École Polytechnique Fédérale de Lausanne
Chair of Computational Mathematics and Simulation Science

A Physics-Informed Neural Network For The Helmholtz Impedance Problem

May 20, 2022

Author: Sepehr Mousavi

Supervisors: Jan S. Hesthaven
Fernando José Henriquez Barraza

Contents

Abstract

In this work, we propose to use the Variational Physics Informed Neural Networks for the solution of the Helmholtz equation equipped with impedance boundary conditions in a high-frequency regime. We provide a detail theoretical analysis, together with extensive computational experiment supporting our claims.

1 Introduction

The Helmholtz equation has the general form

$$-\nabla^2 u(\underline{x}) - k^2 u(\underline{x}) = f(\underline{x}), \qquad \underline{x} \in \Omega \tag{1}$$

and represents a time-independent form of the wave equation after applying the technique of separation of variables. This equation often arises in various physical problems including the study of electromagnetic radiation, seismology, and acoustics.

In this study, we will consider the 1-dimensional Helmholtz equation with impedance boundary conditions in the domain $\Omega = [a, b]$

$$\begin{aligned} -u''(x) - k^2 u(x) &= f(x) \\ -u'(a) - \imath k^2 u(a) &= g_a \in \mathbb{C} \\ +u'(b) - \imath k^2 u(b) &= g_b \in \mathbb{C} \end{aligned} \tag{2}$$

where k is often called the frequency of the equation, as larger values of this parameter result in more oscillating solutions. We will mainly consider Equation 2 in the domain $[-1, +1]$.

With the recent improvements in knowledge and practical aspects of machine learning techniques, and more specifically, deep learning and neural networks, these methods are being successfully implemented in domains and applications such as computer vision, recommender systems, generative models, etc., where they can benefit from the abundance of data and learn the features that best represent the problem. However, in the context of analyzing complex physical, biological, or engineering systems, these methods are facing the challenge of high cost of data acquisition, which forces us to find a way to make decisions in an semi-supervised fashion while retaining the robustness and convergence of the methods. Recent studies in the literature have introduced deep learning frameworks for solving nonlinear partial differential euqations (PDEs) that have achieved this goal.

Raissi *et. al.* [1] have introduced the physics-informed neural networks (PINNs) scheme, which uses the prior knowledge that usually stems from the physics of the system in a structured way as a regularization method to narrow the range of the admissible solutions. They consider the general form of a nonlinear PDE as

$$u_t + N[u; \lambda] = 0, x \in \Omega, t \in [0, T] \tag{3}$$

where $u(t, x)$ denotes the unknown solution, $N[.; \lambda]$ is a nonlinear operator parameterized by λ, and Ω is a subset which defines the domain of the system. With $f(t, x)$ as the left-hand-side of this equation, the method consists of defining two neural networks for $u(t, x)$ and $f(t, x)$

with shared parameters, and optimizing these parameters based on a suitable loss function that takes into account the initial condition, boundary conditions, and satisfaction of the equation in quadrature points of the domain. They have shown that their method performs well even with small data for several nonlinear PDEs by comparing their results with the exact solution of those equations.

Kharazmi *et. al.* [2] have taken the next step by developing a Petrov-Galerkin version of the PINNs by selecting the trial space to be the space of the neural networks and the test space to be the space of trigonometric functions or Legendre polynomials. They introduce the variational physics-informed neural networks (VPINNs) by incorporating the variational residual of the PDE in the loss function of the network. They show that by integrating by parts the integrand part of the variational form, they can reduce the training time of the VPINNs and increase their accuracy compared to PINNs. They obtain the explicit form of the variational residual for shallow neural networks with one hidden layer, and suggest that numerical integration techniques should be employed for deep neural networks.

The goal of this study is to implement the VPINNs scheme introduced in [2] for solving Equation 2 and to explore the behavior of these networks with different structures, activation functions, etc. and for different parameters of the equation. For this purpose, we will present the variational formulation of Equation 2 in section 2. Then, we will proceed with presenting the frameworks of a finite elements scheme and a VPINN scheme in subsection 2.2 and subsection 2.3, respectively. In section 3, we will present and discuss the results of the implemented frameworks with various parameters and conditions Section 4 summerizes our conclusions and provides some suggestions for future works.

2 Frameworks And Methods

** empty **

2.1 Variational formulation of the Helmholtz problem

We take the 1-D version of the Helmholtz equation as expressed in Equation 2, and test this equation by a arbitrary smooth test function $v \in H^1_{0\,(\Omega)}$ with compact support in the domain, to get

$$\int_\Omega (-u'' - k^2 u)v = \int_\Omega fv \qquad \forall v \in H^1_{0\,(\Omega)}. \tag{4}$$

Integration by parts gives $\int_a^b u''v = [u'v]_a^b - \int_a^b u'v'$ which could be replaced into Equation 4 to eliminate the second derivative from the equation. Doing this, and by setting $u'(a)v(a) = (-g_a - \imath k u(a))v(a)$ and $u'(b)v(b) = (g_b + \imath k u(b))v(b)$, we get the variational formulation of the Helmholtz impedance problem, which is to find $u \in H^1{}_{(\Omega)}$ such that $a(u,\, v) = b(v)$ for all $v \in H^1_{0\,(\Omega)}$, where $a(u,\, v)$ and $b(v)$ are defind as

$$a(u,\, v) = \int_a^b u'v' - k^2 \int_a^b uv - \imath k(u(a)v(a) + u(b)v(b)) \tag{5}$$

$$b(v) = \int_a^b fv + g_a v(a) + g_b v(b) \tag{6}$$

2.2 Framework of the finite elements scheme

We descritize the domain $\Omega = [a,\ b]$ by defining $N+1$ basis functions as

$$\varphi_j(x) = \begin{cases} 1 - \frac{N}{2}|x - x_j|, & x_{j-1} < x < x_{j+1} \\ 0, & otherwise. \end{cases} \tag{7}$$

