Neural Network Simulation of Time-variant Waves on Arbitrary Grids with Applications in Active Sonar

Yash Ranjith
Westmont High School
Campbell, California, USA
`yash.ranjith@gmail.com`

Abstract

Wave behavior underlies useful technologies such as active sonar, passive sonar, and echolocation. For instance, wave behavior is used to determine the sonar signature of underwater vehicles for stealth designs. To obtain a sonar signature, the fastest method is to simulate a sonar testbed using physics software. However, these software rely on the finite element method or finite difference method, which are slow due to their dependence on explicit computations. To overcome this issue, researchers have begun developing physics-informed neural networks (PINNs) to learn wave behavior. As PINNs rely on soft computing and act as operators, they can learn from noisy data and make quick predictions. Our PINN differs from other PINNs that are trained on static simulation grids with open boundaries. The goal of our PINN is to predict the reflection of a pressure wave off of a dynamic obstruction in an arbitrary grid, mimicking in part, the behavior of active sonar. Given a random source location, static obstructions, and a dynamic obstruction, the PINN predicted the wave evolution for 500 timesteps in 1.54 seconds and was over 1200 times faster than the finite difference method. The PINN was trained using supervised and unsupervised learning and reached a mean squared error of 3E-5. Results show that the PINN demonstrates visual and numerical accuracy and avoids learning unwanted artifacts like spurious waves, due to its ability to generalize. For future work, we will add more degrees of freedom to define complex dynamic obstruction shapes.

1 Introduction

Wave behavior underlies many useful technologies, such as active sonar, passive sonar, and echolocation [1, 2, 3, 4]. Engineers rely on their understanding of wave behavior to predict the sonar signatures of submarines and unmanned underwater vehicles. To obtain these sonar signatures engineers use physics simulation software. However, such software typically relies on numerical methods like the finite element method (FEM), finite difference method (FDM), or finite volume method (FVM), all of which require explicit computations. This process can be slow, especially when the step size is small, necessitating the use of many steps. A faster alternative to explicit methods is a physics-informed neural network [5]. Because neural networks act as operators, they can solve time-dependent partial differential equations (PDEs) for any time without having to solve for previous timesteps. Further, neural networks can provide the solution for any sub-domain within a larger space. These qualities allow PINNs to take the fastest route that is mathematically possible.

In recent years, there has been a growing number of publications on the use of neural networks to solve the wave equation under various conditions [6]. In a recent paper, authors trained a neural network to learn the wave equation on a static grid with open boundaries [7]. In [8], authors developed a PINN that can learn the wave equation on a layered grid where each layer uses a different wave speed. This PINN has applications when the simulation medium changes, such as different layers of sediment in the ground. However, the layered grid was still a static grid with open boundaries. In [9], authors developed a PINN that can learn the wave equation when the frequency of the point source is allowed to vary. However, the PINN again learned on a static grid with open boundaries. In [10], authors developed a PINN that could learn the wave equation with soft initial conditions. However, the simulation took place in 1D.

Our background review found that recent PINN studies on wave behavior focus on static grids with open boundaries and hard initial conditions. We extend this research by introducing (a) grids with some arbitrariness, (b) reflecting boundaries, and (c) soft initial conditions. We introduce grid arbitrariness by training the neural network on the wave equation with two static rectangular obstructions and one dynamic square obstruction, using the dynamic obstruction's coordinates as inputs, providing two degrees of freedom. We apply the zero-flux condition to obstruction edges for reflecting boundaries. Finally, we allow the initial source coordinates to be input for soft initial conditions. This paper presents a PINN that predicts wave behavior and reflection on grids with some arbitrariness with multiple applications, including predicting the sonar signature of underwater vehicles.

2 Methods

2.1 The Wave Equation and Finite Difference Method

The propagation of waves through a medium can be modeled by the 2D wave equation as shown in (1), which describes the spread of wave pressure *u* in relation to spatial positions *x* and *y*, and time *t*. The finite difference method was employed to solve the wave equation [11]. The forward difference method was used for partial derivatives with respect to time, and the central difference method was used for partial derivatives in space. Equation (2) depicts the discretized wave equation. We then rearrange (2) to arrive at (3). We denote a timestep as the number of fixed time increments needed to reach a specific point in time. We use a point source with the sinusoidal function shown in (4) to create the initial wave. This point source is turned on for t_1 timesteps and is then turned off. The wave is then allowed to propagate for an additional t_2 timesteps. This process can be visualized in Figs. 1a-1b.

