

# Finding the Shape of Lacunae of the Wave Equation Using Artificial Neural Networks

Alina Chertock, Christopher Leonard, and Semyon Tsynkov

**Abstract** We apply a fully connected neural network to determine the shape of the lacunae in the solutions of the wave equation. Lacunae are the regions of quietness behind the trailing fronts of the propagating waves. The network is trained using a computer simulated data set containing a sufficiently large number of samples. The network is then shown to correctly reconstruct the shape of lacunae including the configurations when it is fully enclosed.

## 1 Introduction

Consider the inhomogeneous scalar wave (d'Alembert) equation in the three-dimensional space (3D):

$$\frac{1}{c^2} \frac{\partial^2 u}{\partial t^2} - \Delta u = f(\mathbf{x}, t), \quad \mathbf{x} \in \mathbb{R}^3, \quad t \geq 0, \quad (1)$$

subject to zero initial conditions, and with the source term  $f$  that is compactly supported on a bounded (3+1)D domain  $Q_f \subset \mathbb{R}^3 \times [0, +\infty)$ . The solution  $u$  to equation (1) is given by the Kirchhoff integral:

$$u(\mathbf{x}, t) = \frac{1}{4\pi} \iiint_{|\mathbf{x}-\boldsymbol{\xi}| \leq ct} \frac{f(\boldsymbol{\xi}, t - |\mathbf{x} - \boldsymbol{\xi}|/c)}{|\mathbf{x} - \boldsymbol{\xi}|} d\boldsymbol{\xi}. \quad (2)$$

The integration in (2) is performed in space over the ball of radius  $ct$  centered at  $\mathbf{x}$ , but as  $f$  is taken at retarded moments of time, this can be interpreted as integration in the (3+1)D space-time over the surface of a backward characteristic cone of equation (1) (light cone of the past) with the vertex  $(\mathbf{x}, t)$ . This surface may or may

---

Department of Mathematics, North Carolina State University, Raleigh, NC 27695, USA. e-mail: [chertock@math.ncsu.edu](mailto:chertock@math.ncsu.edu), [cleonar@ncsu.edu](mailto:cleonar@ncsu.edu), [tsynkov@math.ncsu.edu](mailto:tsynkov@math.ncsu.edu)

not intersect with the support  $Q_f$  of the right-hand side  $f$ . If there is no intersection, then  $u(\mathbf{x}, t) = 0$ , which implies, in particular, that the solution  $u = u(\mathbf{x}, t)$  of equation (1) will have a lacuna (secondary lacuna in the sense of Petrowsky [1]):

$$u(\mathbf{x}, t) \equiv 0 \quad \forall (\mathbf{x}, t) \in \bigcap_{(\xi, \tau) \in Q_f} \{(\tilde{\mathbf{x}}, \tilde{t}) \mid |\tilde{\mathbf{x}} - \xi| < c(\tilde{t} - \tau), \tilde{t} > \tau\} \stackrel{\text{def}}{=} \Lambda. \quad (3)$$

Mathematically, the lacuna  $\Lambda$  is the intersection of all forward characteristic cones (i.e., light cones of the future) of the wave equation (1) once the vertex of the cone sweeps the support  $Q_f$  of the right-hand side  $f(\mathbf{x}, t)$ . From the standpoint of physics,  $\Lambda$  is the part of space-time where the waves generated by a compactly supported source have already passed and the solution has become zero again. The primary lacuna (as opposed to secondary lacuna (3)) is the part of space-time ahead of the propagating fronts where the waves have not reached yet.

The phenomenon of lacunae is inherently three-dimensional (more precisely, it pertains to spaces of odd dimension). The surface of the lacuna includes the trajectory of aft (trailing) fronts of the propagating waves. The existence of sharp aft fronts in odd-dimension spaces is known as the (strong) Huygens' principle, as opposed to the so-called wave diffusion, which takes place in spaces of even dimension [2, 3].

The question of identifying the hyperbolic equations and systems that admit the diffusionless propagation of waves has been first formulated by Hadamard [4–6]. He, however, did not know any other examples besides the d'Alembert equation (1). The notion of lacunae was introduced and studied by Petrowsky in [1], where conditions for the coefficients of hyperbolic equations that guaranteed the existence of lacunae have been obtained (see also [3, Chapter VI]). Subsequent developments can be found in [7, 8]. However, since work [1] no other constructive examples of either scalar equations or systems that satisfy the Huygens' principle have been found except for the wave equation (1) and its equivalents. Specifically, it was shown in [9] that in the standard (3 + 1)D space-time with Minkowski metric, the only scalar hyperbolic equation that has lacunae is the wave equation (1). The first examples of nontrivial diffusionless equations (i.e., irreducible to the wave equation) were constructed in [10–12], but the space must be  $\mathbb{R}^d$  for odd  $d \geq 5$ . Examples of nontrivial diffusionless systems (as opposed to scalar equations) in the standard Minkowski (3 + 1)D space-time were presented in [13–15], as well as examples of nontrivial scalar Huygens' equations in a (3 + 1)D space-time equipped with a different metric (the plane wave metric that contains off-diagonal terms), see [14–16]. It was shown in [17] that the wave equation on the  $d$ -dimensional sphere, where  $d \geq 3$  is odd, satisfies the Huygens' principle; this spherical wave equation can be transformed to the Euclidean wave equation locally, but not globally.