for $j = 0, \cdots, N+1$. The resulting basis functions with $N = 9$ are illustrated in Figure 1. Let $V_N = span(\varphi_j)_{j=0,\cdots,N+1} \subset H^1$ be a finite-dimensional subset of H^1. We search for solutions in this subspace, and modify the variational version of the Helmholtz impedance problem as finding $u_N \in V_N$ such that $a(u_N,\ v_N) = b(v_N)$ for all $v_N \in V_N$, where $a(u,\ v)$ and $b(v)$ are defind in Equations 5 and 6, respectively. We can easily see that it is actually sufficient to ensure that the aforementioned equation is satisfied for all the basis functions φ_i, $i = 0, \cdots, N+1$:

$$a(u_N,\ \varphi_i) = b(\varphi_i) \tag{8}$$

Let $u_N \in V_N$ be a linear combination of the basis of V_N, $u_N(x) = \sum_{j=0}^{N} c_j \varphi_j$. By plugging u_N into Equation 8, we can verify that the stated problem could be expressed as a linear system of equations $A\underline{c} = \underline{f}$, where

$$A = \begin{bmatrix} a(\varphi_0,\ \varphi_0) & a(\varphi_1,\ \varphi_0) & \cdots & a(\varphi_N,\ \varphi_0) \\ a(\varphi_0,\ \varphi_1) & a(\varphi_1,\ \varphi_1) & \cdots & a(\varphi_N,\ \varphi_1) \\ \vdots & \vdots & \ddots & \vdots \\ a(\varphi_N,\ \varphi_N) & a(\varphi_0,\ \varphi_N) & \cdots & a(\varphi_N,\ \varphi_N) \end{bmatrix}, \quad \underline{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_N \end{bmatrix}, \quad \underline{f} = \begin{bmatrix} b(\varphi_0) \\ b(\varphi_1) \\ \vdots \\ b(\varphi_N) \end{bmatrix}. \tag{9}$$

The solution of this system of equations could be calculated using methods such as Gauss-elimination. We don't need any quadrature rule to calculate the integrals in $a(\varphi_j,\ \varphi_i)$ as these result in determined values depending on i and j.

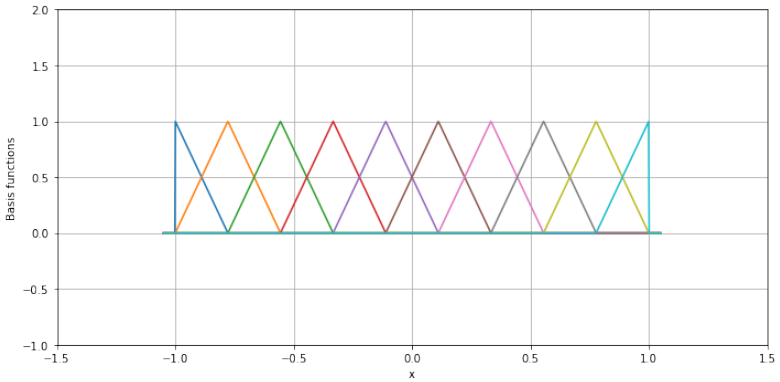


Figure 1: The finite element basis functions over the domain $\Omega = [a,\ b]$ for $N = 9$.

2.3 Framework and structure of the VPINNs scheme

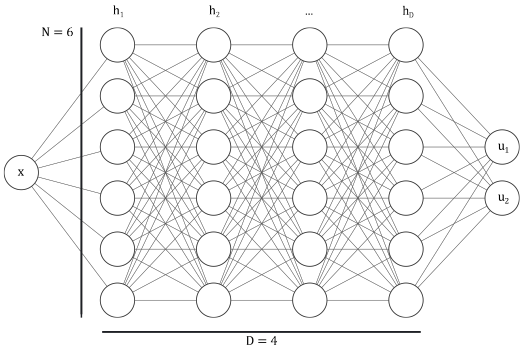
2.3.1 Multilayer perceptron

Let $\mathcal{N}$ be the space of a fully-connected multilayer perceptron (MLP) with 1 node at the input layers, N nodes on each of D hidden layers, and 2 nodes at the output layer as depicted in

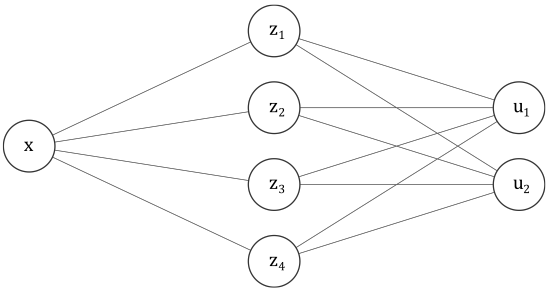
Figure 2a. The 2 dimensional output represents the real and the imaginary part of the solution $u_N(x) = u_{N,1}(x) + \imath u_{N,2}(x)$. In case of shallow networks, it could be easily shown that this will be the same as using complex model parameters.
For this generic structure of the MLP, each element of the the 2-dimensional solution could be expressed as

$$u_{N,i} = c_0^i + \sum_{j=1}^{N} c_j^i \sigma(h^D \circ h^{D-1} \circ \cdots \circ h^1(x)), \tag{10}$$

where $\sigma(x)$ is a non-linear activation function, $h_i^1 = \sigma(w_i^1 x + b_i^1)$, and $h_i^d = \sigma(\sum_{n=1}^{N} w_{i,n}^d h_n^{d-1} + b_i^d)$ for $d = 2, \cdots, D$.



(a) Generic mulilayer perceptron with 4 hidden layer and $N = 6$.



(b) Shallow multilayer perceptron with $N = 3$.

With this structure, the network has $2N$ parameters for the first hidden layer, $(D-1)(N^2+N)$ parameters for other hidden layers, and $2N+2$ parameters for the last year, which sums up as

$$N_p = (D-1)N^2 + (D+3)N + 2, \tag{11}$$

where N_p is total number of parameters of the network.

