NEURAL NETWORKS

The strategy of PINNs is to represent the sought solution of a physical system as a neural network and train it using the describing physical laws (Raissi et al., 2019). The neural network's architecture is typically a multilayer perceptron (MLP), although all of the below also applies to different architectures, e.g., a Kolmogorov-Arnold network (Liu et al., 2025b; Shukla et al., 2024). The parameters of the neural network are trained by minimizing the composite loss function $\mathcal{L} = \lambda_{\mathrm{PDE}}\,\mathcal{L}_{\mathrm{PDE}} + \lambda_{\mathrm{IC}}\,\mathcal{L}_{\mathrm{IC}} + \lambda_{\mathrm{BC}}\,\mathcal{L}_{\mathrm{BC}}$. The individual loss terms $\mathcal{L}_{\mathrm{PDE}}$, $\mathcal{L}_{\mathrm{BC}}$, and $\mathcal{L}_{\mathrm{IC}}$ are (usually) the mean squared error of the residual of the PDE, the initial condition, and the boundary condition evaluated at randomly sampled collocation points $(x_j^{\mathrm{PDE}}, t_j^{\mathrm{PDE}})_{j=1,\dots,N_{\mathrm{PDE}}} \subset [0,1]^2$, $(x_j^{\mathrm{IC}}, 0)_{j=1,\dots,N_{\mathrm{IC}}} \subset [0,1] \times \{0\}$, and $(x_j^{\mathrm{BC}}, t_j^{\mathrm{BC}})_{j=1,\dots,N_{\mathrm{BC}}} \subset \{0,1\} \times [0,1]$, respectively. The weights $\lambda_{\mathrm{PDE}}, \lambda_{\mathrm{IC}}, \lambda_{\mathrm{BC}} > 0$ can be used to balance the individual loss contributions. Here, the default choice $\lambda_{\mathrm{PDE}} = \lambda_{\mathrm{IC}} = \lambda_{\mathrm{BC}} = 1$ is used. More background on PINNs can be found in Raissi et al. (2019); Lagaris et al. (1998); Karniadakis et al. (2021); Wang et al. (2023).

2.2 EXISTING HARD-CONSTRAINING TECHNIQUES FOR NEUMANN BOUNDARY CONDITIONS

The main idea of hard-constraining is to incorporate certain conditions on the sought solution directly into the neural network's structure. An approach to achieve this for Neumann boundary conditions on general domains has been presented in Sukumar & Srivastava (2022). It is based on distance functions, i.e., functions giving the signed distance to the boundary of the problem's domain. These distance functions are then used to hard-constrain different types of boundary conditions into neural networks. The formulae to hard-constrain Neumann boundary conditions are given in Sukumar & Srivastava (2022, Sc. 5.1.2) and rely on a suitable transformation of the neural network's output to modify the value of its derivatives at the boundary. To explicitly demonstrate this method in the case of the vanishing Neumann boundary conditions given by equation 2, let $u^{\mathrm{NN}} \colon [0,1]^2 \to \mathbb{R}$ be an arbitrary smooth representation model, e.g., a neural network. Inserting the canonical distance function $\phi(x) = x(1-x)$ of $[0,1]$ into the formulae from Sukumar & Srivastava (2022) yields the following transformation of u^{NN}:

$$u^{\mathrm{trafo}}(x,t) = u^{\mathrm{NN}}(x,t) - x(1-x)^2\,\partial_x u^{\mathrm{NN}}(0,t) - x^2(x-1)\,\partial_x u^{\mathrm{NN}}(1,t). \tag{4}$$

The original output u^{NN} is replaced by this transformed version u^{trafo} when evaluating the model and when computing the loss $\mathcal{L}$. The key idea of this transformation is to use the values of the x-derivative of the original representation model $\partial_x u^{\mathrm{NN}}$ to ensure the desired behavior of the transformation u^{trafo} at the spatial boundary. Indeed, a straight-forward calculation shows that u^{trafo} satisfies the vanishing boundary conditions independently of $u^{\mathrm{NN}}(x,t)$, and hence the loss term $\mathcal{L}_{\mathrm{BC}}$ in $\mathcal{L}$ can be dropped when training the model. In Sukumar & Srivastava (2022), it is observed that this remains true if the additional term $x^2(1-x)^2 v^{\mathrm{NN}}(x,t)$ is added to the right-hand side of equation 4, where $v^{\mathrm{NN}} = v^{\mathrm{NN}}(x,t)$ is another trainable representation model. Note, however, that the computational costs of evaluating or backpropagating the transformed model u^{trafo} are significantly increased by incorporating the additional derivatives $\partial_x u^{\mathrm{NN}}$ in equation 4, cf. section 4 for details.

The similar method in the case of an ordinary differential equation with a single Neumann condition has also been proposed in Leake & Mortari (2020). Mathematically more advanced schemes in the situation where Neumann boundary conditions are imposed together with Dirichlet conditions are presented in McFall & Mahan (2009); Shekari Beidokhti & Malek (2009). The latter also rely on output transformations containing suitable evaluations of the neural network at the boundary.

