

## REPORT OF PINN PROJECT IN 02456

Group 104: *Ninna Juul Ligaard (S203978), Markus Bjerg Mouritzen (S214518)*  
*Anna Emilie Lunde Borre (S214524), Clara Ruby Wimmelmann (S214471)*

Github link for our jupyter notebook: [https://github.com/StefanKarry/02456\\_PINN\\_Project/blob/main/PINN/Wave\\_equation\\_PINN.ipynb](https://github.com/StefanKarry/02456_PINN_Project/blob/main/PINN/Wave_equation_PINN.ipynb)

The notebook is dependent on all the files in the following Github main repository: [https://github.com/StefanKarry/02456\\_PINN\\_Project/tree/main/PINN](https://github.com/StefanKarry/02456_PINN_Project/tree/main/PINN)

### ABSTRACT

This work investigates two Physics Informed Neural Networks (PINNs) to solve the linear wave equation in the 1D and 3D domain. The PINNs were trained using Partial Differential Equations and relevant boundary and initial conditions as penalty terms in the loss functions. The validation was performed by comparing the predictions and the exact analytical solution. This showed that the error between the predictions using the PINNs and the exact solution was smaller within the training domain. As a PINN was constructed, trained, and outputted interpretable results, it is concluded that the implementation and the project was successful.

### 1. INTRODUCTION

In this report, the wave equation is presented and it is showed how Physics Informed Neural Networks (PINNs) can be used to solve it. An application of the analytical solution for the 1D and 3D domain is used to evaluate the results. PINNs include physical constraints in the loss function to learn the solution to a Partial Differential Equation (PDE), Initial Value Problem (IVP) or Boundary Value Problem (BVP) or combinations thereof, such as the wave equations described later. As such, it effectively learns the solution space from the physical equation itself. The PINN can also be used for prediction on unseen domains, with the analytical solution used for validation, illustrating this networks scalability and flexibility.

### 2. PROBLEM FORMULATION

The PDE under evaluation is the linear wave equation, which is analyzed in both the 1D domain and 3D spherical domain. The analytical solutions presented in the following will be used to validate the trained models.

#### 2.1. 1D Domain

In the 1D domain, the linear wave equation is defined by

$$u_{tt}(x, t) = c^2 u_{xx}(x, t) \quad (1)$$

Here,  $c$  denotes the propagation speed and  $u_{xx}$  the second derivative with respect to  $x$  [1]. The wave equation is considered in the domain  $(x, t) \in [-1, 1] \times [0, T]$  with the initial condition  $\psi(x) = u(x, 0) = \exp[-(\frac{x-x_0}{\sigma})^2]$ .

For the boundaries, the homogeneous Neumann Boundary Conditions (BC) are used, which means that the derivative at the boundaries is zero, as seen in the last line of Equation 2. The 1D wave is then defined by

$$\begin{cases} \frac{\partial^2}{\partial t^2} u(x, t) - c^2 \frac{\partial^2}{\partial x^2} u(x, t) = 0 & (x, t) \in [-1, 1] \times [0, T], \\ u(x, 0) = \psi(x), & -1 \leq x \leq 1, \\ \frac{\partial}{\partial t} u(x, 0) = 0 & -1 \leq x \leq 1, \\ \frac{\partial}{\partial x} u(-1, t) = \frac{\partial}{\partial x} u(1, t) = 0, & t \geq 0 \end{cases} \quad (2)$$

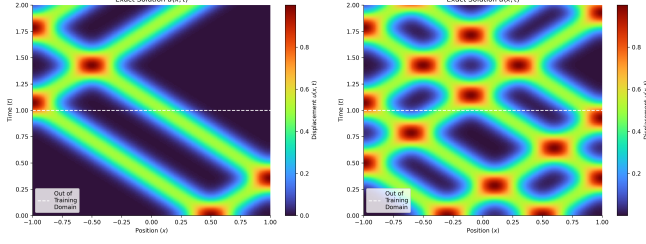
Here,  $t$  denotes the time for which the wave is evaluated and the training output of the PINN is  $\hat{u}(x_i, t_i)$ . The components of the loss functions are derived from the conditions in (2) and given by

$$\begin{cases} \mathcal{M}_{\Omega, PDE}(\Theta) = \frac{1}{N_f} \sum_{i=1}^{N_f} |\hat{u}_{tt}(x_i, t_i) - c^2 \hat{u}_{xx}(x_i, t_i)|^2, \\ \mathcal{M}_{\Omega, IC}(\Theta) = \frac{1}{N_i} \sum_{i=1}^{N_i} (|\hat{u}(x_i, 0) - \psi(x_i)|^2 + |\hat{u}_t(x_i, 0)|^2), \\ \mathcal{M}_{\partial\Omega}(\Theta) = \frac{1}{N_g} \sum_{i=1}^{N_g} |\hat{u}_x(x_i, t_i)|^2. \end{cases} \quad (3)$$