2.3.2 Basis representation

2.3.3 Variational formulation for VPINNs

Let $V_K = span(v_1, v_2, \cdots, v_K)$ be the space of the test functions. The helmholtz impedance problem in the context of VPINNs is defined as finding $u_{\mathcal{N}} \in \mathcal{N}$ such that the loss function

$$\mathcal{L}^2(u_{\mathcal{N}}) = \frac{1}{K} \sum_{k=1}^{K} |R_k^{(2)} - F_k|^2 \tag{12}$$

is minimized, where

$$R_k^{(2)} = \int_a^b u'_{\mathcal{N}} v'_k - k^2 \int_a^b u_{\mathcal{N}} v_k - (g_a + \imath k u_{\mathcal{N}}(a)) v_k(a) - (g_b + \imath k u_{\mathcal{N}}(b)) v_k(b), \tag{13}$$

$$F_k = \int_a^b f v_k. \tag{14}$$

2.3.4 Morawetz- and Rellich-Type Identities

Another variational formulation of the Helmholtz impedance problem is presented and analyzed in [3]. A modification of this formulation for the one dimensional case is provided in [4], which is presented in this subsection. Let $u, v \in L^2(\Omega)$ and consider $\Omega = [-1, +1]$. For $k, \beta \in \mathbb{R}$, set $\mathcal{L}v = v'' + k^2 v$ and define

$$\mathcal{M}v = xv' - \imath k\beta v. \tag{15}$$

Consider the sesquilinear form $a : H^2(\Omega) \times H^2(\Omega) \to \mathbb{C}$ defined as

$$\begin{aligned} a(u,v) := &\int_\Omega u'\bar{v}' + k^2 \int_\Omega u\bar{v} + \int_\Omega \left(\mathcal{M}u + \frac{A}{k^2}\mathcal{L}u\right)\overline{\mathcal{L}v} \\ &- \imath k(\overline{\mathcal{M}v(b)}u(b) + \overline{\mathcal{M}v(a)}u(a)) - \bar{v}'\mathcal{M}u|_{\partial\Omega} - xk^2 u\bar{v}|_{\partial\Omega} + xu'\bar{v}'|_{\partial\Omega} \end{aligned} \tag{16}$$

and the linear form $b : H^2(\Omega) \to \mathbb{C}$ defined as

$$b(v) := \int_\Omega \left(\overline{\mathcal{M}v} - \frac{A}{k^2}\overline{\mathcal{L}v}\right) f + \overline{\mathcal{M}v(b)}g_b + \overline{\mathcal{M}v(a)}g_a, \tag{17}$$

where $A > 0 \in \mathbb{R}$. The problem would be defined as finding $u \in H^2(\Omega)$ such that $a(u, v) = b(v)$ for all $v \in H^2(\Omega)$. As suggested in [4], setting $\beta = 1$ and $A = 1/3$ makes the sesquilinear form a V-elliptic.

2.3.5 Quadrature rules for integrations

3 Results And Discussion

The results of each method are presented and discussed in this section. We define the error of the solution as the $H^1(\Omega)$ norm of difference between the analytical solution $u(x)$, and the solution of the numerical scheme $u_N(x)$ as

$$||u(x) - u_N(x)||_{H^1(\Omega)} = ||u(x) - u_N(x)||_{L^2(\Omega)} + ||u'(x) - u'_N(x)||_{L^2(\Omega)}, \tag{18}$$

where

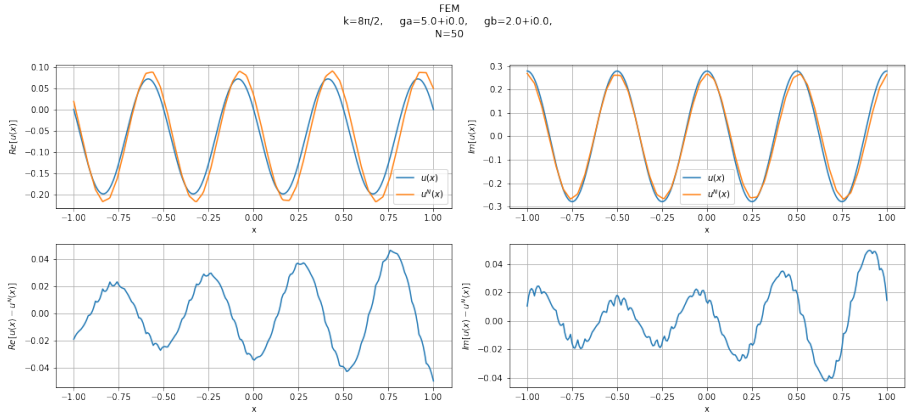
$$||f(x)||_{L^2(\Omega)} = \left(\int_\Omega |f(x)|^2 dx\right)^{1/2}, \tag{19}$$

and use this error as a measure to evaluate the solution from each numerical scheme.

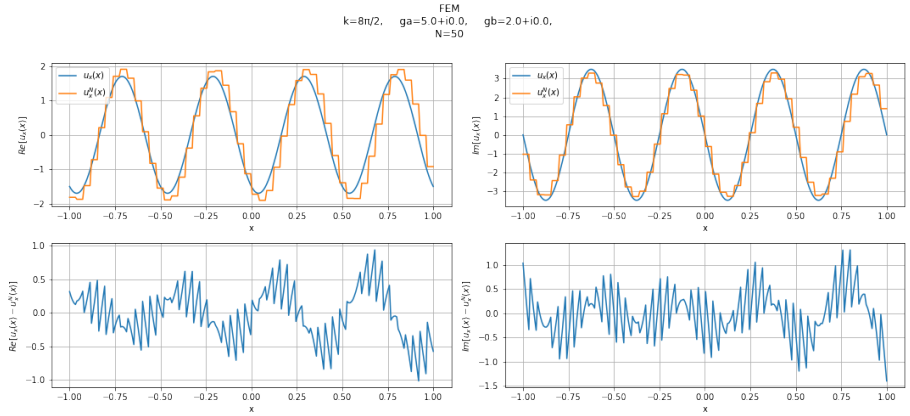
3.1 Finite elements scheme

Implementing the framework described in subsection 2.2, we investigated the results of the finite element scheme for the Helmholtz impedance problem with different frequencies. The results for a relatively moderate k validated against the exact solution are presented in Figures 3 and 4. These plots show how using a finer mesh (bigger N) on the domain improves the quality of the solution. The same trend has been observed for other k's.