While the examples of nontrivial diffusionless equations/systems built in [10–16] are primarily of a theoretical interest, the original wave equation (1) accounts for a variety of physically relevant (albeit sometimes simplified) models in acoustics, electromagnetism, elastodynamics, etc. Accordingly, understanding the shape of the lacunae (3) is of interest for the aforementioned application areas as lacunae represent the regions of “quietness” where the corresponding wave field is zero.

The objective of our work is to determine the shape of the lacunae in the solutions of the wave equation using the fully connected artificial neural networks. In the current paper, we adopt a simplified scenario to construct, test, and verify the proposed machine learning approach. Specifically, while the true phenomenon of lacunae is 3D and applies to solutions given by the Kirchhoff integral (2), hereafter we conduct the analysis and simulations in a one-dimensional (1D) setting. The domain of dependence for  $u(x, t)$  determined by the Kirchhoff integral is the surface of the backward light cone. To mimic that in 1D, we consider the function  $u = u(x, t)$  and define its domain of dependence as the sum of two backward propagating rays:

$$\{(\xi, \tau) : \xi - x = \pm c(\tau - t), \quad \xi \in \mathbb{R}, \quad \tau \leq t\} \stackrel{\text{def}}{=} \mathcal{L}(x, t). \quad (4)$$

We do not need a full specification of  $u$  for our subsequent considerations. We only need a sufficient condition for  $u(x, t)$  to be equal to zero, which we take as

$$u(x, t) = 0 \quad \text{if} \quad \mathcal{L}(x, t) \cap Q_f = \emptyset,$$

where  $Q_f$  is a given bounded domain in (1+1)D space-time:  $Q_f \subset \mathbb{R} \times [0, +\infty)$ . Accordingly, the function  $u$  is going to have a lacuna:

$$u(x, t) \equiv 0 \quad \forall (x, t) \in \{(\tilde{x}, \tilde{t}) : \mathcal{L}(\tilde{x}, \tilde{t}) \cap Q_f = \emptyset\} \stackrel{\text{def}}{=} \Lambda_1(Q_f). \quad (5)$$

The lacuna  $\Lambda_1(Q_f)$  combines both the secondary and primary lacuna as per the discussion in Section 1. A purely secondary lacuna would be given by [cf. (3)]

$$u(x, t) \equiv 0 \quad \forall (x, t) \in \bigcap_{(\xi, \tau) \in Q_f} \{(\tilde{x}, \tilde{t}) : |\tilde{x} - \xi| < c(\tilde{t} - \tau), \quad \tilde{t} > \tau\} \subset \Lambda_1(Q_f). \quad (6)$$

We emphasize that the proposed 1D construct is not accurate on the substance. Its only purpose is to provide an inexpensive testing framework for the neural networks described in the paper. This construct is designed as a direct counterpart of the physical 3D setting and does not represent a true solution of the 1D wave equation:

$$\frac{1}{c^2} u_{tt} - u_{xx} = f(x, t), \quad x \in \mathbb{R}, \quad t > 0. \quad (7)$$

The solution of (7) subject to zero initial conditions is given by the d'Alembert integral [cf. the Kirchhoff integral (2)]:

$$u(x, t) = \frac{c}{2} \int_0^t d\tau \int_{x+c(\tau-t)}^{x-c(\tau-t)} f(\xi, \tau) d\xi. \quad (8)$$

Unlike (4), the domain of dependence for the 1D solution (8) contains not only the two rays, but the entire in-between region as well:

$$\{(\xi, \tau) : x + c(\tau - t) \leq \xi \leq x - c(\tau - t), \quad \tau \leq t\} \stackrel{\text{def}}{=} \mathcal{D}(x, y).$$

Therefore, the solution  $u = u(x, t)$  defined by (8) will not, generally speaking, have secondary lacunae as presented by (6).<sup>1</sup>

The paper is organized as follows. In Section 2, we provide a description of the numerical algorithm to build the data set. In Section 3, we introduce the neural networks (NN) and discuss their training. In Section 4, we present the numerical results. Section 5 contains our conclusions and identifies directions for future work.