This section specifies the constants and notations used in the simulation. All simulations were run in a 500 x 500 grid with an arbitrary unit distance. The coordinates in the simulation environment are represented in a row x column format, with the origin at the upper left corner of the grid. The distance between two horizontally adjacent lattice points, Δx, was set to 0.50. The distance between two vertically adjacent lattice points, Δy, was set to 0.50. The time difference between two timesteps, Δt, was set to 0.001 seconds. The amplitude *A* was set to 1 and the wave speed *c* was set to 343. Finally, timesteps t_1 and t_2 were set to 50 and 500 respectively.

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right] \tag{1}$$

$$\frac{u_{i,j}^{t} - 2u_{i,j}^{t-1} + u_{i,j}^{t-2}}{(\Delta t)^2} = c^2 \left[\frac{u_{i-1,j}^{t-1} - 2u_{i,j}^{t-1} + u_{i+1,j}^{t-1}}{(\Delta x)^2} + \frac{u_{i,j-1}^{t-1} - 2u_{i,j}^{t-1} + u_{i,j+1}^{t-1}}{(\Delta y)^2}\right] \tag{2}$$

$$u_{i,j}^{t} = 2u_{i,j}^{t-1} - u_{i,j}^{t-2} + (\Delta t)^2 \cdot c^2 \left[\frac{u_{i-1,j}^{t-1} - 2u_{i,j}^{t-1} + u_{i+1,j}^{t-1}}{(\Delta x)^2} + \frac{u_{i,j-1}^{t-1} - 2u_{i,j}^{t-1} + u_{i,j+1}^{t-1}}{(\Delta y)^2}\right] \tag{3}$$

$$s(t) = A \sin(2\pi f t) \tag{4}$$

2.2 Defining Boundary Conditions

Equation (5) represents the generalized boundary condition used to reflect wave pressure off obstructions. This equation relates the rate of change of wave pressure at points normal to an obstruction to the reflective parameter γ, wave speed c, and the rate of change of wave pressure over time. By adjusting γ, we control the absorption of the wave by obstructions. To reflect wave pressure, we set γ to 0, indicating zero absorption. We use the finite difference method and rearrange the equation as shown in (6) and (7). The appropriate adj coordinate is used based on the direction relative to the obstruction (left, right, upward, downward). To create a more realistic simulation, we apply an open boundary condition, allowing the wave to propagate outside the environment, similar to sound exiting a window. Here, γ is set to 1, indicating complete absorption. By using the finite difference method on (5) and rearranging the terms in (8), we arrive at the generalized open boundary condition equation in (9), with the corresponding adj coordinate used based on the obstruction location. We incorporate both full reflection and zero reflection into the numerical model, as shown in Fig. 2a, near the obstruction and the environment's edges. Although the wave is expected to exit at the edges, Fig. 2a shows traces of inward reflection due to spurious waves. However, as later results show, the PINN successfully removes this noise.

$$\frac{\partial u}{\partial n} = \frac{\gamma}{c}\frac{\partial u}{\partial t} \tag{5}$$

$$\frac{u^t_{i,j} - u^t_{adj}}{\Delta x^2} = 0 \tag{6}$$

$$u^t_{i,j} = u^t_{adj} \tag{7}$$

$$\frac{u^t_{i,j} - u^t_{adj}}{\Delta x^2} = \frac{\gamma}{c}\frac{u^t_{adj} - u^{t-1}_{adj}}{\Delta t^2} \tag{8}$$

$$u^t_{i,j} = \frac{\Delta t \cdot u^t_{adj} + \frac{\gamma}{c} \cdot \Delta x \cdot u^{t-1}_{i,j}}{\Delta t + \frac{\gamma}{c}\Delta x} \tag{9}$$

2.3 Data Collection for Supervised Learning

The initial step in data collection involves setting up a 500 x 500 simulation grid. We placed two static rectangular obstructions (25 x 100) at (25, 50) and (25, 350). To train the PINN with dynamic obstructions, we randomly selected coordinates (obs_x, obs_y) and placed an 80 x 80 square obstruction at these points, with obs_x ranging from 350 to 400 and obs_y from 225 to 275. Fig. 2b shows the grid with both static and dynamic obstructions. Next, we generated the wave source. The PINN also learns wave behavior for varying source locations by randomly selecting a coordinate ($source_x$, $source_y$) with $source_x$ ranging from 150 to 200 and $source_y$ from 225 to 275, propagating the wave from that point using FDM. Fig. 2c visualizes this setup. Finally, we sampled random points during the simulation. The numerical model was run 200 times with random obs_x, obs_y, $source_x$, and $source_y$ for each iteration. Every 10 timesteps, 5000 random coordinates not overlapping with an obstruction were sampled and included in the supervised dataset. Each collocation point in this dataset is represented as $[t\,,\, x,\, y,\, source_x,\, source_y,\, obs_x,\, obs_y]$, with seven inputs and one output (wave pressure). Data was scaled between [-1, 1] for training, using the formulas in (10) and (11).