However, for larger values of k, there is an issue with this method. From the nature of the equation we know that for larger k's, we have more oscillations in the solution. Since our FEM is estimating the solution with piecewise linear basis functions in a uniform mesh, the resulting



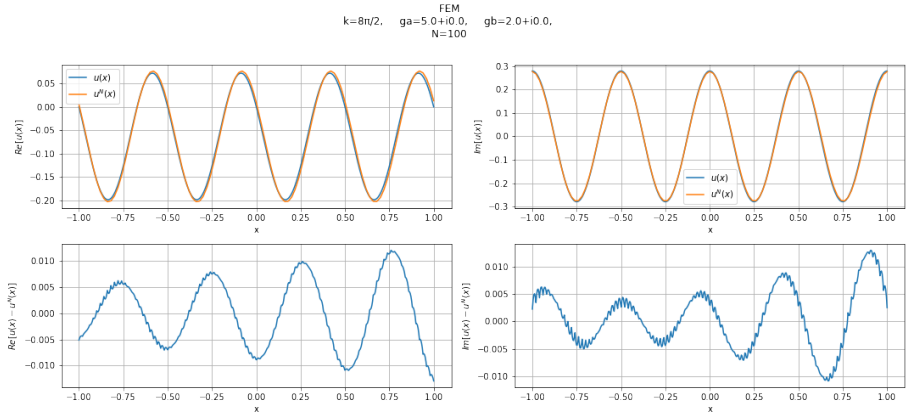
(a) $u_N(x)$



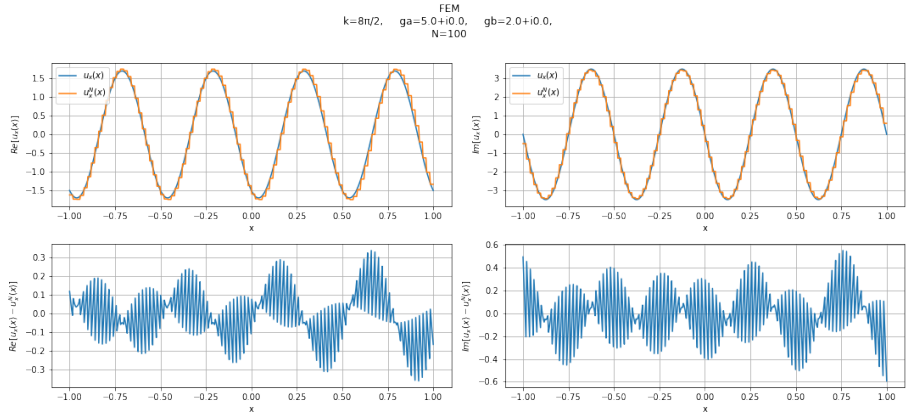
(b) $u'_N(x)$

Figure 3: $u_N(x)$ (a) and $u'_N(x)$ (b) plotted against the exact solution for a relatively moderate k and $N = 50$. On each subfigure, the bottom row represents the difference between the numerical and exact solutions. The source function has a constant value of $f(x) = 10$, and other parameters are indicated on the figures.

numerical solution will also be piecewise linear. This will require a minimum number of grid points on each oscillation for the estimation to have a decent accuracy. Consider the solution in Figure 4 where we used 100 grid points. If we wanted to estimate this function with 8 grid points, for instance, it was not possible to even capture the general shape of the function. This phenomenon is the major observation in Figure 5, where the H1-error of the numerical solution is plotted against mesh refinement parameter, N, for different values of k. For each constant k, there is no improvement in the error as we refine the mesh until a certain N, which will be called N_c in the rest of this report. However, for $N > N_c$, we can see the first order accuracy of the method as we expected. Another important observation is that the value of N_c increases with increasing k, which means that for high values of k, the improvement in the accuracy only begins at a very high N.



(a) $u_N(x)$



(b) $u'_N(x)$

Figure 4: $u_N(x)$ (a) and $u'_N(x)$ (b) plotted against the exact solution for a relatively moderate k and $N = 100$. On each subfigure, the bottom row represents the difference between the numerical and exact solutions. The source function has a constant value of $f(x) = 10$, and other parameters are indicated on the figures.

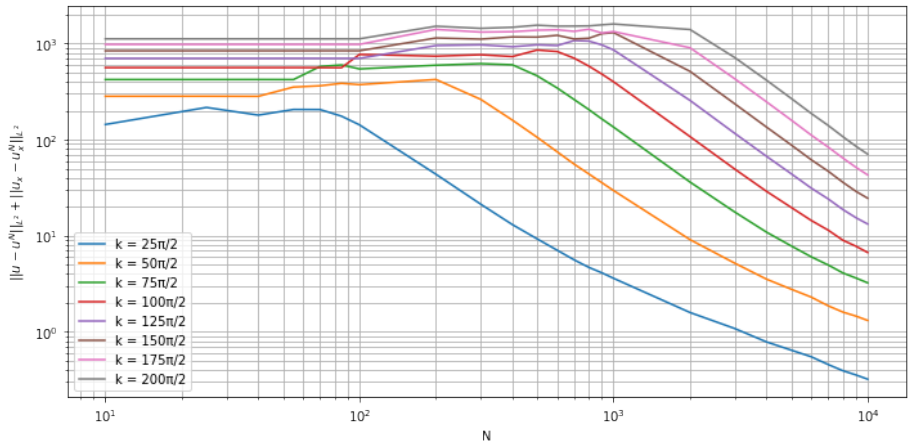


Figure 5: Order of accuracy of the FEM scheme for different values of k.