## 2 Construction of the Data Set

In this section, we describe the construction of the data set needed for the network training. To this end, we introduce a computational domain  $\Omega := [a, b] \times [0, T]$  and discretize it in space and time using a uniform spatial  $x_1, \dots, x_{N_x}$  and temporal  $t_1, \dots, t_{N_t}$  mesh so that

$$\begin{aligned} x_j &= a + (j - 1)\Delta x, \quad j = 1, \dots, N_x \\ t_n &= (n - 1)\Delta t, \quad n = 1, \dots, N_t, \end{aligned}$$

where  $\Delta x = \frac{b-a}{N_x-1}$  and  $\Delta t = \frac{T}{N_t-1}$ .

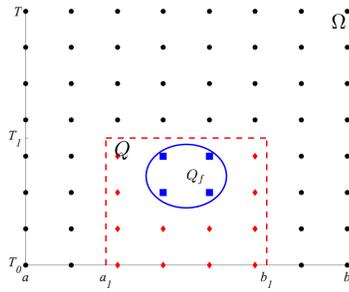


Fig. 1: The computational domain  $\Omega$  with subdomain  $Q$  inside the dotted red line, and  $Q_f$  inside the blue circle.

determined by identifying the set of nodes, for which the characteristic lines  $\mathcal{L}(x_j, t_n)$  emerging from the node  $(x_j, t_n) \in \Omega$ ,  $j = 1, \dots, N_x$ ,  $n = 1, \dots, N_t$  (see formula (4)), pass through the domain  $Q_f$ . We therefore construct  $M$  training data sets by implementing the following algorithm:

We assume that the domain  $Q_f$  that defines the lacuna  $\Lambda_1$  in (5) lies inside a subdomain  $Q \subset \Omega$ , which, for the simplicity of implementation, is taken as a box  $Q = [a_1, b_1] \times [T_0, T_1] \subseteq \Omega$ , where  $a \leq a_1 < b_1 \leq b$  and  $0 \leq T_0 < T_1 \leq T$ . We can then denote the nodes inside  $Q$  by

$$\begin{aligned} x_{j_\ell} &= x_{j_1+(\ell-1)}, \quad \ell = 1, \dots, N'_x \leq N_x, \\ t_{n_p} &= t_{n_1+(p-1)}, \quad p = 1, \dots, N'_t \leq N_t, \end{aligned}$$

where  $x_{j_1}$  and  $t_{n_1}$  are the smallest  $x_j$  and  $t_n$  values that are inside the the set  $Q$  and  $x_{j_1+N'_x-1}$  and  $t_{n_1+N'_t-1}$  are the largest  $x_j$  and  $t_n$  values that are inside the the set  $Q$ , see Figure 1.

<sup>1</sup> The 1D case is special in the sense that while the dimension of the space is odd, the wave equation (7) is not Huygens' when driven by a source term. It may, however, demonstrate a Huygens' behavior with respect to the initial data [2].

**Algorithm 1**

1. *Start:* Introduce the computational domain  $\Omega$  and its discretization by a uniform mesh  $(x_j, t_n) \in \Omega$ ,  $j = 1, \dots, N_x$ ,  $n = 1, \dots, N_t$ , and identify the set of nodes  $(x_{j_\ell}, t_{n_p}) \in \mathcal{Q}$ ,  $\ell = 1, \dots, N'_x$ ,  $p = 1, \dots, N'_t$ .
2. *Iterate:* For  $m = 1, 2, \dots, M$ 
  - a. Generate a random positive integer  $I^{(m)}$  and a set of domains  $\mathcal{Q}_{f_i^{(m)}} \subset \mathcal{Q}$  for each  $i = 1, \dots, I^{(m)}$ , and define

$$\mathcal{Q}_f^{(m)} = \left( \bigcup_{i=1}^{I^{(m)}} \mathcal{Q}_{f_i^{(m)}} \right). \quad (9)$$

- b. Construct the following matrices  $\Phi^{(m)}$  and  $\Psi^{(m)}$ :
  - Matrix  $\Phi^{(m)}$  with entries  $\phi_{\ell,p}^{(m)}$  that indicate for each node  $(x_{j_\ell}, t_{n_p}) \in \mathcal{Q}$  whether or not it belongs to the domain  $\mathcal{Q}_f^{(m)}$ , namely,

$$\phi_{\ell,p}^{(m)} = \begin{cases} 1, & (x_{j_\ell}, t_{n_p}) \in \mathcal{Q}_f^{(m)}, \\ -1, & \text{otherwise.} \end{cases}$$