Where,  $\Omega = [-1, 1] \times [0, T]$  denotes the domain upon which the wave is evaluated. The total loss function is then given by a weighted sum of the three in Equation 3 with the weights denoted as  $\beta$ :

$$\begin{aligned} \mathcal{M}_{\text{total}}(\Theta) &= \beta_{\Omega, PDE} \mathcal{M}_{\Omega, PDE}(\Theta) \\ &+ \beta_{\Omega, IC} \mathcal{M}_{\Omega, IC}(\Theta) + \beta_{\partial\Omega} \mathcal{M}_{\partial\Omega}(\Theta), \quad \beta > 0 \end{aligned} \quad (4)$$

In Figure 1 and Figure 2 an example of the exact solution is shown for one and two wave sources, with the initial condition of  $c = 1.4$  and  $t \in [0, 2]$ . Here, displacement  $u$  is defined as a function of time relative to the initial position of the wave.



**Fig. 1.** Exact solution of the linear wave equation on a 1D domain with one source at  $x = 0.5$

**Fig. 2.** Exact solution of the linear wave equation on a 1D domain with two sources at  $x = -0.3$  and  $x = 0.5$

## 2.2. 3D domain

When the linear wave equation is constrained to a spherical domain with radius  $R$ , it becomes the following:

$$u_{tt}(\theta, \phi, t) = c^2 \Delta_{S^2} u(\theta, \phi, t) \quad (5)$$

where  $c$  is the propagation speed,  $(\theta, \phi)$  are the spherical coordinates, and  $\Delta_{S^2}$  is the Laplace-Beltrami operator in the spherical domain given by;  $\Delta_{S^2} = \frac{1}{R^2} [\frac{1}{\sin\theta} \frac{\partial}{\partial\theta} (\sin\theta \frac{\partial}{\partial\theta}) + \frac{1}{\sin^2\theta} \frac{\partial^2}{\partial\phi^2}]$ , in spherical coordinates. The wave equation constrained to a spherical domain is then

$$\begin{cases} \frac{\partial}{\partial t^2} u(\theta, \phi, t) = \frac{v^2}{R^2} [\frac{1}{\sin\theta} \frac{\partial}{\partial\theta} (\sin\theta \frac{\partial u}{\partial\theta}) + \frac{1}{\sin^2\theta} \frac{\partial^2 u}{\partial\phi^2}] \\ u(\theta, \phi, 0) = \psi(\theta, \phi) \\ \frac{\partial}{\partial t} u(\theta, \phi, 0) = g(\theta, \phi) = 0 \end{cases} \quad (6)$$

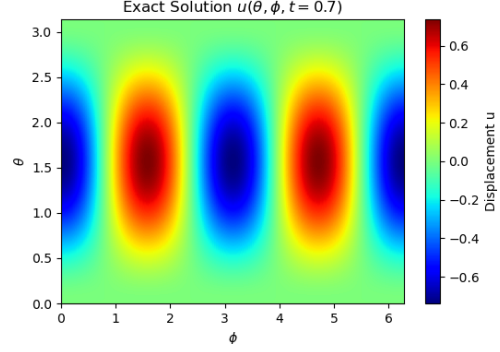
There are no spatial BC's on a sphere, since it does not have a boundary. It is also assumed that the initial state of the wave is Gaussian  $\psi(\theta, \phi) = \exp[-(\frac{d(\theta, \phi)}{\sigma^2})^2]$ , where  $d(\theta, \phi) = (\theta - \theta_0)^2 + (\phi - \phi_0)^2$  is the geodetic distance from the center point  $(\theta_0, \phi_0)$ . For simplicity, the wave is assumed to have a velocity of zero at  $t = 0$ .