3.2 VPINNs Scheme

Implementing the framework explained in subsection 2.3, we investigated the performace of this scheme for different network structures, different activation functions, different training methods, and for different frequencies k. The Adam optimizer [5] is used for all trainings. The weights of each layer are initialized using the normal Xavier initialization [6], and the biases are initialized using a uniform distribution in the domain Ω. Each training is done multiple times to ensure independance from randomness and the best one is selected for comparisons. On each set of the trainings, the biggest possible learning rate lr that ensures convergence for all the networks being compared is selected.

3.2.1 Shallow Networks

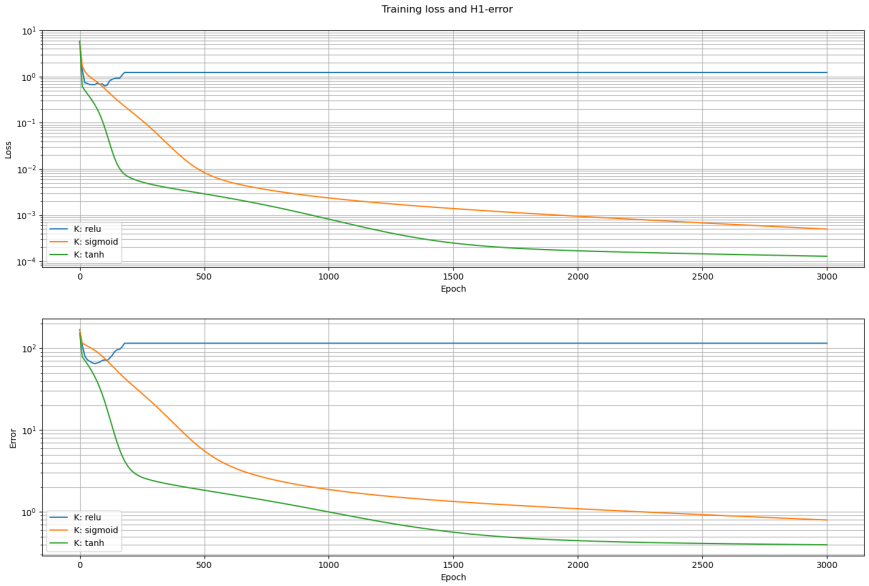
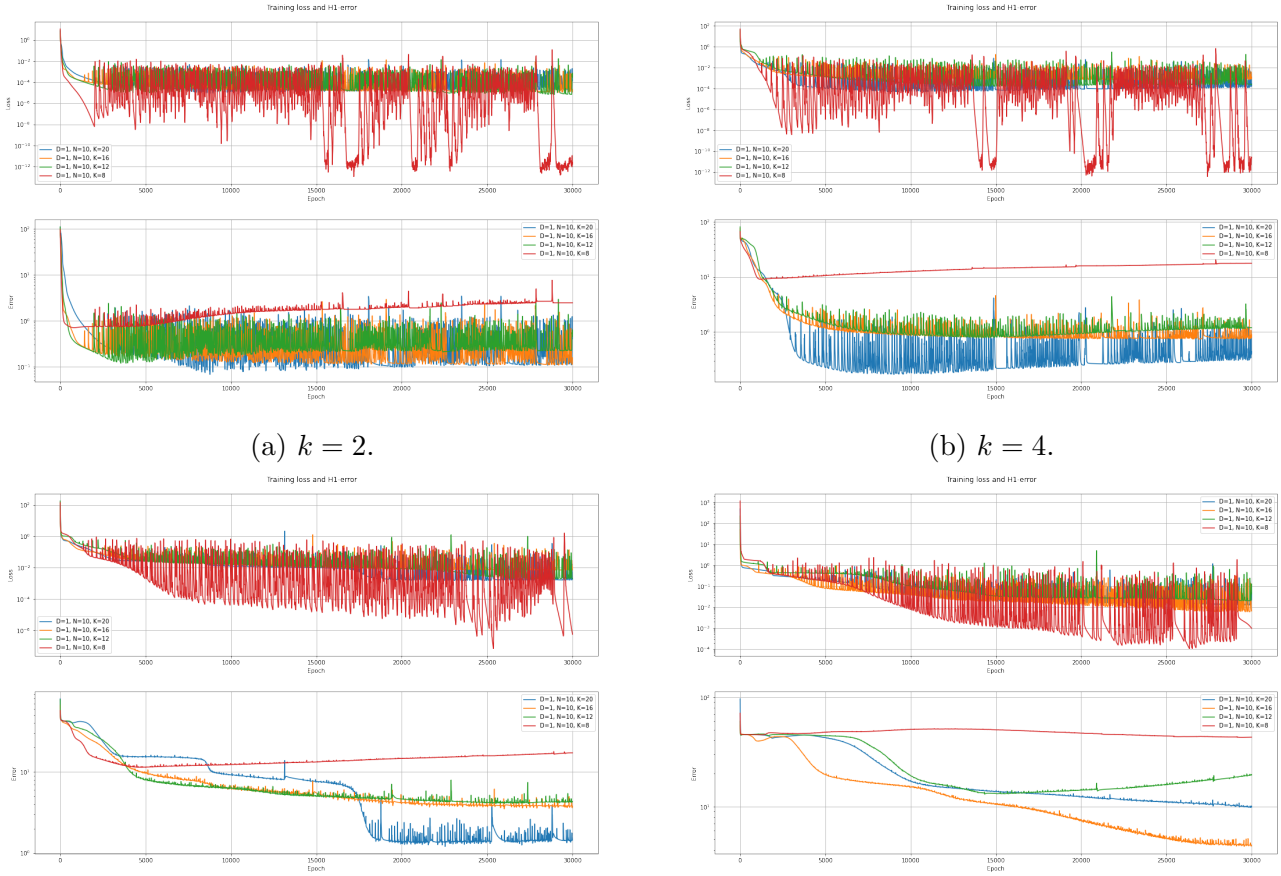


Figure 6: Training loss and H1-error of a shallow network with $N = 10$ and $K = 10$ with different activation functions for $k = 1.0$.