- Matrix  $\Psi^{(m)}$  with entries  $\psi_{j,n}^{(m)}$  indicates whether or not the characteristic lines  $\mathcal{L}(x_j, t_n)$  intersect with the domain  $\mathcal{Q}_f^{(m)}$ , namely,

$$\psi_{j,n}^{(m)} = \begin{cases} -1, & (x_j, t_n) \in \Lambda_1(\mathcal{Q}_f^{(m)}), \\ 1, & \text{otherwise.} \end{cases} \quad (10)$$

In practice, we check whether there is a node  $(x_{j_\ell}, t_{n_p}) \in \mathcal{Q}_f^{(m)}$  and point  $(\xi, t_{n_p}) \in \mathcal{L}(x_j, t_n)$ , such that  $|\xi - x_{j_\ell}| < \Delta x$ , in which case  $\psi_{j,n}^{(m)} = 1$ ; otherwise  $\psi_{j,n}^{(m)} = -1$ . Note that, the point  $(\xi, t_{n_p})$  is not, generally speaking, a grid node. The construction of  $\Psi^{(m)}$  is visualized in Figure 2.

3. *Form data set:* Store the set  $\{\Phi^{(m)}, \Psi^{(m)}\}_{m=1}^M$ .

### 3 Construction of the Neural Network and Training

In this paper, we use a fully connected feed forward neural network  $N_\Theta : \mathbb{R}^{N'_x \times N'_t} \rightarrow \mathbb{R}^{N_x \times N_t}$  to approximate the shape of the lacunae. To train  $N_\Theta$ , we search for the parameter set  $\Theta$  that minimizes the loss function:

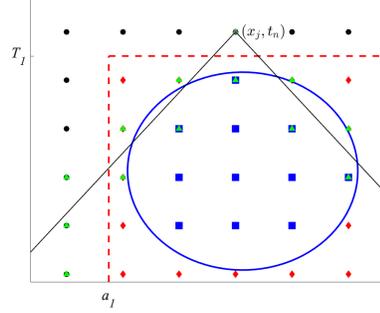


Fig. 2: Characteristic lines from the point  $(x_j, t_n)$ . Green triangles indicate nodes within  $\Delta x$  of the characteristic lines at each time step. If there exist a green triangle and blue square at the same node, then  $\Psi_{j,n} = 1$ , else  $\Psi_{j,n} = -1$ .

$$L(\Theta) = \frac{1}{M} \sum_{m=1}^M \|\Psi^{(m)} - N_{\Theta}(\Phi^{(m)})\|, \quad (11)$$

where  $\|\cdot\|$  is a given norm. The fully connected neural network with  $K$  hidden layers is the composition of functions

$$N_{\Theta} = N_{\Theta_{K+1}} \circ N_{\Theta_K} \circ \dots \circ N_{\Theta_1}, \quad (12)$$

with

$$N_{\Theta_k}(z) = \sigma_k(A_k), \quad A_k = W_k z + b_k, \quad k = 1, 2, \dots, K+1,$$

where  $A_k$  is the connection between layers  $k-1$  and  $k$ , and  $\sigma_k$  is the activation for the layer  $k$ . Here,  $k=0$  corresponds to the input layer and  $k=K+1$  corresponds to the output layer, and we assume that there are  $w_k$  nodes on each layer  $k=0, \dots, K+1$ . For each connection between layers,  $W_k \in \mathbb{R}^{w_k \times w_{k-1}}$  is the weight matrix and  $b_k \in \mathbb{R}^{w_k}$  is the bias vector. The parameters in the sets  $\Theta_k$  are the element values for the weight matrix  $W_k$  and bias vector  $b_k$ , and  $\Theta = \cup_{k=1}^{K+1} \Theta_k$  is the parameter set that we train the neural network to learn by minimizing the loss function (11). We can represent the architecture of a fully connected feed forward neural network as shown in Figure 3.

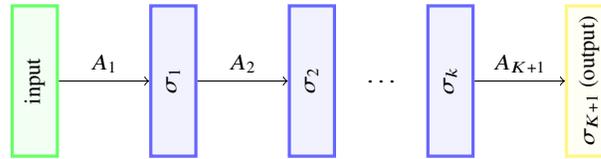


Fig. 3: Neural Network Architecture

For each  $k = 1, 2, \dots, K + 1$ , the activation function  $\sigma_k$  is usually a nonlinear function that is applied to each element of its input vector. There are many activation functions that can be used, some of the most popular activation functions used include the rectified linear unit (ReLU), sigmoid functions, softmax, etc [18].