2.3 Fourier feature embeddings

The main idea of Fourier feature embeddings is to expand the input variable(s) of a neural network into multiple frequencies to help the model approximate phenomena with different frequencies, in particular, high frequencies. This has first been proposed in Tancik et al. (2020) in the context of computer vision problems. In Wang et al. (2021), it has then been demonstrated that adding Fourier feature embeddings to PINNs allow them to learn multiscale processes while vanilla PINNs struggle at this task. Concretely, for the diffusion problem given by equations 1–3, the Fourier feature embedding of the spatial coordinate $x \in [0,1]$ proposed in Wang et al. (2021) is

$$x \mapsto (\cos(b_1\pi x), \sin(b_1\pi x), \ldots, \cos(b_m\pi x), \sin(b_m\pi x)),$$

for frequencies $b_1, \ldots, b_m \in \mathbb{R}$. Together with the time variable t, this spatial embedding is then input into a trainable neural network. In Wang et al. (2021), it is further shown how to use multiple spatial Fourier embeddings as well as additional temporal Fourier embeddings. In the present work, we restrict the discussion to the use of one spatial Fourier embedding of a prescribed length $2m$. The frequencies used for the Fourier embedding are typically sampled from a Gaussian distribution with zero mean and prescribed standard deviation σ – this is referred to as a *random* Fourier feature embedding. Other approaches are to learn the frequencies (Dong & Ni, 2021) or to explicitly prescribe them. Fourier feature embeddings can also be used to hard-constrain periodic boundary conditions into PINNs (Dong & Ni, 2021; Lu et al., 2021b). This requires the frequency values $b_1, \ldots, b_m$ to be chosen such that the Fourier mappings are periodic on the problem's domain.

3 Methodology

We now present a new approach to hard-constrain Neumann boundary conditions in PINNs. To introduce the method, we focus on the vanishing Neumann boundary conditions from equation 2 imposed on the spatial domain $x \in [0,1]$. Nonetheless, the method is straight-forward to extend to non-vanishing Neumann boundary conditions and to more general and higher dimensional spatial domains, see appendix B for the formulae in these more general settings. The key mechanisms of the new method are illustrated in Figure 1: A Fourier embedding of frequency 1 (red) is applied to the spatial input x. The Fourier embedding can naturally be extended by higher frequency embeddings (green). The key idea is to only use specific Fourier mappings (only the Cosine with integer frequencies) that are flat at the spatial boundary. Together with additional input coordinates (e.g., the time t), the whole Fourier embedding is input into a trainable neural network. Due to the chain rule, the derivative w.r.t. the spatial variable x vanishes at the boundary of the spatial domain. To adjust for non-vanishing Neumann boundary data, an explicit expression is added to the neural network's output to arrive at the model's final output.

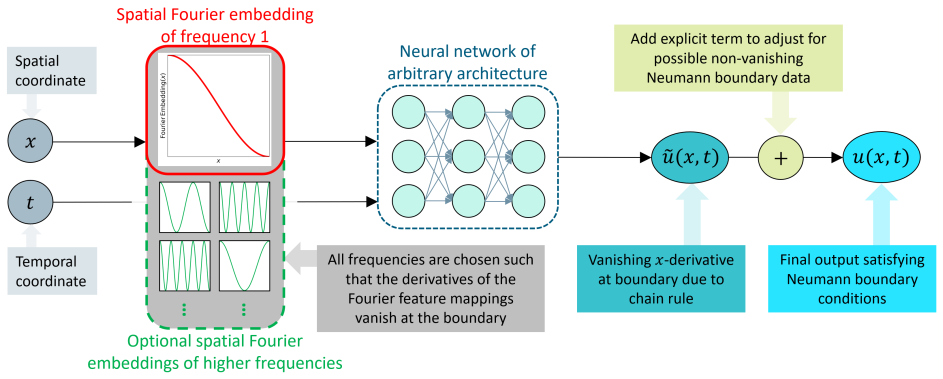
Spatial Fourier embedding of frequency 1
Spatial coordinate
x
t
Temporal coordinate
Fourier Embedding(x)
x
Neural network of arbitrary architecture
Add explicit term to adjust for possible non-vanishing Neumann boundary data
$\tilde{u}(x,t)$
$+$
$u(x,t)$
All frequencies are chosen such that the derivatives of the Fourier feature mappings vanish at the boundary
Vanishing x-derivative at boundary due to chain rule
Final output satisfying Neumann boundary conditions
Optional spatial Fourier embeddings of higher frequencies

Figure 1: The proposed method to hard-constrain Neumann boundary conditions.

Let us now provide the mathematical formulae underpinning these main ideas. The single spatial Cosine Fourier embedding of frequency 1 is given by the following transformation to the spatial input variable $x \in [0, 1]$:

$$x \mapsto \gamma_1(x) = \cos(\pi x). \tag{5}$$

The key property of γ_1 is that it is flat at the spatial boundary, i.e., $\gamma_1'(x) = 0$ for $x = 0$ and $x = 1$. This is also true for the higher frequency embedding $x \mapsto \cos(\pi b x)$ provided that b is an integer. To better capture high frequency phenomena, these embeddings can be appended to $\gamma_1(x)$ to arrive at the total embedding

$$x \mapsto \gamma_{1,b_2,\ldots,b_n}(x) = \left(\cos(\pi x), \cos(\pi b_2 x), \ldots, \cos(\pi b_n x)\right), \tag{6}$$

where the integer frequencies $b_2, \ldots, b_n \in \mathbb{Z}$ are randomly sampled. The embedding of frequency 1 is always included in equation 6 to ensure that no information is lost in the transformation – note that $\gamma_1 \colon [0, 1] \to [-1, 1]$ is one-to-one and $\gamma_1'(x) \neq 0$ for $0 < x < 1$. The embedding given by equation 5 or equation 6 is then inserted into an arbitrary smooth representation model u^{NN}, e.g., a neural network. The resulting model is

$$\tilde{u}(x, t) = u^{\mathrm{NN}}(\gamma(x), t), \tag{7}$$

where γ can be γ_1 or $\gamma_{1,b_2,\ldots,b_n}$. Because the derivative of γ vanishes at the boundary, i.e., $\frac{d}{dx}\gamma(x) = 0$ for $x = 0$ and $x = 1$, the chain rule implies that $\tilde{u}$ indeed satisfies the vanishing Neumann boundary condition from equation 2. Similar to section 2.2, this property remains valid if the additional term $x^2(1 - x)^2 v^{\mathrm{NN}}(x, t)$ is added to the right-hand side of equation 4, where $v^{\mathrm{NN}} = v^{\mathrm{NN}}(x, t)$ is another trainable representation model. Hard-constraining non-vanishing Neumann boundary conditions, i.e., $\partial_x u(0, t) = A$ and $\partial_x u(1, t) = B$ for prescribed $A, B \in \mathbb{R}$ instead of equation 2, is achieved by the transformation

$$u(x, t) = u^{\mathrm{NN}}\left(\gamma(x), t\right) + x(1 - x)^2 A + x^2(x - 1)B. \tag{8}$$

Notice that this transformation is fundamentally simpler than the one presented in section 2.2: In equation 4, derivatives of the neural network evaluated at the boundary are added in the transformation, while the functions added in equation 8 are explicit expressions of the spatial variable x.