We start with shallow neural networks with only one hidden layer, i.e. $D = 1$. We use the local hat functions presented in Equation 7 as test functions, and train a network with $N = 10$ and 10 test functions ($K = 10$) with different types of activation functions for a low frequency. The loss functions and the errors are compared in Figure 6. The learning rate is $1E{-}01$ for all the activation functions. We can see that the hyperbolic tangent activation function outperforms the sigmoid activation function for our setup both in terms of accuracy and convergence rate. Regarding the ReLU activation function, there are some issues inherited with this set-up of networks. If we train the weights and biases of the hidden layer, these parameters drive the threshold of each ReLU outside of the support domain of the test functions, and the gradient of the loss with respect to these parameters go to zero, so technically, the hidden neurons will be lost one by one. As done in [7], it could be easily shown that in case of the ReLU activation function, the weights of the hidden layer are redundent for this kind of networks and it is suggested to set the weights to 1. To address the issue with ReLU activation function, the network is trained with fixed biases being uniformly distributed in the domain $[a, b]$ and weights fixed to 1, and the

training could be completed withough this issue. However, by fixing the parameters of the hidden layer and the ReLU activation function, this framework has no advantage over the original FEM scheme in case of capability and adaptability. In the rest of the experiments, the hyperbolic tangent activation function is used for the trainings.



(a) $k = 2$.

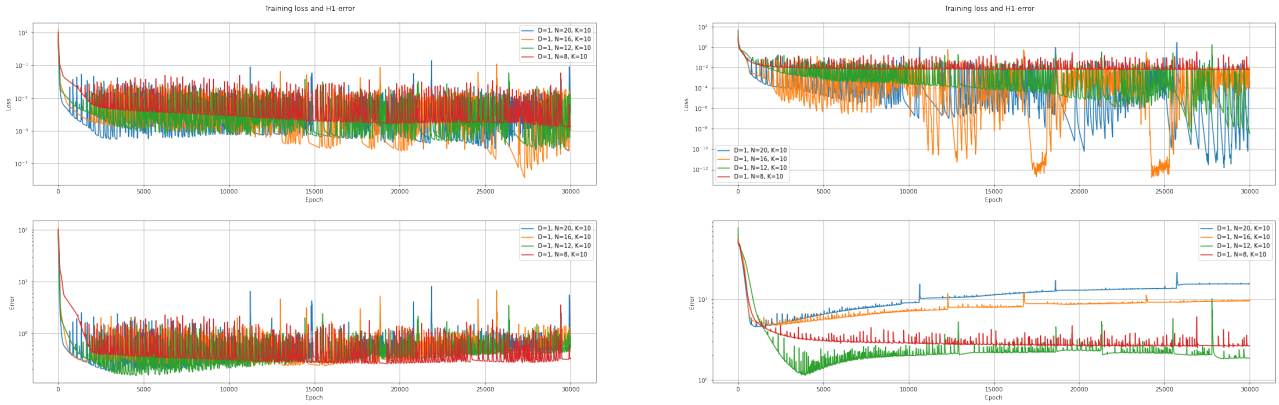
(b) $k = 4$.

(c) $k = 6$.

(d) $k = 8$.

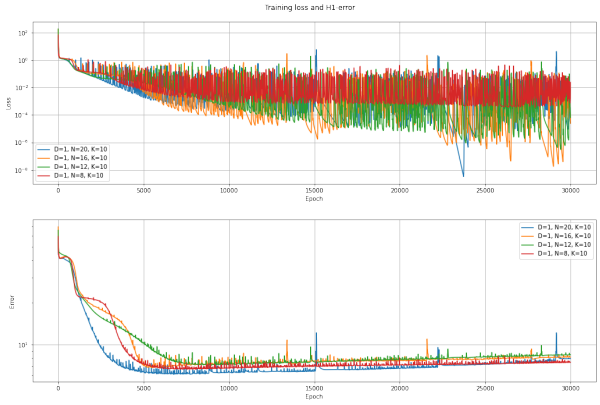
Figure 7: Training loss and H1-error with fixed network structure and increasing number of test functions for different frequencies.

In an effort to investigate the effect of increasing the number of test functions on the accuracy and the training of the network, with a fixed width $N = 10$, the training is done for different values of K. The results are presented in Figure 7. For the lowest frequency $k = 2$ in Figure 7a, we can see that with 8 test functions, the training loss becomes unstable and the H1-error takes an increasing trend after around 2000 epochs. With 12 test functions, this behavior is not observed and the H1-error converges. Increasing the number of test functions to 16, we can see a slight improvement, which does not repeat with further increasing the number of test functions to 20. For a higher frequency $k = 4$ in Figure 7b, we can see the same behavior for 8 test functions but with the difference that the final H1-error keeps being improved with increasing the number of test functions to 20. For the highest frequency $k = 8$, we can observe the unstable behavior extends to 12 test functions, and the improvement with increasing the number of test functions becomes more evident. We can conclude that as the frequency increases, more number of test functions will be needed for training the network.



(a) $k = 2$.

(b) $k = 4$.