We train the neural network  $N_{\Theta}$  with the following algorithm:

### Algorithm 2

1. *Start*: Randomly split the data set into two sets, the training set of size  $M_{tr}$ ,  $\{\Phi^{(m_i)}, \Psi^{(m_i)}\}$ ,  $i = 1, 2, \dots, M_{tr}$  and the validation set of size  $M_{val}$ ,  $\{\Phi^{(m_l)}, \Psi^{(m_l)}\}$ ,  $l = 1, 2, \dots, M_{val}$ .
  - a. The training set is used in the optimization algorithm to find the parameter set  $\Theta$  that minimizes (11).
  - b. The validation set is used to check that the neural network can generalize to data that are not in the training set.
2. *Set hyperparameters*: loss function, optimizer, learning rate, number of epochs, batch size, number of hidden layers, number of nodes per hidden layer, and activation functions.
  - a. loss function: The function that is to be minimized given the training data set, see (11).
  - b. optimizer: The algorithm that is used to try and find the global minimum of the loss function.
  - c. learning rate: The initial step size that the optimization algorithm takes. Depending on the optimization algorithm, the step sizes may change between steps.
  - d. number of epochs: The number of times the optimization algorithm goes through the entire training data set.
  - e. batch size: The number of samples from the training data that will propagate through the network for each update of the parameters.
  - f. number of hidden layers:  $K$ .
  - g. number of nodes per each hidden layer:  $w_k$ ,  $k = 1, \dots, K$ .
  - h. activation functions:  $\sigma_k$ ,  $k = 1, \dots, K + 1$ .
3. Randomly initialize the parameter set:  $\Theta^{(0)}$ .
4. *Iterate*: For  $e = 1, 2, \dots$ , number of epochs
  - a. Randomly split the training set into  $\beta$  separate batches
  - b. *Iterate*: For batch=1,2,..., $\beta$ 
    - Update the neural network parameters using one step of the optimization algorithm.
  - c. With the current parameter set  $\Theta^{(e)}$ , evaluate the output of  $N_{\Theta^{(e)}}$  (see formula (12)) applied to all of the input values  $\Phi^{(m_l)}$ ,  $l = 1, 2, \dots, M_{val}$ , in the validation

set, and calculate the loss value  $L(\Theta^{(e)})$  defined in (11). If this loss value is smaller than at the end of every other epoch before it, set  $\Theta = \Theta^{(e)}$ .

5. *Retrieve parameters*: Return  $N_\Theta$  with the parameter set  $\Theta$  obtained in step 3c.

**Remark** To apply the neural network to any input matrix  $\Phi$ , one needs to reshape the matrix into a vector of size  $N'_x N'_t$ . One can then reshape the vector output to be a matrix of the size of  $\Psi$ , which is  $N_x \times N_t$ .

**Remark** The training set is usually about 80% of the total data set and the other 20% is the validation set.

**Remark** Finding good hyperparameters is often done by a running multiple trials of the training algorithm using different hyperparameters and identifying which one results in the best outcome. The specific hyperparameters we have used are presented in Section 4.

## 4 Numerical Results

In this section, we conduct a number of numerical experiments to demonstrate the performance of the machine learning approach to detect the lacunae. All of our models are built and trained using the open source library PyTorch [18].

To define the set  $Q_{f_i^{(m)}}$  in (9), we draw uniformly distributed values  $x_i^{(m)} \sim \mathcal{U}(a_1, b_1)$ ,  $t_i^{(m)} \sim \mathcal{U}(0, T_1)$ , and  $r_i^{(m)} \sim \mathcal{U}(0, R)$  for  $i = 1, 2, \dots, I^{(m)}$ , and then let

$$Q_{f_i^{(m)}} = \{(\mathbf{x}, t) \in [a_1, b_1] \times [0, T_1] : (\mathbf{x} - \mathbf{x}_i^{(m)})^2 + (t - t_i^{(m)})^2 \leq (r_i^{(m)})^2\}. \quad (13)$$

Recall that  $I^{(m)}$  is the number of sets on the right hand side of (9). As mentioned in Algorithm 1, this integer is randomly generated, and in our numerical examples it is an integer between 1 and 4. If  $I^{(m)} > 1$ , then we may have a disconnected set  $Q_{f^{(m)}}$ .

For all of the examples, the computational domains are  $\Omega = [-20, 20] \times [0, 20]$  and  $Q = [-10, 10] \times [0, 10]$ . We discretize  $\Omega$  and  $Q$  such that  $N_x = 64$ ,  $N_t = 64$ ,  $N'_x = 32$ , and  $N'_t = 32$ . The max radius in (13) we use is  $R = 5$ . We generated  $M = 10,000$  data pairs for our training process, randomly splitting the data such that  $M_{tr} = 8,000$  and  $M_{val} = 2,000$ . For the hyperparameters in Algorithm 2 we use:

- a.  $L(\Theta) = \frac{1}{M} \sum_{m=1}^M \|\Psi^{(m)} - N_\Theta(\Phi^{(m)})\|_F$ , where  $\|\cdot\|_F$  is the Frobenius norm.
- b. The Adam optimizer [19].
- c. Learning rate =  $10^{-4}$ .
- d. Number of epochs = 200.
- e. Batch size = 32.
- f.  $K = 3$ .
- g.  $w_k = 256$  for all  $k = 1, \dots, K$ .

h.  $\sigma_k = \text{LeakyReLU}$  [18] for  $k = 1, \dots, K$  and  $\sigma_{K+1} = \tanh$ .

