Applications of deep learning in numerical solutions of partial differential equations

Journal of Computational Physics 378 (2019) 686–707

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

M. Raissi [a], P. Perdikaris [b,*], G.E. Karniadakis [a]

[a] Division of Applied Mathematics, Brown University, Providence, RI, 02912, USA
[b] Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA, 19104, USA

Physics-informed Neural Network

Example: Solving a nonlinear Schrödinger equation with periodic boundary conditions

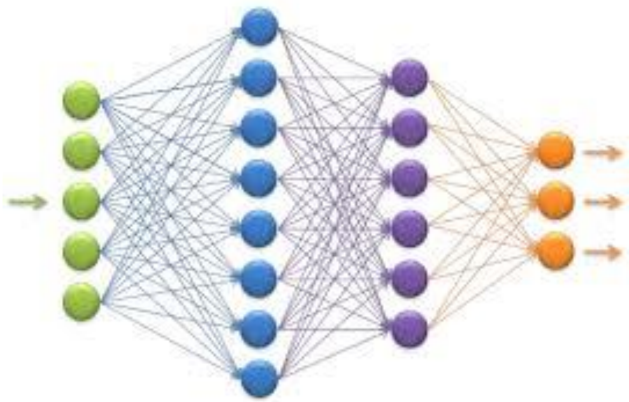
$$ih_t + 0.5h_{xx} + |h|^2 h = 0\ , \qquad x \in [-5,5],\ \ t \in [0, \pi/2],$$

$$h(x,0) = 2\operatorname{sech}(x), \qquad h(-5,t) = h(5,t),\ \ h_x(-5,t) = h_x(5,t).$$

Input to the network – any set of values (x,t),

Network - 5 FC hidden layers, 100 neurons per layer,

Output layer $\tilde{h}(x,t) = [\tilde{u}(x,t), \tilde{v}(x,t)]$.



Training loss function I

Define $\tilde{f} := i\tilde{h}_t + 0.5\tilde{h}_{xx} + \left|\tilde{h}\right|^2 \tilde{h}$

Physics-informed loss - $N_f = 20,000$ randomly selected collocation points $\left(x_i, t_i\right)$ inside the domain

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\tilde{f}\left(x_i, t_i\right)\right|^2.$$

Loss at t=0 - $N_0 = 50$ points $\left\{x_j\right\}$, $-5 < x_j < 5$,

$$MSE_0 = \frac{1}{N_0}\sum_{j=1}^{N_0}\left|\tilde{h}\left(x_j, 0\right) - h\left(x_j, 0\right)\right|^2$$

Training loss function II

Boundary loss - $N_b = 50$ time samples $\{t_k\}$

$$MSE_b = \frac{1}{N_b}\sum_{k=1}^{N_b}\left\{\left|\tilde{h}(-5,t_k)-\tilde{h}(5,t_k)\right|^2+\left|\tilde{h}_x(-5,t_k)-\tilde{h}_x(5,t_k)\right|^2\right\}$$