4 Experiments

In this section, the performance of the newly proposed method (cf. section 3) is compared with the existing methods (cf. section 2) for the diffusion problem given by equations 1–3. The same problem has been considered in Wang et al. (2021), with the exception that Neumann instead of Dirichlet boundary conditions are used here. The reason for considering this problem is that it exhibits a multitude of qualitatively different solutions for different initial data. The initial conditions used here are given in Table 1. Low or high frequency initial data lead to the respective spatial frequencies of the solution with an additional temporal dependency. In the multiscale case, the solution manifests low frequency on a macro scale and high frequency on a micro scale, resembling numerous practical multiscale scenarios (Wang et al., 2021). As canonical examples for general initial conditions, polynomials of orders 3 and 4 satisfying the vanishing Neumann boundary conditions are considered. These polynomials are strictly increasing and parabola-shaped, respectively, and are normalized to range in $[0, 1]$. They result in solutions composed of superpositions of infinite spatial frequencies, although the solutions are dominated by low frequency effects. The diffusivity parameters D are chosen such that the solutions' temporal decay speeds are similar (cf. appendix A) – the same strategy has been used in Wang et al. (2021).

Table 1: Initial conditions and diffusivity parameters in the diffusion problem (equations 1–3) considered for the numerical experiments. The resulting solutions are stated in appendix A.

Description	Initial condition	Diffusivity parameter
Low frequency	$g(x) = \cos(2\pi x)$	$D = (2\pi)^{-2}$
High frequency	$g(x) = \cos(50\pi x)$	$D = (50\pi)^{-2}$
Multiscale	$g(x) = \cos(2\pi x) + 0.1\cos(50\pi x)$	$D = (50\pi)^{-2}$
Polynom 3rd order	$g(x) = 3x^2 - 2x^3$	$D = \pi^{-2}$
Polynom 4th order	$g(x) = 16x^4 - 32x^3 + 16x^2$	$D = \pi^{-2}$

The basic hyperparameters of all models are chosen similarly to Wang et al. (2021), see appendix C for details. The code is written in the widely used PINN framework DeepXDE (Lu et al., 2021a). Fourier embeddings are easy to implement in DeepXDE by using the pre-build function `apply_feature_transform`. Implementing the existing hard-constraining technique (cf. section 2.2) is not that straightforward due to the derivative evaluation of the neural network contained in equation 4. It can be realized by suitable modifications to the neural network class.

For each of the settings from Table 1, nine different PINN methods are compared with one another. Firstly, the vanilla PINN approach is considered, i.e., no hard-constraining of the Neumann boundary condition. Secondly, a PINN with the existing approach to hard-constrain Neumann boundary conditions (recall section 2.2) is examined. In addition, for both of these methods the use of random Fourier feature embeddings (recall section 2.3) is analyzed. More precisely, Fourier embeddings of sizes 20 and 50 with random frequencies sampled from $\mathcal{N}(0, 20)$ are studied. This distribution and the embedding sizes have been chosen according to the guidelines outlined in Wang et al. (2021). Lastly, the newly proposed approach of hard-constraining Neumann boundary conditions is considered (recall section 3). To compare its performance with the aforementioned methods, a Fourier embedding with single frequency 1 (cf. equation 5) as well as Fourier embeddings of sizes 20 and 50 (cf. equation 6) with random (integer) frequencies sampled from $\mathcal{N}(0, 20)$ are used. For the sake of clarity, we abstain from adding the term $x^2(1-x)^2 v^{\mathrm{NN}}(x,t)$ with an additional trainable neural network v^{NN} to the output of both the existing and the newly proposed method of hard-constraining.

The performances of the different methods are compared in the same way as in Liu et al. (2025a) by computing the relative improvements of the accuracies w.r.t. a reference method. For instance, relative improvements of $+50\%$ and -100% correspond to half and twice the error of the reference method, respectively. The reference method is always the vanilla PINN (i.e., no hard-constraining) with the Fourier embedding strategy leading to the best accuracy. The accuracy always refers to the relative L^2-distance to the analytical solution. To obtain a fair comparison, all models are trained for the same time – the time it took the vanilla PINN to complete 10^6 iterations. The relative improvements of all methods are shown in Figure 2, the accuracies are stated in Table 2 in appendix C.

In the cases where the solution is dominated by low frequency effects, hard-constraining Neumann boundary conditions via a single Fourier embedding of frequency 1 yields the best accuracies. Concretely, compared to the best vanilla PINN (no hard-constraining and no Fourier embeddings), this newly proposed method achieves relative improvements of 54.2%, 47.2%, and 25.6% for low frequency, 3rd order polynomial, and 4th order polynomial initial conditions, respectively. In all of these three cases, the existing hard-constraining technique performs slightly worse than the vanilla

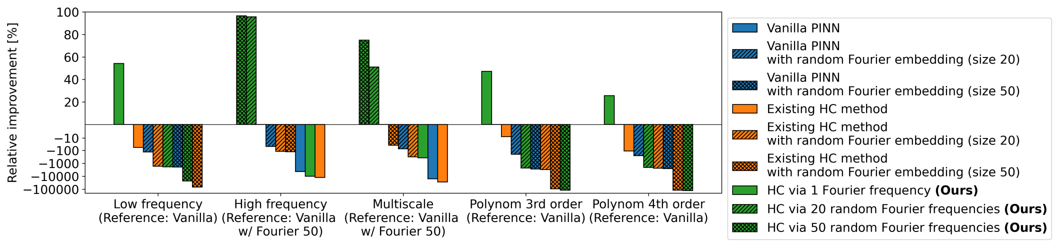

Figure 2: Relative improvements of the accuracies of PINNs with different hard-constraining (HC) and Fourier embedding strategies w.r.t. the reference PINN. All models have been trained for the same time. More details on the simulations are provided in Figure 3 in appendix C.