Equipped with all the parameters and data, we train a neural network  $N_\Theta$  such that for a given set  $Q_f$  represented by the matrix  $\Phi$  as described in Algorithm 1, the neural network will produce

$$N_\Theta(\Phi) \approx \Psi,$$

where  $\Psi$  is the matrix representing the location of the lacunae  $\Lambda_1(Q_f)$  and its compliment  $(\Lambda_1(Q_f))^c$ .

#### 4.1 Example 1: Case $I^{(m)} = 1$

In the first example, we consider a particular case where  $I^{(m)} = 1$  for each  $m = 1, \dots, M$ , and  $M = 10,000$ . After training the neural network,  $N_\Theta$ , we apply it to a test set that is independent of the training and validation sets. For a given input matrix  $\Phi$ , let  $\Psi^{ref}$  denote the reference solution matrix with elements  $\psi_{j,n}^{ref} \in \{-1, 1\}$ ,  $j = 1, \dots, N_x$ ,  $n = 1, \dots, N_t$ , determining whether or not the node  $(x_j, t_n)$  is in the lacuna, see (10). Then for the neural network solution,  $\Psi^{NN} := N_\Theta(\Phi) \in (-1, 1)$ , we state that it correctly identifies that the node  $(x_j, t_n)$  is in the right set,  $\Lambda_1(Q_f)$  or  $\Lambda_1(Q_f)^c$ , if

$$(\psi_{j,n}^{NN} \leq 0 \text{ and } \psi_{j,n}^{ref} = -1) \quad \text{or} \quad (\psi_{j,n}^{nn} > 0 \text{ and } \psi_{j,n}^{ref} = 1),$$

and incorrectly identifies which set the node is in if

$$(\psi_{j,n}^{NN} > 0 \text{ and } \psi_{j,n}^{ref} = -1) \quad \text{or} \quad (\psi_{j,n}^{nn} \leq 0 \text{ and } \psi_{j,n}^{ref} = 1).$$

We can then determine how accurate the neural network is by calculating

$$accuracy = \frac{\# \text{ of nodes correctly identified}}{\text{total \# of nodes}}. \quad (14)$$

From 1,000 test cases, the neural network was able to identify which set each node belonged to with approximately 99.12% accuracy. Figure 4 shows an examples taken from the test set. The top left represents the values for the reference solutions  $\Psi^{ref}$ , the top right represents the values of the neural network solution  $\Psi^{NN}$ , the bottom left shows nodes in the set  $Q_f$ , and the bottom right represents the difference  $\Psi^{ref} - \Psi^{NN}$ . For the top graphs, the values range from  $[-1, 1]$  with  $-1$  indicating nodes in  $\Lambda_1(Q_f)$  with blue dots and  $1$  indicating nodes in  $\Lambda_1(Q_f)^c$  with red dots. For the reference solution, the element values are either  $-1$  or  $1$  so all the nodes are either dark blue or dark red respectively. The neural network has values between  $-1$  and  $1$ , thus the node colors in its graph might be different shades of red, blue, and white. On the plot for  $\Psi^{ref} - \Psi^{NN}$ , the white nodes indicates that the two solution are close to each other, the red nodes indicates that the  $\Psi^{ref}$  is greater than  $\Psi^{NN}$  and the blue nodes indicate  $\Psi^{ref}$  is less than  $\Psi^{NN}$ . Note that, the interior of the

sets  $\Lambda_1(Q_f)$  and  $\Lambda_1(Q_f)^c$  for the neural network solution are clearly defined as very close to -1 or 1, but there is some uncertainty from the neural networks along the boundary. It is expected that the neural network would in general have more difficulty learning the edges of these sets.