The network is trained with aggregated loss

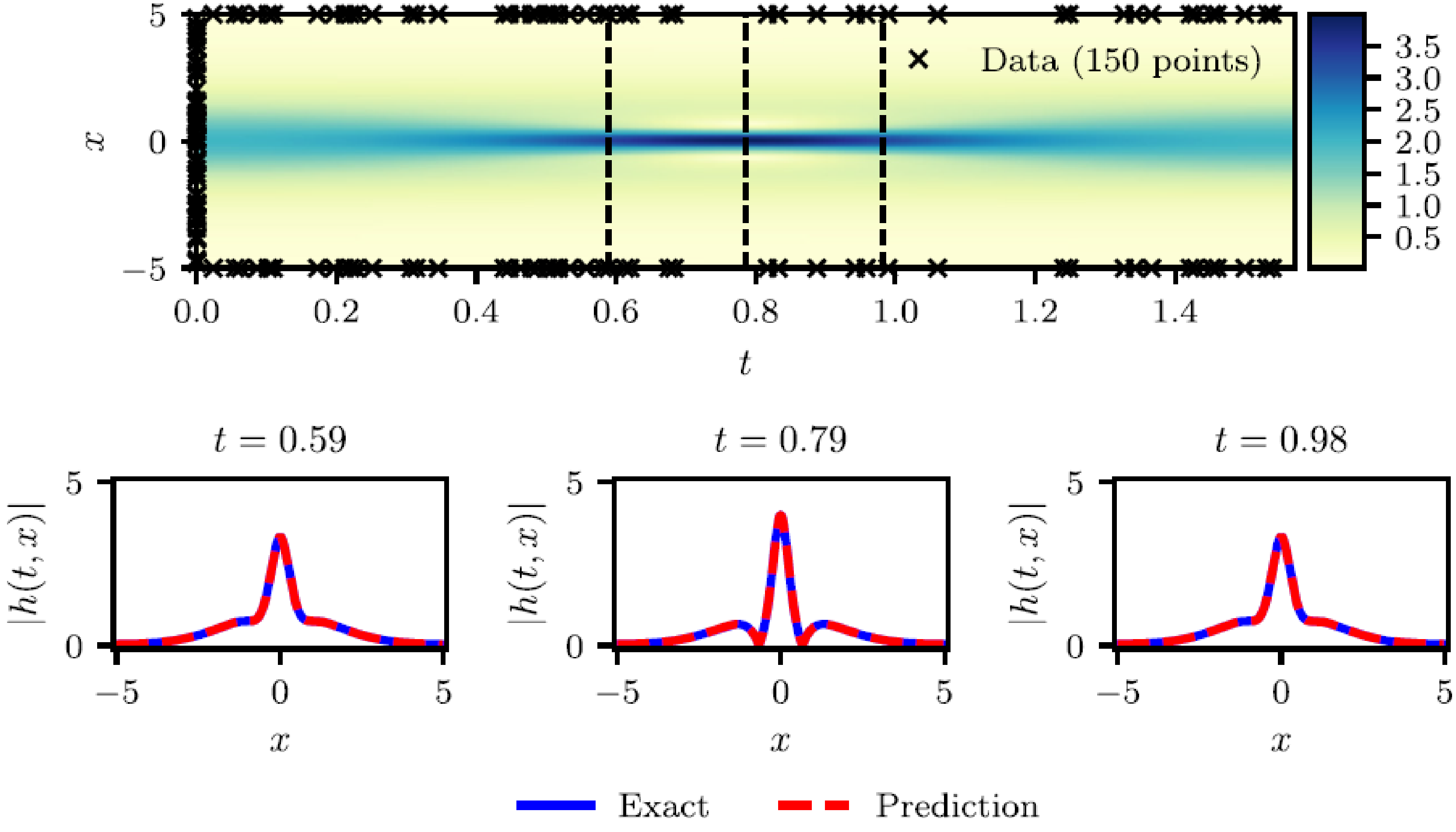
$$MSE = MSE_f + MSE_0 + MSE_b$$

Notice that MSE_f requires "automatic differentiation" of the neural network by the parameters x, t.

```
def f(t, x):
    u = u(t, x)
    u_t = tf.gradients(u, t)[0]
    u_x = tf.gradients(u, x)[0]
    u_xx = tf.gradients(u_x, x)[0]
```

Physics-informed Neural Network

$$\left|\tilde{h}(x,t)\right|$$



Physics-informed NN

"We must note however that the proposed methods should not be viewed as replacements of classical numerical methods for solving partial differential equations (e.g., finite elements, spectral methods, etc.). Such methods have matured over the last 50 years and, in many cases, meet the robustness and computational efficiency standards required in practice. Our message … is that classical methods … can coexist in harmony with deep neural networks"