PINN. Adding high frequency random Fourier embeddings results in a substantial deterioration in the performance of all hard-constraining techniques, where it can be observed that the accuracy decreases as the size of the Fourier embedding increases from 20 to 50. The reason for this is that the PINN's training process to approximate a low frequency solution becomes significantly more complex and unstable when the spatial input x is mainly provided via high frequency feature embeddings.

For the high frequency and multiscale initial conditions, the reference method is the vanilla PINN with random Fourier embedding of size 50. Better performances are achieved only by the newly proposed method of using Fourier embeddings of 20 or 50 random integer frequencies to hard-constrain Neumann boundary conditions. Concretely, hard-constraining the Neumann boundary conditions via a Fourier embedding of size 50 leads to a relative improvement of 96.5% and 74.8% in the high frequency and multiscale setting, respectively. The existing hard-constraining technique combined with a random Fourier embedding again performs slightly worse than the reference method. It can further be seen in Figure 2 that using smaller Fourier embeddings or no Fourier embedding leads to considerably worse accuracies (across all hard-constraining techniques), which is consistent with the analysis in Wang et al. (2021).

In all settings, the main reason for the inferior performance of the existing hard-constraining technique compared to both the vanilla PINN and the newly proposed method of hard-constraining is its significantly longer training time. Concretely, with no Fourier feature embeddings, the vanilla PINN takes (on average) 13.5ms to complete a single training iteration, the existing hard-constraining method 63.1ms, and the newly proposed approach of hard-constraining Neumann boundary conditions via a single Fourier feature mapping takes 14.6ms. When training all models for 10^6 iterations (instead of fixing the training time as in Figure 2), it can be observed that the existing hard-constraining technique mostly performs better than the best vanilla PINN, but still worse than the newly proposed method, cf. Figure 3 in appendix C. This shows that hard-constraining boundary conditions indeed improves the performance of PINNs due to the reduction of the number of training task, which is consistent with the analyses from Lu et al. (2021b); Zeinhofer et al. (2024) in the case of Dirichlet conditions. The newly proposed approach offers a computationally efficient way of benefiting from this effect in the case of Neumann boundary conditions.

The results further show that choosing a suitable Fourier embedding strategy is crucial to obtain accurate results – sufficiently large Fourier embeddings are needed to approximate high frequency behavior and no Fourier embeddings should be used for low frequency behavior. While this can be observed for all hard-constraining techniques, Figure 2 indicates that it is particularly important for the newly proposed approach. Some guidelines on how to choose the Fourier embedding's parameters (i.e., its size and the distribution from which to sample the random frequencies) are outlined in Wang et al. (2021).

5 Conclusion

We have presented a new approach utilizing specific Fourier feature embeddings to hard-constrain Neumann boundary conditions into PINNs. Compared to existing hard-constraining techniques for Neumann boundary conditions, the new method is easier to implement and more computationally efficient. In addition, it demonstrates superior performance in solving a diffusion problem, particularly excelling in high frequency and multiscale settings. Moreover, although we have focused on a simple forward diffusion problem in this study, the method's versatility allows for its application across various contexts including inverse problems and operator learning tasks. The new approach is expected to serve as a valuable tool in representing multiscale processes through PINNs, although further exploration is needed to fully understand its capabilities.

Acknowledgments

This work was supported by the Fraunhofer Internal Programs under Grant No. PREPARE 40-08394.

Data availability statement

The source code used for the experiments is publicly available at

`https://github.com/FhG-IISB-MKI/HC_Neumann_Experiments`

References

O.V. Aleksandrov and V.V. Kozlovski. Simulation of interaction between nickel and silicon carbide during the formation of ohmic contacts. *Semiconductors*, 43:885–891, 2009. doi: 10.1134/S1063782609070100.

Suchuan Dong and Naxian Ni. A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. *Journal of Computational Physics*, 435: 110242, 2021. doi: 10.1016/j.jcp.2021.110242.

Marc Doyle, Thomas F. Fuller, and John Newman. Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell. *Journal of The Electrochemical Society*, 140(6):1526, 1993. doi: 10.1149/1.2221597.

Lawrence C. Evans. *Partial Differential Equations (second edition)*. Graduate studies in mathematics 19. American Mathematical Society, 2010.

Mohammad Ghalambaz, Mikhail A. Sheremet, Mohammed Arshad Khan, Zehba Raizah, and Jana Shafi. Physics-informed neural networks (PINNs): application categories, trends and impact. *International Journal of Numerical Methods for Heat & Fluid Flow*, 34:3131–3165, 2024. doi: 10.1108/HFF-09-2023-0568.

Ryan M. Holmes, Jan D. Zika, Stephen M. Griffies, Andy Hogg, Andrew E. Kiss, and Matthew H. England. The geography of numerical mixing in a suite of global ocean models. *Journal of Advances in Modeling Earth Systems*, 13(7):e2020MS002333, 2021. doi: 10.1029/2020MS002333.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3:422–440, 2021. doi: 10.1038/s42254-021-00314-5.

Isaac Elias Lagaris, Aristidis Likas, and Dimitrios I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9:987–1000, 1998. doi: 10.1109/72.712178.