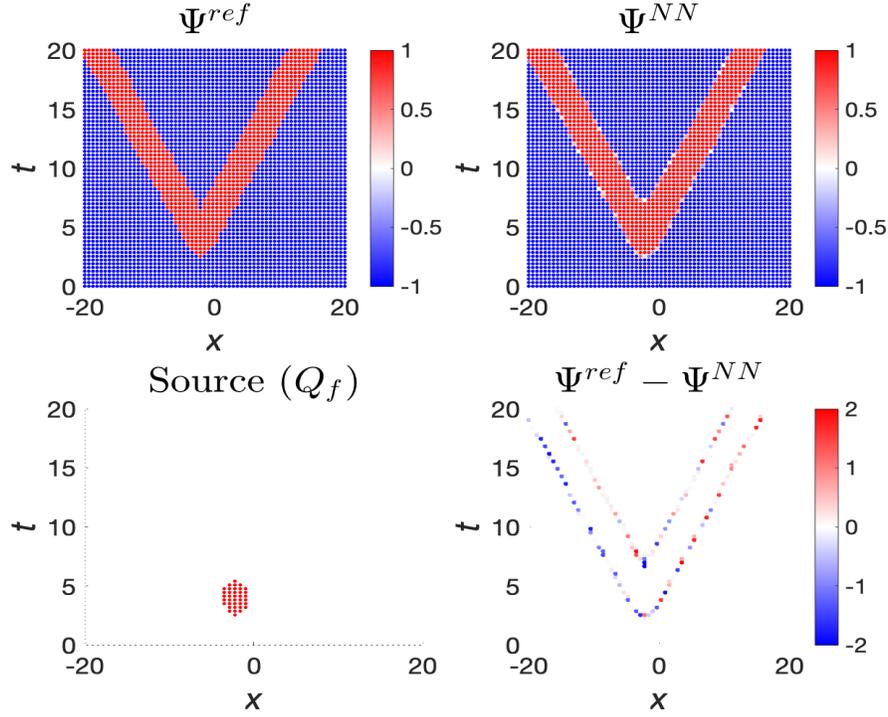


Fig. 4: Reconstruction of the shape of the lacuna (5) by neural network (12) in the case  $I^{(m)} = 1$ . Top left: Reference solutions  $\Psi^{ref}$ . Top right: Neural Network solutions  $\Psi^{NN}$ . Bottom left: The sets  $Q_f$ . Bottom right:  $\Psi^{ref} - \Psi^{NN}$ .

#### 4.2 Example 2: Case $1 \leq I^{(m)} \leq 4$

In this case, we generate our data choosing  $I^{(m)}$  to be a random integer such that  $1 \leq I^{(m)} \leq 4$  for each  $m = 1, \dots, M$ . Once trained, the neural network was able to predict which set,  $\Lambda_1(Q_f)$  or  $(\Lambda_1(Q_f))^c$ , each node belongs to with approximately 98.55% accuracy over 1,000 test cases where the accuracy is calculated as in (14). Figure 5 shows 2 examples (using the same layout as in Figure 4). The first example is taken from the test set, and in the second example we chose  $Q_f$  such that the

lacuna (5) has a “pocket,” i.e., a fully enclosed area. It is a part of the secondary lacuna (6).