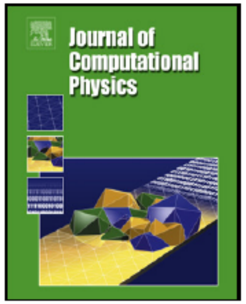
M. Raissia, P.Perdikarisb and G.E.Karniadakisa, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707

Journal of Computational Physics 413 (2020) 109458

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

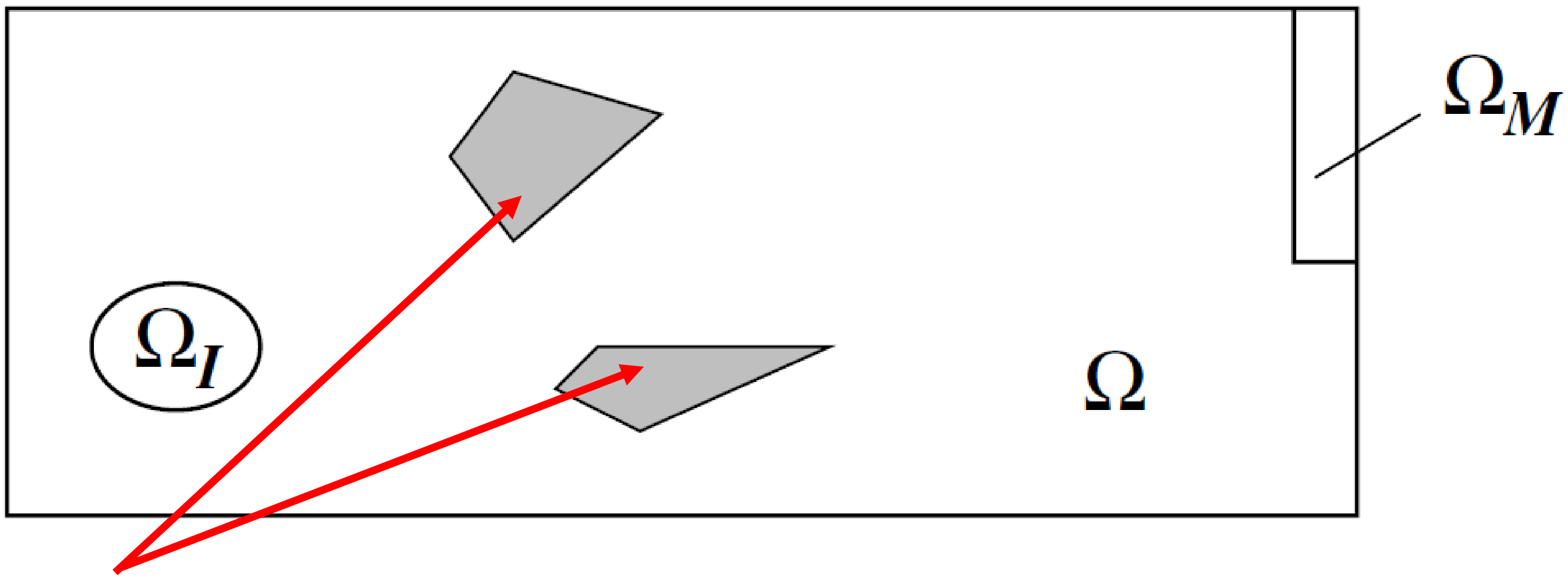


Obstacle segmentation based on the wave equation and deep learning

Adar Kahana [a,*], Eli Turkel [a], Shai Dekel [a], Dan Givoli [b]

[a] Department of Applied Mathematics, Tel-Aviv University, Tel-Aviv 69978, Israel
[b] Department of Aerospace Engineering, Technion, Haifa 32000, Israel

Numerical methods for the wave equation



Obstacles (scatters)

Ω_M - Sensor domain

Ω_I - Source location

Basic notation

$$u(x,t):\Omega\times[0,T]\to\mathbb{R},\quad \Omega\subset\mathbb{R}^n,\qquad \nabla u := \nabla_x u = \left(\frac{\partial u}{\partial x_1},\dots,\frac{\partial u}{\partial x_n}\right)$$