$$x_{mid} = \frac{x_{min} + x_{max}}{2} \tag{10}$$

$$x_{scaled} = \frac{x - x_{mid}}{x_{max} - x_{min}} \tag{11}$$

2.4 Supervised learning, Unsupervised learning, and Hybrid Learning

The goal of the supervised learning portion of our neural network training is to minimize the mean squared error (MSE). At a collocation point, if we assume the PINN's prediction to be y_{pred} and the actual wave pressure to be y_{act}, then the supervised loss function can be described as shown in (12).

The wave PDE was coded into the neural network for unsupervised learning. Specifically, the equation discretized using FDM, as shown in (3) was used in the loss function to train the PINN on wave propagation at a general point. We executed this method for 128 randomly chosen points that did not fall on an obstruction. Assuming a batch size of 128, (13) represents the MSE of the PINN's prediction at the next timestep against its prediction at the previous timesteps. We represented the PINN's prediction at a given time, location, source location, and obstruction location to be Q, any vertically or horizontally adjacent point at the previous timestep to be Q^{t-1}_{adj}, and the model's prediction two timesteps ago at the same point, source, and obstruction location to be Q^{t-2}. The loss function was constructed under the assumption that Δx is equal to Δy.

We also trained the PINN on the zero-flux boundary condition. To do this, we passed the model's prediction to (7). Then, we used the MSE as the loss function for obstruction boundaries. This method was executed for 128 randomly chosen points that fall on the edges of the obstruction. This new loss function is shown in (14).

It is important that the PINN receives specialized training to accurately learn the initial state. Since wave propagation can commence at any point in time, the neural network must be trained to recognize the initial state at timesteps 0 and 1. The corresponding loss is calculated using the MSE between the actual values sampled at the beginning of the simulation and the network's predictions at that time. This method was executed for 128 randomly selected points from the initial two timesteps. The resulting loss function is outlined in (15), where t is defined as either 0 or 1 to represent one of the first two timesteps. Finally, we define the total unsupervised loss, $\mathcal{L}_U$, as the aggregate of all three unsupervised losses, as detailed in (16).

A hybrid learning approach enables the PINN to learn various scenarios through both supervised and unsupervised learning [12], combining losses from both. Solely using supervised training risks overfitting, while unsupervised training alone may struggle with convergence but tends to generalize better. The hybrid method balances quick convergence from supervised learning with the generalization ability of unsupervised learning. Equation (17) calculates the hybrid learning loss. Initially, the weight *w* in (17) was set to a small value and increased periodically, allowing the network to first learn the wave equation from supervised data, and then generalize using unsupervised learning for un-sampled scenarios.

$$\mathcal{L}_S = \frac{1}{n}\sum_{i=1}^{n}(y_{pred} - y_{act})^2 \tag{12}$$

$$\mathcal{L}_G = \frac{1}{128}\sum_{i=1}^{128}\left(Q^t - \left(\frac{\Delta t \cdot c}{\Delta x}\right)^2 \cdot \sum Q^{t-1}_{adj} + 2Q^{t-1} - Q^{t-2}\right)^2 \tag{13}$$

$$\mathcal{L}_O = \frac{1}{128}\sum_{i=1}^{128}\left(Q^t - Q^t_{adj}\right)^2 \tag{14}$$

$$\mathcal{L}_I = \frac{1}{128}\sum_{i=1}^{128}\left(Q^t - u^t_{i,j}\right) \tag{15}$$

$$\mathcal{L}_U = \mathcal{L}_G + \mathcal{L}_O + \mathcal{L}_I \tag{16}$$

$$\mathcal{L}_H = \mathcal{L}_S + w \cdot \mathcal{L}_U \tag{17}$$

2.5 PINN Architecture

The PINN's input layer has 7 neurons, corresponding to the elements in a collocation point. Each of the four hidden layers has 200 densely connected neurons for sufficient complexity. The output layer has 1 neuron, representing the predicted wave pressure. We used the hyperbolic tangent as the activation function, scaling input data to [-1, 1]. Adam was selected as the optimizer due to its momentum, which helps the PINN avoid local minima and maxima. The batch size was set to 128 across the supervised, unsupervised, and hybrid learning sections. Training began with a learning rate of 0.001 and was later decayed to 0.0005.