The loss function is then given by

$$\begin{cases} \mathcal{M}_{S^2}(\Theta) = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| \hat{u}_{tt}(\theta_i, \phi_i, t_i) - \frac{v^2}{R^2} \left( \frac{1}{\sin\theta_i} \frac{\partial}{\partial\theta} (\sin\theta_i \hat{u}_\theta(\theta_i, \phi_i, t_i)) + \frac{1}{\sin^2\theta_i} \hat{u}_{\phi\phi}(\theta_i, \phi_i, t_i) \right) \right|^2, \\ \mathcal{M}_{IC}(\Theta) = \frac{1}{N_i} \sum_{i=1}^{N_i} \left( |\hat{u}(\theta_i, \phi_i, 0) - \psi(\theta_i, \phi_i)|^2 + |\hat{u}_t(\theta_i, \phi_i, 0)|^2 \right). \end{cases} \quad (7)$$

$\mathcal{N}_f$  is the grid points in the spherical domain  $S^2$  and  $\mathcal{N}_{IC}$  is the grid points at time  $t = 0$ . Here, the total loss function is the weighted sum of the two equations above,

$$\mathcal{M}_{\text{total}}(\Theta) = \beta_{S^2} \mathcal{M}_{S^2}(\Theta) + \beta_{IC} \mathcal{M}_{IC}(\Theta), \quad \beta > 0 \quad (8)$$



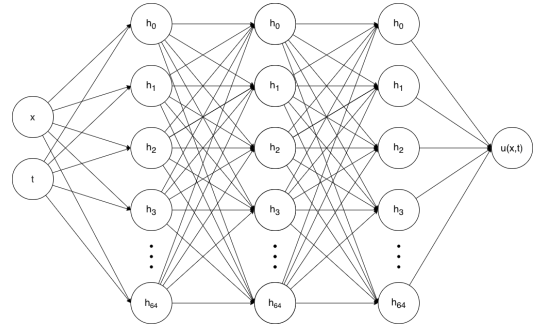
**Fig. 3.** Exact solution of the linear wave on a 3D domain with one wave source.

In Figure 3 a 2D plot of the exact solution is seen for the sphere with radius  $R = 1$ ,  $c = 1.4$  to time  $t = 0.7$ .

## 3. THE PINN ARCHITECTURE

The architecture of a PINN matches that of a simple multi layer perceptron, but often with fewer hidden layers. Both 1D and 3D models are simple models that vary minimally in the number of inputs, layers, and neurons, as seen in Table 1. The activation function used in each layer is tangent hyperbolic (tanh) and sinus (sine), as we must abide by the physics, and hence the popular ReLU activation will not suffice. Sigmoid activation could have been another possibility since it is smooth as well, but because tanh is zero-centered it serves a better purpose for this task.

Since the architectures of the networks used for 1D and 3D are almost alike, only the one for 1D is shown below in Figure 4.



**Fig. 4.** The network takes two inputs, position ( $x$ ) and time ( $t$ ). It consists of 3 hidden layers each with  $64 + 1$  hidden units including bias. The final output is  $u(x, t)$ . Each neuron is activated using tanh in the 1D model's network.

The main component of the PINN is the physics described by the loss function of the network, which makes the need of an overly complex network architecture redundant to obtain good results, as will be evident later in this report.

The loss functions, Equation 4 and Equation 8, require calculation of both first and second order derivatives, which is handled by a function utilizing torch.autograd.

### 3.1. Training

The models for 1D and 3D were trained using two different optimizers, AdamW and L-BFGS[2]. AdamW serves the purpose of roughly estimating the parameters, and L-BFGS was called once at the end to finetune the parameters. Furthermore, before the final optimization using L-BFGS, new data was created and evaluated using the semi-trained model, from which 10.000 collocation points with the highest loss were used for better transparency. Table 1 lists the parameters used throughout the training.

**Table 1.** Model parameters for the 2 models.

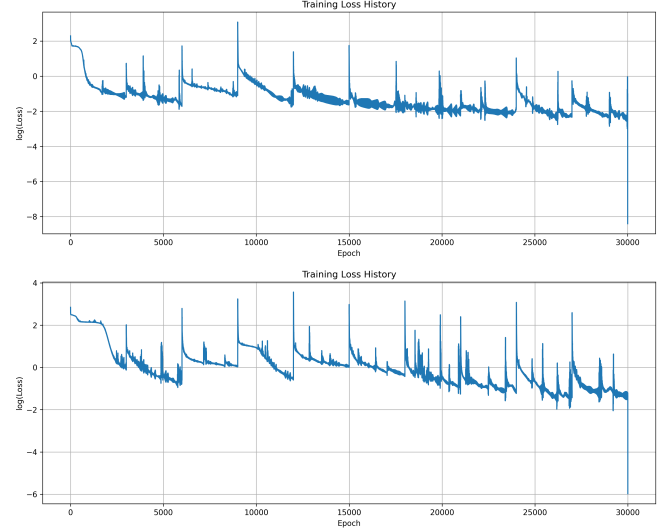
Parameter	1D	3D
Learning rate, AdamW	1e-3	
Learning rate, L-BFGS	1.0	
Hidden layers	3	4
Neurons per layer	64	[64,128,50,25]
Trainable parameter	8577	16,327
<b>Loss function</b>		
Wave speed, c	1.4	
Time domain, t	[0:1]	[0:1]
Radius, R	-	1
Weights	$\beta_{\Omega, PDE} = 10$ $\beta_{\Omega, IC} = 100$ $\beta_{\partial\Omega} = 100$	$\beta_{S^2} = 10$ $\beta_{IC} = 100$