Carl Leake and Daniele Mortari. Deep theory of functional connections: A new method for estimating the solutions of partial differential equations. *Machine Learning and Knowledge Extraction*, 2(1):37–55, 2020. doi: 10.3390/make2010004.

Qiang Liu, Mengyu Chu, and Nils Thuerey. ConFIG: Towards conflict-free training of physics informed neural networks. In *13th ICLR*, 2025a. doi: 10.48550/arXiv.2408.11104.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov-Arnold networks. In *13th ICLR*, 2025b. doi: 10.48550/arXiv.2404.19756.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63:208–228, 2021a. doi: 10.1137/19M1274067.

Lu Lu, Raphaël Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G. Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021b. doi: 10.1137/21M1397908.

Kevin Stanley McFall and James Robert Mahan. Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. *IEEE Transactions on Neural Networks*, 20(8):1221–1233, 2009. doi: 10.1109/TNN.2009.2020735.

Paolo Mereghetti, Daria Kokh, J. Andrew McCammon, and Rebecca C. Wade. Diffusion and association processes in biological systems: theory, computation and experiment. *BMC Biophysics*, 4 (2), 2011. doi: 10.1186/2046-1682-4-2.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Mark A. Pinsky. *Introduction to Fourier Analysis and Wavelets*. Graduate studies in mathematics 102. American Mathematical Society, 2008.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Maithili Sharan and Sundararaman Gopalakrishnan. Mathematical modeling of diffusion and transport of pollutants in the atmospheric boundary layer. *Pure Appl. Geophys.*, 160:357–394, 2003. doi: 10.1007/s00024-003-8784-5.

R. Shekari Beidokhti and Alaeddin Malek. Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques. *Journal of the Franklin Institute*, 346(9):898–913, 2009. doi: 10.1016/j.jfranklin.2009.05.003.

Khemraj Shukla, Juan Diego Toscano, Zhicheng Wang, Zongren Zou, and George Em Karniadakis. A comprehensive and fair comparison between MLP and KAN representations for differential equations and operator networks. *Computer Methods in Applied Mechanics and Engineering*, 431:117290, 2024. doi: 10.1016/j.cma.2024.117290.

Natarajan Sukumar and Ankit Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022. doi: 10.1016/j.cma.2021.114333.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. doi: 10.48550/arXiv.2006.10739.

Juan Diego Toscano, Vivek Oommen, Alan John Varghese, Zongren Zou, Nazanin Ahmadi Daryakenari, Chenxi Wu, and George Em Karniadakis. From PINNs to PIKANs: Recent advances in physics-informed machine learning. *Mach. Learn. Comput. Sci. Eng.*, 1, 2025. doi: 10.1007/s44379-025-00015-1.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021. doi: 10.1016/j.cma.2021.113938.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022. doi: 10.1016/j.jcp.2021.110768.

Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. An expert's guide to training physics-informed neural networks. *CoRR*, 2023. doi: 10.48550/arXiv.2308.08468.

Marius Zeinhofer, Rami Masri, and Kent–André Mardal. A unified framework for the error analysis of physics-informed neural networks. *IMA Journal of Numerical Analysis*, pp. drae081, 2024. doi: 10.1093/imanum/drae081.

A THEORETICAL BACKGROUND ON THE ONE-DIMENSIONAL DIFFUSION PROBLEM

In this appendix, we provide the analytical formula of the solution of the diffusion problem stated in equations 1–3. Applying basic separation-of-variables and superposition techniques (Evans, 2010) yields that the unique solution u of the system is of the form

$$u(x,t) = \frac{a_0}{2} + \sum_{j=1}^{\infty} a_j \exp\left(-D\pi^2 j^2 t\right) \cos\left(\pi j x\right), \qquad x,t \in [0,1], \tag{9}$$

where $(a_j)_{j\in\mathbb{N}_0} \subset \mathbb{R}$ are the Fourier coefficients of the initial condition g:

$$a_j = 2\int_0^1 g(x) \cos\left(\pi j x\right) \, dx, \qquad j \in \mathbb{N}_0. \tag{10}$$

More details on how the limit in equation 9 can be interpreted under which assumptions on g can be found in (Evans, 2010; Pinsky, 2008). In the case of sufficiently smooth initial data g satisfying the vanishing Neumann boundary conditions (equation 2), i.e., $g'(0) = 0 = g'(1)$, the infinite sum in equation 9 converges uniformly on $[0,1]^2$ (Pinsky, 2008).

In the case where g consists of a single spatial frequency, all Fourier coefficients a_j except of one vanish. Concretely, if $g(x) = \cos(\pi n x)$ for some $n \in \mathbb{N}$, evaluating the integrals in equation 10 yields $a_n = 1$ and $a_j = 0$ for $j \neq n$. This is because $(\cos(\pi m \cdot))_{m\in\mathbb{N}_0}$ is an orthogonal system of $L^2([0,1])$ (Pinsky, 2008). In this case, the solution of the diffusion problem given by equations 1–3 is therefore of the form

$$u(x,t) = \exp\left(-D\pi^2 n^2 t\right) \cos\left(\pi n x\right), \qquad x,t \in [0,1]. \tag{11}$$

This formula gives the explicit solutions in the low frequency and high frequency cases described in Table 1 and, due to the linearity of the diffusion problem, also in the multiscale case.

In the case of simple initial data g, the Fourier coefficients can be computed explicitly. For instance, if g is a polynomial, this is possible by integrating by parts in the integrals from equation 10. In this way, one can obtain the explicit solution formula for the two polynomials from Table 1. Notice, however, that in this case, the sum in equation 9 is indeed infinite. For the numerics, we thus have to truncate the sum, i.e., we approximate equation 9 by

$$u(x,t) \approx \frac{a_0}{2} + \sum_{j=1}^{N} a_j \exp\left(-D\pi^2 j^2 t\right) \cos\left(\pi j x\right), \qquad x,t \in [0,1], \tag{12}$$

for some sufficiently large $N \in \mathbb{N}$. For the numerical experiments conducted in section 4, we used $N = 200$ terms.