3 Results

Cumulatively, the PINN was trained for 180 epochs. Initially, the learning rate of the PINN was set to 0.001 and trained for 80 epochs to achieve an MSE of 9E-5, after which the hybrid loss remained stagnant. So, we reduced the learning rate to 0.0005 and trained the neural network for another 100 epochs. The weight of the unsupervised section was also increased to help the PINN train better. After these changes, the PINN proceeded to smoothly reduce the MSE down to 3E-5. On average, each epoch took 3641 seconds to complete.

To verify the accuracy of the PINN, we compared the numerical model's output to the PINN's output. Fig. 3a shows the results of the numerical model versus the PINN after 500 timesteps for the noise source located at (175, 225) and the dynamic obstruction located at (400, 275). Fig. 3b shows the results of the numerical model versus the PINN after 500 timesteps for the noise source located at (200, 275) and the dynamic obstruction located at (350, 225). The images show that the PINN learned the general structure of the wave around static and dynamic obstructions. We can expect the error rate to continue to drop with further training.

To provide another statistic to benchmark the PINN against the numerical model, we compared the time taken by the PINN with the baseline from the numerical model. To predict the wave pressure at one specific point on the grid after 500 timesteps, the PINN, on average, took 0.0042 seconds. To predict the spread of wave pressure for the entire 500 x 500 grid after 500 timesteps, the PINN, on average, took 1.54 seconds. In contrast, the numerical simulation took 31.49 minutes on average for both a single point and the entire grid.

A key feature of the PINN became apparent after testing its predictive capabilities. In Fig. 3a, the numerical model displays spurious waves in the form of multiple reflections off the top of the environment and waves off obstructions, which should not occur. We can also see this effect at the top of Fig. 3b. By using unsupervised learning, the PINN was able to ignore these noisy components. We can see the PINN's ability to do this in the lower half of Figs. 3a-3b, where there are no spurious waves in the simulation.

4 Conclusion

The PINN was trained on the wave equation using custom features, including an arbitrary grid, random source location, and dynamic obstruction. Numerical simulations were generated via the finite difference method, applying full reflection boundary conditions on obstructions and full absorption on the simulation edges. These simulations provided around 50 million data points for supervised learning. Additionally, finite difference versions of the wave equation and boundary conditions were integrated into the PINN's loss function, enabling unsupervised learning. Together, these methods allowed the PINN to accurately model wave pressure.

The PINN achieved an MSE under 3E-5 and visually matched the baseline. Training spanned 180 epochs, each lasting approximately 3641 seconds. Further training could likely reduce the MSE even more. The unsupervised learning aspect likely contributed to the PINN's ability to remove boundary-induced noise. The PINN was, on average, 1200 times faster than the numerical method in predicting wave pressure at timestep 500. Moreover, the PINN can predict wave pressure at any specific grid point or timestep, whereas the numerical method must solve the entire grid.

Our PINN, and others of similar design, can only have practical application if they are able to solve for arbitrary as opposed to static grids. This work introduced arbitrariness via dynamic obstructions. Future work could increase arbitrariness by defining obstructions with more coordinates, allowing for more degrees of freedom to support complex shapes.

References

[1] Hubert Chanson. Tsunami surges on dry coastal plains: Application of dam break wave equations. *Coastal Engineering Journal*, 48(4):355–370, 2006. `https://doi.org/10.1142/S0578563406001477`.

[2] Jeno Gazdag. Modeling of the acoustic wave equation with transform methods. *Geophysics*, 46(6):854–859, June 1981. `https://doi.org/10.1190/1.1441223`.

[3] O. I. Berngardt and A. P. Potekhin. Radar equations in the radio wave backscattering problem. *Radiophysics and Quantum Electronics*, 43(6):484–492, June 2000. `https://doi.org/10.1007/BF02677176`.

[4] Ariana Mendible, James Koch, Henning Lange, Steven L. Brunton, and J. Nathan Kutz. Data-driven modeling of rotating detonation waves. *Phys. Rev. Fluids*, 6:050507, May 2021. `https://link.aps.org/doi/10.1103/PhysRevFluids.6.050507`.