The loss history during training for the 1D domain for both one and two sources is seen in Figure 5, where the effect of the added L-BFGS optimizer is evident at the end of the graph, as it made the loss drop to  $\mathcal{M}_{total} \approx 10^{-8}$  and  $\mathcal{M}_{total} \approx 10^{-6}$ , respectively.

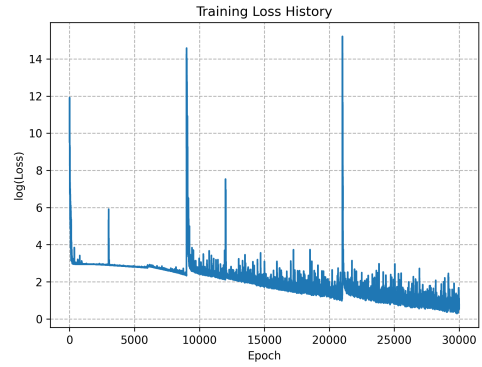
The spikes in the loss history correspond well with the introduction of more data points being introduced in the training loop every 3000 epochs. The loss history for training in the 3D domain is seen in Figure 6, where the effect of the L-BFGS and the spikes are consistent with those in the 1D domain. Both models were trained on GPUs on the DTU HPC.

## 4. RESULTS

The results obtained for the 1D domain are seen in Figure 7. The white dashed line indicates when the PINN predicts outside of the training domain. Inside the training domain for both cases, the PINN works well and aligns great with the exact solution, shown by the absolute error being zero. Outside



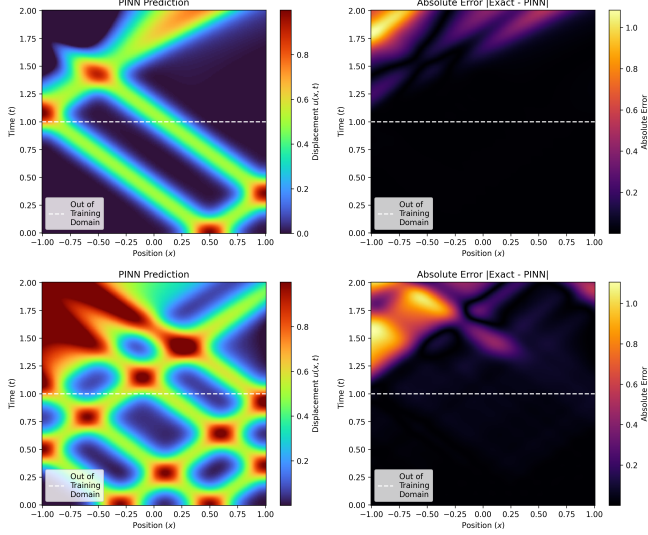
**Fig. 5.** Loss history during training of PINN for one (top) and two (bottom) sources. Trained for 30.000 epochs



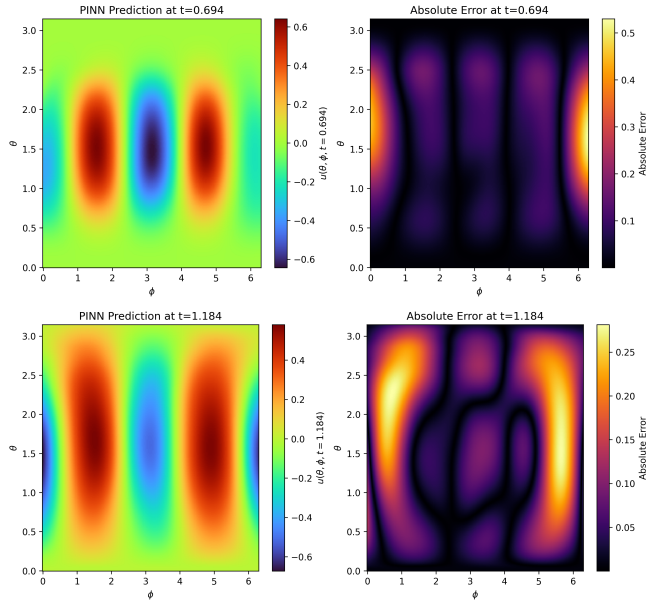
**Fig. 6.** Loss history during training of PINN for one source on the 3D domain. Trained for 30.000 epochs.

of the training domain, the absolute error increases in the top left corner and larger errors are observed when two sources are present, compared to one. This means that PINN works within the training domain, whereas errors occur when predicting outside the domain, the further away from the IC the larger errors.