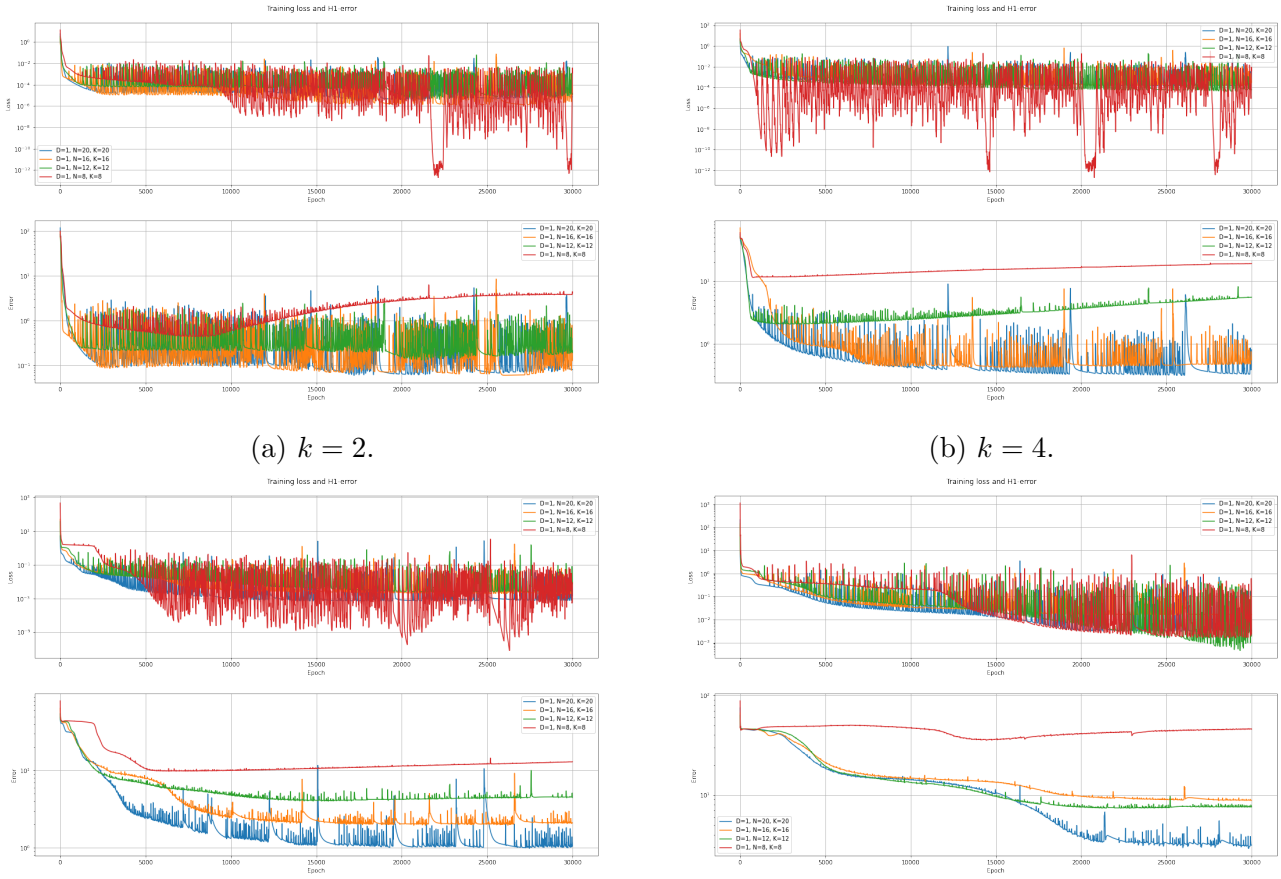
(c) $k = 6$.

(d) $k = 8$.

Figure 8: Training loss and H1-error with fixed number of test function and increasing width of the network.

To investigate the effect of the width of the network, a similar thing is done with fixing the number of test functions to $K = 10$, and training the network for different values of N. The results are presented in Figure 8. For the lowest frequency $k = 2$ in Figure 8a, we can see that increasing the width of the network does not bring much to the table. However, the network reaches its minimum H1-error faster. For $k = 4$ in Figure 8b, we can see that when the width of the network is increased to 16 or 20, the unstable behavior emerges even for sufficient number of test functions. For higher frequencies, the network is not able to converge to the solution with any of these widths, and we can see that increasing the width even makes things worse in Figure 8d. These obseervations allows to conclude that increasing the width of the network cannot heal the need for higher number of test functions, and even require more number of test functions in order to converge to the solution.

In another set of experiments, the width of the network and the number of test functions are kept equal and are increased together. The idea was to treat these two hypermarameters as one parameter since they are bounded together in FEM. The results are presented in Figure 9. We can see that the final H1-error is consistently improved with increasing both N and K together for all of the frequencies, but the improvement becomes more evident as the frequency gets



(a) $k = 2$.

(b) $k = 4$.

(c) $k = 6$.

(d) $k = 8$.

Figure 9: Training loss and H1-error with fixed number of test function and increasing width of the network.

higher. These experiments confirm that the best approach is to keep N and K proportional, and increase them together in order to study the convergence rate of this method.

Figure 10 shows how a quite strong network with $N = K = 20$ performs for different frequencies. Although the network is doing a good job for all of the frequencies, we can see that as the frequency increases, the accuracy of the final solution decreases, which again implies the need to enrich the network for higher frequencies.

3.2.2 Deep Networks

From Equation 11, we can see that for deeper neural networks ($D > 1$), the number of parameters has a quadratic dependancy on the width of the network, N. This gave us the notion that just as increasing N would require more test functions, increasing D might also require the training to be done on more test functions. To address this, we compared the training process for a network with two hidden layers and $N = 12$ with different number of test functions. The results are presented in Figure 11. We can see that with the same number of test functions as for shallow networks, the model is overfitted to the test functions. While with 20 test functions we can get a better H1-error, further increasing the number of test functions to 28 is not advantageous.

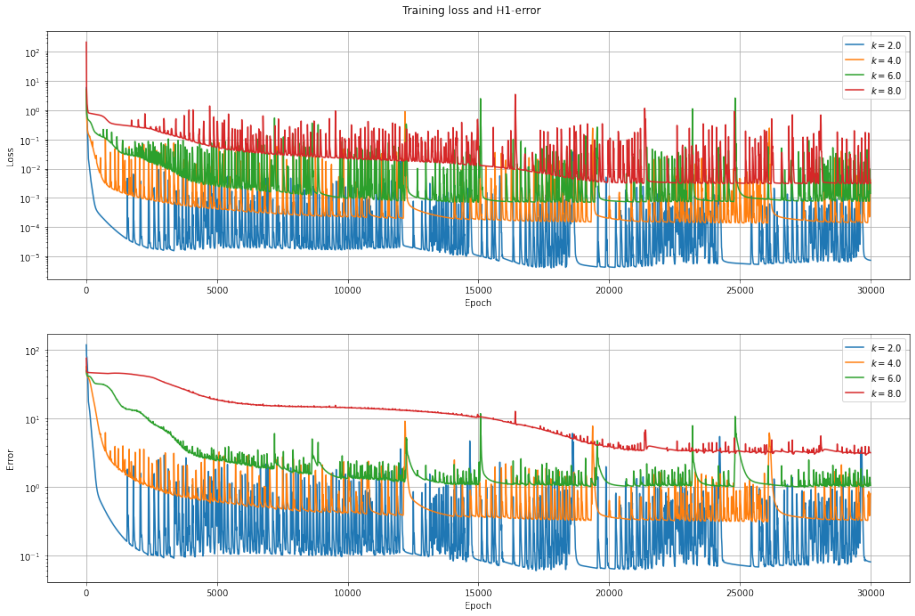


Figure 10: Training loss and H1-error of a shallow network with $N = 20$ and $K = 20$ for different frequencies.