[5] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, mar 2021. `http://dx.doi.org/10.1038/s42256-021-00302-5`.

[6] Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Solving the wave equation with physics-informed deep learning, 2020. `https://doi.org/10.48550/arXiv.2006.11894`.

[7] Chao Song, Tariq Alkhalifah, and Umair Bin Waheed. Solving the frequency-domain acoustic VTI wave equation using physics-informed neural networks. *Geophysical Journal International*, 225(2):846–859, 01 2021. `https://doi.org/10.1093/gji/ggab010`.

[8] Ali Imran Sandhu, Umair bin Waheed, Chao Song, Oliver Dorn, and Pantelis Soupios. Multi-frequency wavefield modeling of acoustic vti wave equation using physics informed neural networks. *Frontiers in Earth Science*, 11, 2023. `https://www.frontiersin.org/articles/10.3389/feart.2023.1227828`.

[9] Harpreet Sethi, Doris Pan, Pavel Dimitrov, Jeffrey Shragge, Gunter Roth, and Ken Hester. Hard enforcement of physics-informed neural network solutions of acoustic wave propagation. *Computational Geosciences*, 27(5):737–751, October 2023. `https://doi.org/10.1007/s10596-023-10232-3`.

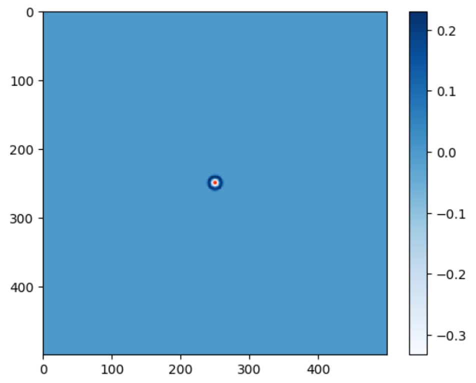
[10] Shaikhah Alkhadhr and Mohamed Almekkawy. Wave equation modeling via physics-informed neural networks: Models of soft and hard constraints for initial and boundary conditions. *Sensors*, 23(5), 2023. `https://www.mdpi.com/1424-8220/23/5/2792`.

[11] M. C. Bhattacharya. Finite-difference solutions of partial differential equations. *Communications in Applied Numerical Methods*, 6(3):173–184, 1990. `https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.1630060303`.

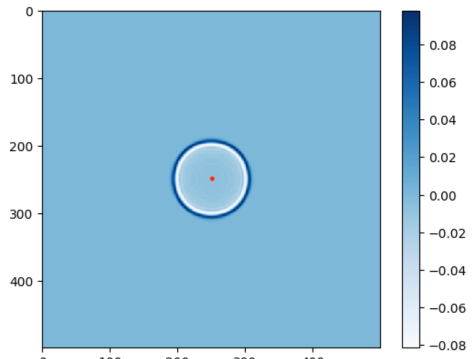
[12] Chunyue Lv, Lei Wang, and Chenming Xie. A hybrid physics-informed neural network for nonlinear partial differential equation. *International Journal of Modern Physics C*, 34(06):2350082, 2023. `https://doi.org/10.1142/S0129183123500821`.

5 Supplemental Material

5.1 Figures

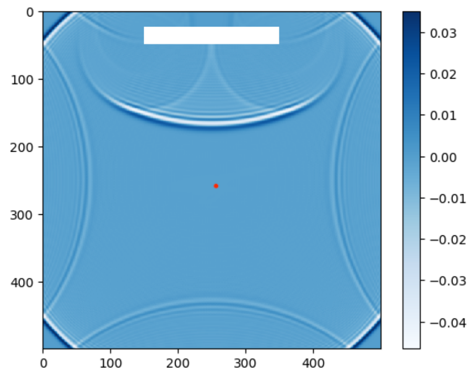


(a)

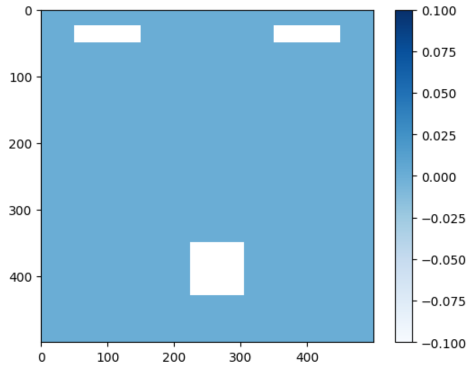


(b)