B METHODOLOGY – MORE GENERAL CASES

In section 3, the formulae for hard-constraining Neumann boundary conditions are restricted to the case of one spatial variable x in the domain $[0,1]$ and boundary conditions at both parts of the boundary. We now show that the same idea can be used to hard-constrain Neumann boundary conditions on general intervals, only on one part of the boundary, and in higher dimensional spatial domains.

In all of the below, we again consider the case where the solution has an additional time dependency, although this is not necessary.

B.1 General intervals

We first consider Neumann boundary conditions on a general interval $[\alpha, \beta]$, i.e.,

$$\partial_x u(\alpha, t) = A, \qquad \partial_x u(\beta, t) = B, \tag{13}$$

for a sought function $u\colon [\alpha, \beta] \times [0, T] \to \mathbb{R}$ and prescribed $\alpha < \beta$, $T > 0$, and $A, B \in \mathbb{R}$. Equation 13 can be hard-constrained by suitably rescaling the functions from section 3. Concretely, the Fourier embeddings defined in equation 5 and equation 6 need to be replaced by

$$[\alpha, \beta] \ni x \mapsto \gamma_1(x) = \cos\left(\pi \frac{x - \alpha}{\beta - \alpha}\right) \tag{14}$$

and

$$[\alpha, \beta] \ni x \mapsto \gamma_{1,b_2,\dots,b_n}(x) = \left(\cos\left(\pi \frac{x - \alpha}{\beta - \alpha}\right), \cos\left(\pi b_2 \frac{x - \alpha}{\beta - \alpha}\right), \dots, \cos\left(\pi b_n \frac{x - \alpha}{\beta - \alpha}\right)\right), \tag{15}$$

respectively, where $b_2, \dots, b_n \in \mathbb{Z}$ are again (randomly sampled) integer frequencies. The output transformation from equation 8 adjusting for non-vanishing Neumann boundary data becomes

$$u^{\text{trafo}}(x, t) = u^{\text{NN}}\big(\gamma(x), t\big) + (x - \alpha)(\beta - x)^2 A + (x - \alpha)^2 (x - \beta) B. \tag{16}$$

where γ can be γ_1 or $\gamma_{1,b_2,\dots,b_n}$. It is straight-forward to verify via the chain rule that u^{trafo} defined by equation 16 indeed satisfies equation 13 for any smooth representation model u^{NN}.

B.2 One-sided Neumann boundary condition

We next consider the situation where only a Neumann boundary condition at one side of the boundary is hard-constrained, i.e.,

$$\partial_x u(\alpha, t) = A, \tag{17}$$

for a sought function $u\colon [\alpha, \beta] \times [0, T] \to \mathbb{R}$ and prescribed $\alpha < \beta$, $T > 0$, and $A \in \mathbb{R}$. In equation 17, the Neumann boundary condition is imposed at the left side of the spatial interval $[\alpha, \beta]$; straightforward modifications of the formulae below allow to treat the right boundary in the same way. To hard-constrain equation 17, we use the feature transformation

$$[\alpha, \beta] \ni x \mapsto \gamma_1(x) = \cos\left(\frac{\pi}{2} \frac{x - \alpha}{\beta - \alpha}\right). \tag{18}$$

This mapping is a suitable substitute for equation 5 because $\gamma_1'(\alpha) = 0$ and $\gamma_1'(x) \neq 0$ for $\alpha < x \leq \beta$. The latter is needed for the transformed model to be able to learn general derivative values on $]\alpha, \beta]$, in particular, at the right boundary $x = \beta$. To better capture high-frequency phenomena, the above transformation can be extended to

$$[\alpha, \beta] \ni x \mapsto \gamma_{1,b_2,\dots,b_n}(x) = \left(\cos\left(\frac{\pi}{2} \frac{x - \alpha}{\beta - \alpha}\right), \cos\left(\frac{\pi}{2} b_2 \frac{x - \alpha}{\beta - \alpha}\right), \dots, \cos\left(\frac{\pi}{2} b_n \frac{x - \alpha}{\beta - \alpha}\right)\right). \tag{19}$$

In this case, the (randomly sampled) frequencies $b_2, \dots, b_n$ can be arbitrary real numbers (not necessarily integers). The key property $\frac{\partial}{\partial x}\gamma_{1,b_2,\dots,b_n}(\alpha) = 0$ is ensured by using only Cosine mappings and no Sines. The overall transformation to hard-constrain equation 17 for a arbitrary smooth representation model u^{NN} takes on the form

$$u^{\text{trafo}}(x, t) = u^{\text{NN}}\big(\gamma(x), t\big) + (x - \alpha) A, \tag{20}$$

where γ can be defined by equation 18 or equation 19.

B.3 Neumann boundary condition in higher dimensions

Lastly, we show how the newly proposed method to hard-constrain Neumann boundary conditions can be applied in the case where the spatial domain is a hyperrectangle $[\alpha_1, \beta_1] \times \ldots \times [\alpha_d, \beta_d]$ of dimension $d \in \mathbb{N}$ for prescribed $\alpha = (\alpha_1, \dots, \alpha_d), b = (\beta_1, \dots, \beta_d) \in \mathbb{R}^d$ with $\alpha_i < \beta_i$ for