The result of the linear wave equation on a 3D domain with one source at  $t = 0.694$  is seen in Figure 8. It shows that the prediction follows the expected oscillating structure along  $\phi$ , where a zero displacement of  $u$  is observed near the edges of  $\theta$ . A larger error is observed near the edges of  $\phi$ , whereas smaller errors are seen at the edges of  $\theta$ . Training outside of the time domain at  $t = 1.184$  shows a smaller maximum error between the prediction and the exact solution compared to training inside the domain.



**Fig. 7.** (Left) Predicted wave propagation in 1D domain for one and two sources. (Right) The absolute error between the exact solution and PINN prediction.



**Fig. 8.** (Left) Predicted wave propagation in 3D both inside (top) and outside (bottom) the training domain. (Right) The absolute errors between the predictions and exact solutions.

## 5. DISCUSSION

The error in the 1D results is mostly limited to the predictions outside the training domain, indicating that the PINN has been trained well to the domain, but has not learned the general nature of the 1D linear wave equation, and thus it might overfit. The results of the 3D trained PINN, question

if by enforcing periodic boundary conditions by including an additional term to the 3D PINN's loss function would improve the results. Previously in the report, it was stated that a sphere has no boundary conditions. Geometrically, this is correct. However, the  $\phi$ -domain  $\in [0, 2\pi]$  does not know this. Therefore, it is assessed that this is a mistaken assumption and that it would be interesting to implement in the future, as it could possibly make the error at the  $\theta$  boundaries decrease significantly, i.e. Figure 8.

Both models ended up being more complex than they probably needed to be. Thus, it would be interesting to reduce the complexity of the entire setup drastically, to investigate whether similar results can be obtained, but with fewer resources.

Generally, PINN's, like the ones discussed here, have some major limitations. Most noticeably, the PINN requires re-training every time one might want to investigate new initial conditions, which could be endless. With the (DTU) GPU interactive node, training takes 10-15 minutes, since it is a very inefficient way of solving PDE's, IVP's or BVP's (compared to numerical methods of no-closed-form equations). It could probably speed up the process to run on batch jobs asking for multiple nodes. However, the general picture is that even on GPU it even takes a finite amount of time to train the fully connected forward network depending on the amount of weights chosen, as well as training epoch and backpropagation optimization complexity.

## 6. CONCLUSION

It is concluded that the two PINNs have been constructed, studied, trained and validated for the linear wave equation successfully in the 1D and 3D domains. This was done by designing simple multi-layer perceptrons, and training them on the suitable domains with the derived loss functions constraining the output to the wave equation on the domain as well as initial and boundary conditions. The training of these PINNs was less sensitive to variations in the network architecture, but extremely sensitive to the  $\beta$  weights in the loss functions, as well as the specific training parameters such as the number of epochs (for this problem 30k), the optimization during backpropagation and the initial wave conditions. In this project, no hyperparameter tuning was performed, but simply varied the beta parameters and training iteratively while comparing the model output in the domain with the analytical solution. As seen in Figure 8 and Figure 7, the absolute errors of the PINN generated solution space compared to the analytical solutions are limited well within the training domain, while respectively for the 1D and 3D cases, the absolute error grows to unfortunately larger values outside and at the boundary of the training domains. It illustrates that the PINN learns from (3D: part of) the domain, and as the PINN results are interpretable, the project is successful.

## 7. REFERENCES

- [1] Hollis Williams, “Neumann and dirichlet boundary conditions for the one-dimensional wave equation,” *Physics Education*, vol. 57, 2022.
- [2] PyTorch, “LFBGS,” <https://docs.pytorch.org/docs/stable/generated/torch.optim.LBFGS>, n.d, Accessed: 01-12-2025.

## DECLARATION OF USE OF GENERATIVE AI

We have used generative AI tools for code debugging and translating a Matlab script to Python, provided by Allan Peter Engsig-Karup. This mentioned Matlab-to-python translated file is called *sphre.py* in our main GitHub repository, which we linked to at the front page of this report.

Furthermore for the Jupyter Notebook to compile the function `parse_arguments` in the script `exact_1D_grid.py` under `lib/dataset` was generated using AI.