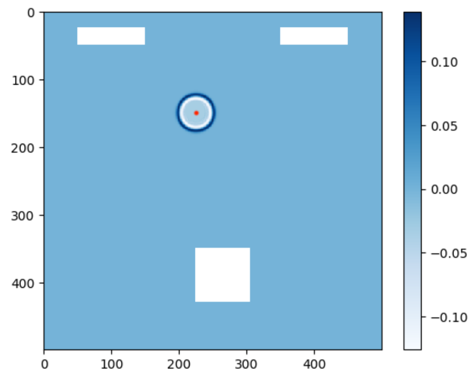
Figure 1: (a) One period of propagation by the numerical model, (b) Numerical simulation 100 timesteps after Fig. 1a



(a)

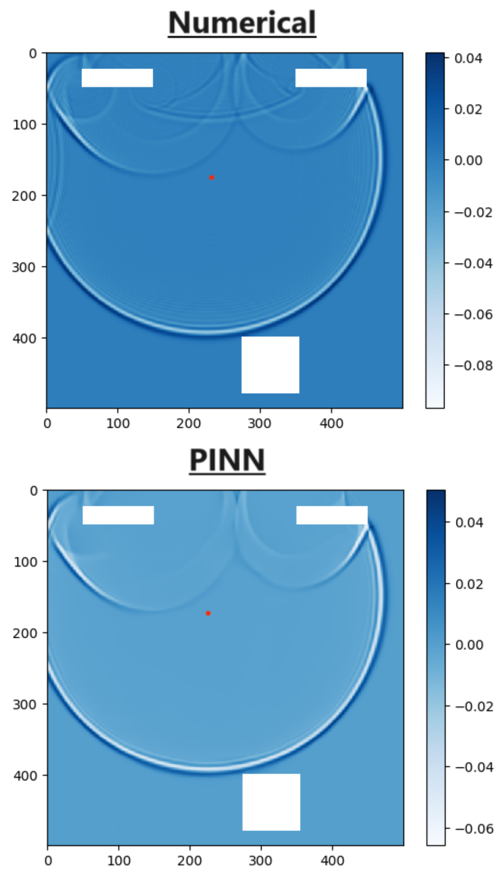


(b)

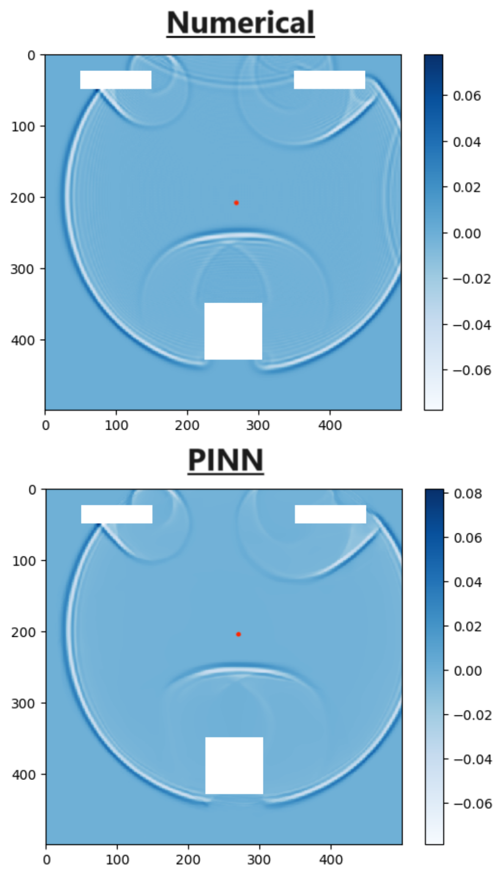


(c)

Figure 2: (a) Numerical simulation with zero and full absorption boundaries, (b) Graphical representation of generated obstructions with obs_x = 350 and obs_y = 225, (c) Graphical representation of generated obstructions with obs_x = 350, obs_y = 225, $source_x$ = 150, and $source_y$ = 225.



(a)



(b)

Figure 3: (a) Test 1: Numerical model versus PINN for 500 timesteps with obs_x = 400, obs_y = 275, $source_x$ = 175, and $source_y$ = 225, (b) Test 2: Numerical model versus PINN for 500 timesteps with obs_x = 350, obs_y = 225, $source_x$ = 200, and $source_y$ = 275.