$i = 1, \dots, d$. We consider the case where Neumann boundary conditions are imposed at all parts of the boundary, i.e.,

$$\partial_{x_i} u(x_1, \dots, x_{i-1}, \alpha_i, x_{i+1}, \dots, x_d, t) = A_i, \qquad i = 1, \dots, d, \tag{21}$$

$$\partial_{x_i} u(x_1, \dots, x_{i-1}, \beta_i, x_{i+1}, \dots, x_d, t) = B_i, \qquad i = 1, \dots, d, \tag{22}$$

where $A_1, \dots, A_d, B_1, \dots, B_d \in \mathbb{R}$ are given and $u\colon [\alpha_1, \beta_1] \times \ldots \times [\alpha_d, \beta_d] \times [0, T] \to \mathbb{R}$ denotes the sought solution. To hard-constrain equations 21–22, the transformations from equations 14–15 need to be applied to all input dimensions. Concretely, let

$$[\alpha_1, \beta_1] \times \ldots \times [\alpha_d, \beta_d] \ni x \mapsto \gamma_1(x) = \left(\cos\left(\pi \frac{x_1 - \alpha_1}{\beta_1 - \alpha_1}\right), \dots, \cos\left(\pi \frac{x_d - \alpha_d}{\beta_d - \alpha_d}\right)\right) \tag{23}$$

and

$$\begin{aligned} &[\alpha_1, \beta_1] \times \ldots \times [\alpha_d, \beta_d] \ni x \mapsto \gamma_{1,b_2,\dots,b_n}(x) = \\ &\qquad = \left(\cos\left(\pi \frac{x - \alpha}{\beta - \alpha}\right), \cos\left(\pi b_2 \frac{x - \alpha}{\beta - \alpha}\right), \dots, \cos\left(\pi b_n \frac{x - \alpha}{\beta - \alpha}\right)\right). \end{aligned} \tag{24}$$

In the latter expression, the frequencies $b_2, \dots, b_n$ are again required to be integers and all vector-operations are meant in a component-wise way, so that the output of $\gamma_{1,b_2,\dots,b_n}$ is of dimension $n \cdot d$. The overall transformation to hard-constrain equations 21–22 is

$$u^{\text{trafo}}(x, t) = u^{\text{NN}}\big(\gamma(x), t\big) + \sum_{i=1}^{d} \left(A_i (x_i - \alpha_i) \prod_{\substack{j=1,\\ j \neq i}}^{d} (\beta_j - x_j)^2 + B_i (x_i - \beta_i) \prod_{\substack{j=1,\\ j \neq i}}^{d} (x_j - \alpha_j)^2 \right) \tag{25}$$

for $x = (x_1, \dots, x_n) \in [\alpha_1, \beta_1] \times \ldots \times [\alpha_d, \beta_d]$ and $0 \leq t \leq T$, where γ can be defined by equation 23 or equation 24 and u^{NN} is an arbitrary smooth representation model.

In order to not hard-constrain Neumann boundary conditions for certain spatial input variables, the feature transformations from equations 23–24 can simply be modified so that they leave the corresponding variables unchanged. Hard-constraining Neumann boundary conditions only at one part of the boundary in certain spatial input dimension can be achieved by modifications similar to those presented in section B.2.

C More details on the experiments

The aim of this appendix is to provide further details on the experiments conducted and mentioned in section 4.

Firstly, we specify the hyperparameters used for all numerical experiments in this study. As mentioned in section 4, these hyperparameters are chosen similarly to those used in Wang et al. (2021). The representation model is always a fully connected neural network with 3 hidden layers consisting of 100 neurons each. The activation function is $\tanh$. The models are trained with a fixed learning rate of 10^{-4} using the Adam optimizer. The numbers of collocation points used to compute the loss $\mathcal{L}$ are $N_{\text{PDE}} = 2 \cdot 10^4$, $N_{\text{IC}} = 500$, and $N_{\text{BC}} = 10^3$. In all settings, the accuracy is computed for the model with the lowest (training) loss encountered during training. The code is written in DeepXDE (Lu et al., 2021a) with backend PyTorch (Paszke et al., 2019). All experiments were carried out on an NVIDIA Quadro RTX 5000 (16GB RAM).

Secondly, we provide more details on the experiments visualized in Figure 2. Concretely, raw data for the accuracies (relative L^2-error w.r.t. the analytical reference solution) of all models from Figure 2 are given in Table 2.

Recall that for the simulations visualized in Figure 2, the training time of all models is fixed to ensure a fair comparison across the different methods. As discussed in section 4, this is the main reason for the consistently inferior performance of the existing hard-constraining method because of the longer time it takes to train. To underpin this statement, all models from section 4 have also been trained for the same number of 10^6 iterations. The results of these modified experiments are given below. Concretely, Figure 3 and Table 3 are the analogues of Figure 2 and Table 2, respectively, in

Table 2: Accuracies for the experiments visualized in Figure 2 in the settings given by in Table 1. In each setting, the data corresponding to the reference method (which is always chosen as the vanilla PINN with the Fourier embedding strategy leading to the best accuracy) is shown in italic font. The data corresponding to the best accuracy is highlighted in bold font.