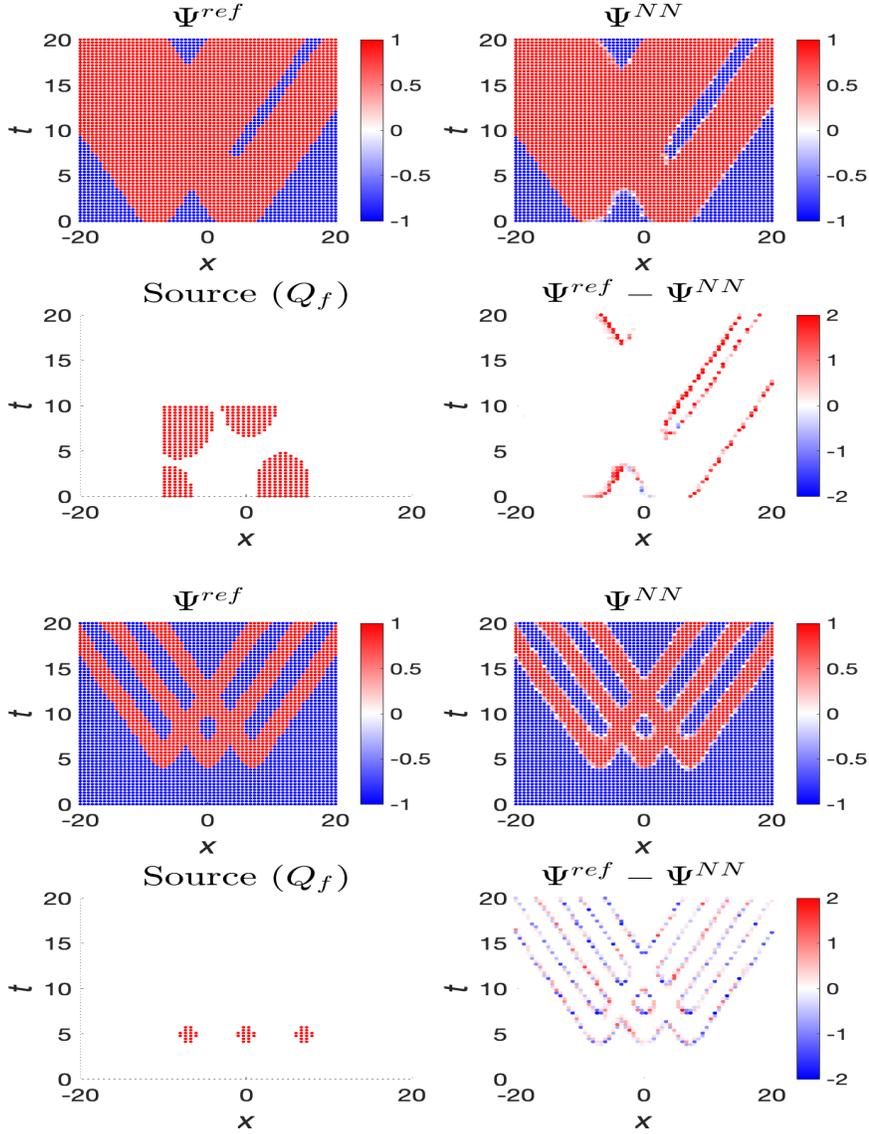


Fig. 5: Reconstruction of the shape of the lacuna (5) by neural network (12) in the case  $1 \leq I^{(m)} \leq 4$  for two different examples. With the same layout as in Figure 4, the top two rows show an example from the test data set and the bottom two rows show a hand crafted example such that the lacunae has a 'pocket'.

## 5 Discussion

We have demonstrated that a fully connected neural network can accurately reconstruct the shape of the lacunae in an artificial one-dimensional setting introduced in Section 1. While we have trained our network to find the shape of a combined lacuna (5), we anticipate that having it identify only the secondary lacunae (6) would not present any additional issues. A challenging next step is to extend the proposed machine learning approach to a realistic three-dimensional setting where the secondary lacunae are defined according to (3) and account for the actual physics of the solutions to the wave equation (1).

**Acknowledgements** The work of A. Chertock and C. Leonard was supported in part by NSF Grant DMS-1818684. The work of S. Tsynkov was partially supported by the US-Israel Binational Science Foundation (BSF) under grant # 2020128.

## References

1. I. Petrowsky, *Matematicheskii Sbornik (Recueil Mathématique)* **17 (59)**(3), 289 (1945)
2. V.S. Vladimirov, *Equations of Mathematical Physics* (Dekker, New-York, 1971)
3. R. Courant, D. Hilbert, *Methods of Mathematical Physics. Volume II* (Wiley, New York, 1962)
4. J. Hadamard, *Lectures on Cauchy's Problem in Linear Partial Differential Equations* (Yale University Press, New Haven, 1923)
5. J. Hadamard, *Problème de Cauchy* (Hermann et cie, Paris, 1932). [French]
6. J. Hadamard, *Ann. of Math. (2)* **43**, 510 (1942)
7. M.F. Atiyah, R. Bott, L. Gårding, *Acta Math.* **124**, 109 (1970)
8. M.F. Atiyah, R. Bott, L. Gårding, *Acta Math.* **131**, 145 (1973)
9. M. Matthiesson, *Acta Math.* **71**, 249 (1939). [French]
10. K.L. Stellmacher, *Nachr. Akad. Wiss. Göttingen. Math. Phys. Kl. Math.-Phys. Chem. Abt.* **1953**, 133 (1953). [German]
11. J.E. Lagnese, K.L. Stellmacher, *J. Math. Mech.* **17**, 461 (1967)
12. K.L. Stellmacher, *Math. Ann.* **130**, 219 (1955). [German]
13. R. Schimming, in *Proceedings of the Joint IUTAM/IMU Symposium "Group-Theoretical Methods in Mechanics"*, ed. by N.H. Ibragimov, L.V. Ovsyannikov (USSR Acad. Sci., Siberian Branch, Institute of Hydrodynamics — Computing Center, USSR, Novosibirsk, 1978), pp. 214–225
14. M. Belger, R. Schimming, V. Wünsch, *Z. Anal. Anwendungen* **16**(1), 9 (1997). Dedicated to the memory of Paul Günther
15. P. Günther, *Huygens' principle and hyperbolic equations, Perspectives in Mathematics*, vol. 5 (Academic Press Inc., Boston, MA, 1988). With appendices by V. Wünsch
16. P. Günther, *Arch. Rational Mech. Anal.* **18**, 103 (1965). [German]
17. P.D. Lax, R.S. Phillips, *Comm. Pure Appl. Math.* **31**(4), 415 (1978)
18. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimeshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, in *Advances in Neural Information Processing Systems 32* (Curran Associates, Inc., 2019), pp. 8024–8035. URL <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
19. D.P. Kingma, J. Ba, arXiv e-prints arXiv:1412.6980 (2014)