We conclude that 24 test functions as adequate for this network structure with 206 parameters. Comparing this number to the number of parameters (50) of the shallow network with the same width, we can assume that the number of adequate test functions scales with the square root of the number of parameters: $K_{adequate} \sim \sqrt{N_p}$.

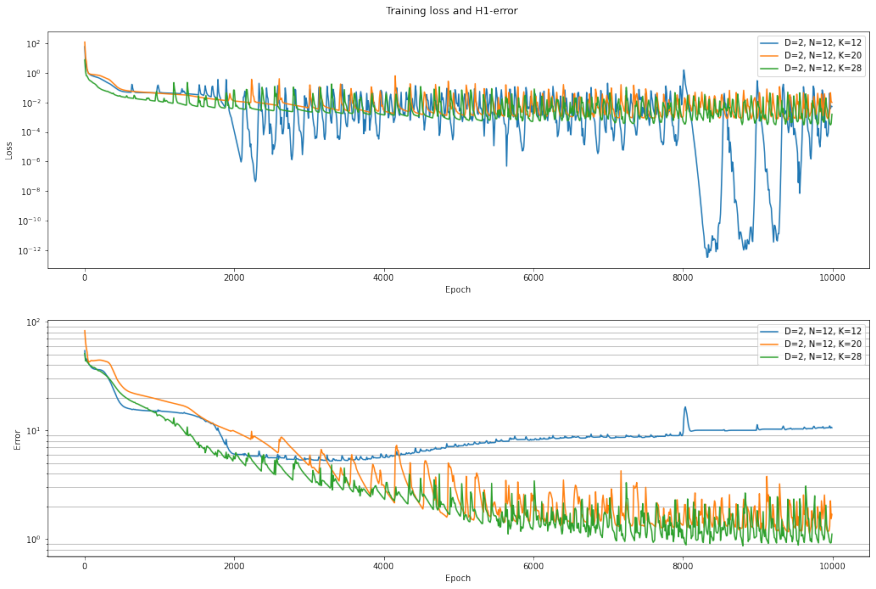


Figure 11: Training loss and H1-error of a network with 2 hidden layers and $N = 12$ for frequency $k = 8.0$, with different number of test functions.

In Figure 12, we keep the width of the network to $N = 12$, increase the depth of the netowrk,

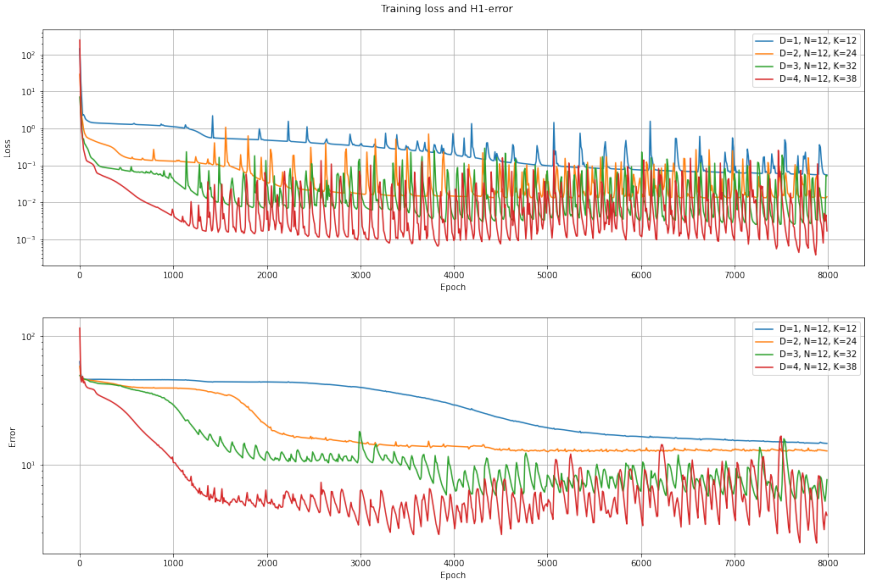


Figure 12: Training loss and H1-error of a network with $N = 20$ for frequency $k = 8.0$, with different depths and number of test functions.

and increase the number of test functions as $K \simeq 12\sqrt{N_p/50}$ as concluded earlier. It shows that with this approach, both the convergence speed and the final accuracy could be improved very well. Comparing this approach with the best approach in shallow neural networks (increasing the width and the number of test functions equally), we saw that on our CPUs, the deep network with $D = 4$, $N = 12$ and 38 test functions reaches a decent H1-error in 15 minutes, while a shallow neural network with 20 nodes and 20 test functions requires 65 minutes for reaching the same accuracy.

3.2.3 V-elliptic Formulation

4 Conclusion

References

[1] M. Raissi, P. Perdikaris, and G.E. 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.

[2] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. Variational physics-informed neural networks for solving partial differential equations, 2019.

[3] Andrea Moiola and Euan A Spence. Is the helmholtz equation really sign-indefinite? *Siam Review*, 56(2):274–312, 2014.

[4] Fernando Henríquez and S Houliston. Neural network galerkin fem for the one-dimensional helmholtz impedance problem in high frequency. 2022.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[7] Min Liu, Zhiqiang Cai, and Jingshuang Chen. Adaptive two-layer relu neural network: I. best least-squares approximation, 2021.

Appendix A