	Low frequency	High frequency	Multiscale	Polynom 3rd order	Polynom 4th order
Vanilla PINN	*$4.14 \cdot 10^{-5}$*	$1.64 \cdot 10^{-1}$	$3.80 \cdot 10^{-2}$	*$1.30 \cdot 10^{-4}$*	*$1.61 \cdot 10^{-4}$*
Vanilla PINN w/ Fourier 20	$9.54 \cdot 10^{-5}$	$5.49 \cdot 10^{-3}$	$4.22 \cdot 10^{-4}$	$3.85 \cdot 10^{-4}$	$5.66 \cdot 10^{-4}$
Vanilla PINN w/ Fourier 50	$8.34 \cdot 10^{-4}$	*$3.69 \cdot 10^{-3}$*	*$2.40 \cdot 10^{-4}$*	$3.66 \cdot 10^{-3}$	$4.26 \cdot 10^{-3}$
Existing HC	$6.58 \cdot 10^{-5}$	$4.50 \cdot 10^{-1}$	$6.57 \cdot 10^{-2}$	$1.40 \cdot 10^{-4}$	$3.39 \cdot 10^{-4}$
Existing HC w/ Fourier 20	$7.33 \cdot 10^{-4}$	$8.08 \cdot 10^{-3}$	$9.86 \cdot 10^{-4}$	$4.13 \cdot 10^{-3}$	$3.99 \cdot 10^{-3}$
Existing HC w/ Fourier 50	$2.80 \cdot 10^{-2}$	$8.42 \cdot 10^{-3}$	$3.34 \cdot 10^{-4}$	$1.22 \cdot 10^{-1}$	$1.92 \cdot 10^{-1}$
New HC via 1 Fourier frequency	$\mathbf{1.90 \cdot 10^{-5}}$	$3.73 \cdot 10^{-1}$	$1.15 \cdot 10^{-2}$	$\mathbf{6.85 \cdot 10^{-5}}$	$\mathbf{1.20 \cdot 10^{-4}}$
New HC via 20 Fourier frequencies	$8.01 \cdot 10^{-4}$	$1.61 \cdot 10^{-4}$	$1.18 \cdot 10^{-4}$	$3.13 \cdot 10^{-3}$	$3.55 \cdot 10^{-3}$
New HC via 50 Fourier frequencies	$9.43 \cdot 10^{-3}$	$\mathbf{1.30 \cdot 10^{-4}}$	$\mathbf{6.05 \cdot 10^{-5}}$	$1.55 \cdot 10^{-1}$	$2.02 \cdot 10^{-1}$

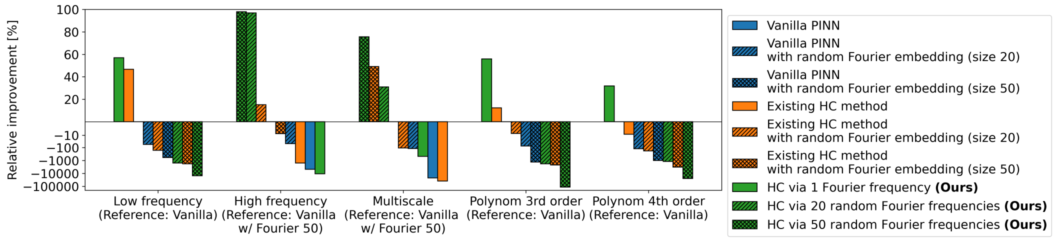

Figure 3: Relative improvements of the accuracies of PINNs with different hard-constraining (HC) and Fourier embedding strategies w.r.t. a reference PINN in the settings described in Table 1. Different to Figure 2, all models are trained for the same number of iterations (10^6 training iterations) here.

the case where all models are trained for a fixed number of iterations. As discussed in section 4, it can be seen that the models based on the existing hard-constraining technique indeed perform better than the vanilla PINN in most of the cases when trained for the same number of iterations. Nonetheless, in all of the settings described in Table 1, the best performances can always be observed when using the newly proposed method of hard-constraining the Neumann boundary conditions via Fourier embeddings.

Table 3: Accuracies for the experiments visualized in Figure 3 in the settings given by in Table 1. Different to Table 2, all models have been trained for the same number of iterations (10^6 training iterations) here. In each setting, the data corresponding to the reference method (which is always chosen as the vanilla PINN with the Fourier embedding strategy leading to the best accuracy) is shown in italic font. The data corresponding to the best accuracy is highlighted in bold font.

	Low frequency	High frequency	Multiscale	Polynom 3rd order	Polynom 4th order
Vanilla PINN	*$4.14 \cdot 10^{-5}$*	$1.64 \cdot 10^{-1}$	$3.80 \cdot 10^{-2}$	*$1.30 \cdot 10^{-4}$*	*$1.61 \cdot 10^{-4}$*
Vanilla PINN w/ Fourier 20	$6.51 \cdot 10^{-5}$	$5.00 \cdot 10^{-3}$	$3.74 \cdot 10^{-4}$	$2.25 \cdot 10^{-4}$	$3.57 \cdot 10^{-4}$
Vanilla PINN w/ Fourier 50	$2.82 \cdot 10^{-4}$	*$3.37 \cdot 10^{-3}$*	*$1.72 \cdot 10^{-4}$*	$1.83 \cdot 10^{-3}$	$1.75 \cdot 10^{-3}$
Existing HC	$2.21 \cdot 10^{-5}$	$5.60 \cdot 10^{-2}$	$6.59 \cdot 10^{-2}$	$1.14 \cdot 10^{-4}$	$1.75 \cdot 10^{-4}$
Existing HC w/ Fourier 20	$1.09 \cdot 10^{-4}$	$2.86 \cdot 10^{-3}$	$3.54 \cdot 10^{-4}$	$1.39 \cdot 10^{-4}$	$4.55 \cdot 10^{-4}$
Existing HC w/ Fourier 50	$7.91 \cdot 10^{-4}$	$3.62 \cdot 10^{-3}$	$8.75 \cdot 10^{-5}$	$3.04 \cdot 10^{-3}$	$5.44 \cdot 10^{-3}$
New HC via 1 Fourier frequency	**$1.78 \cdot 10^{-5}$**	$3.64 \cdot 10^{-1}$	$1.01 \cdot 10^{-3}$	**$5.71 \cdot 10^{-5}$**	**$1.10 \cdot 10^{-4}$**
New HC via 20 Fourier frequencies	$6.81 \cdot 10^{-4}$	$1.06 \cdot 10^{-4}$	$1.19 \cdot 10^{-4}$	$2.47 \cdot 10^{-3}$	$2.05 \cdot 10^{-3}$
New HC via 50 Fourier frequencies	$6.28 \cdot 10^{-3}$	**$7.23 \cdot 10^{-5}$**	**$4.18 \cdot 10^{-5}$**	$1.47 \cdot 10^{-1}$	$4.00 \cdot 10^{-2}$