Vector field $\quad F(x) = \left(F_1(x),\dots,F_n(x)\right):\mathbb{R}^n\to\mathbb{R}^n$

Divergence $\quad div(F) = \langle\nabla, F\rangle := \sum_{i=1}^{n}\frac{\partial F_i}{\partial x_i}$

Laplace operator $\quad \Delta u := div(\nabla u) = \sum_{i=1}^{n}\frac{\partial^2 u}{\partial x_i^2},$

Discrete Laplace as convolution ($n=2$), $\Delta_d = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$

Numerical methods for the wave equation

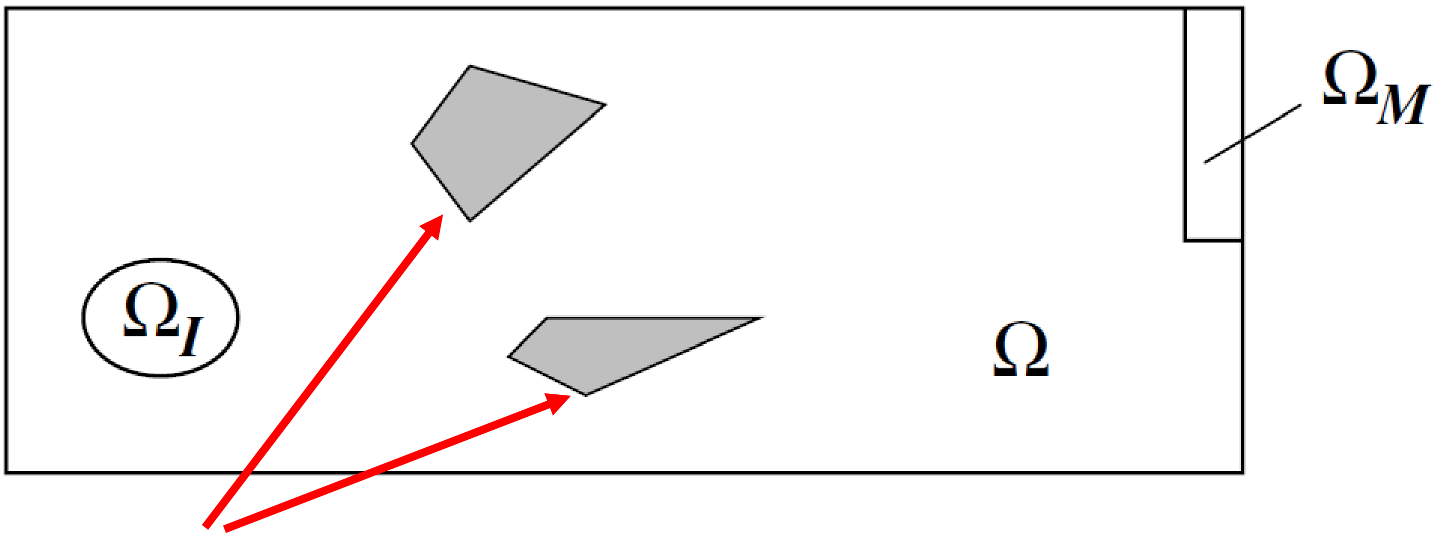
$$\frac{\partial^2 u}{\partial t^2}(x,t) = div\left(c^2(x)\nabla u(x,t)\right),\ 0 \le t \le T,\ x \in \Omega.$$

$$u(x,t) = 0,\quad \forall x \in \partial\Omega, \qquad u(x,0) = u_0(x),\ \frac{\partial u}{\partial t}(x,0) = v_0(x).$$

Finite difference scheme (c constant, $\Delta_d(x,j)$ the discrete Laplace, $\Delta t = \Delta x = 1$)

$$u(x,j+1) = c^2\Delta_d u(x,j) + 2u(x,j) - u(x,j-1).$$

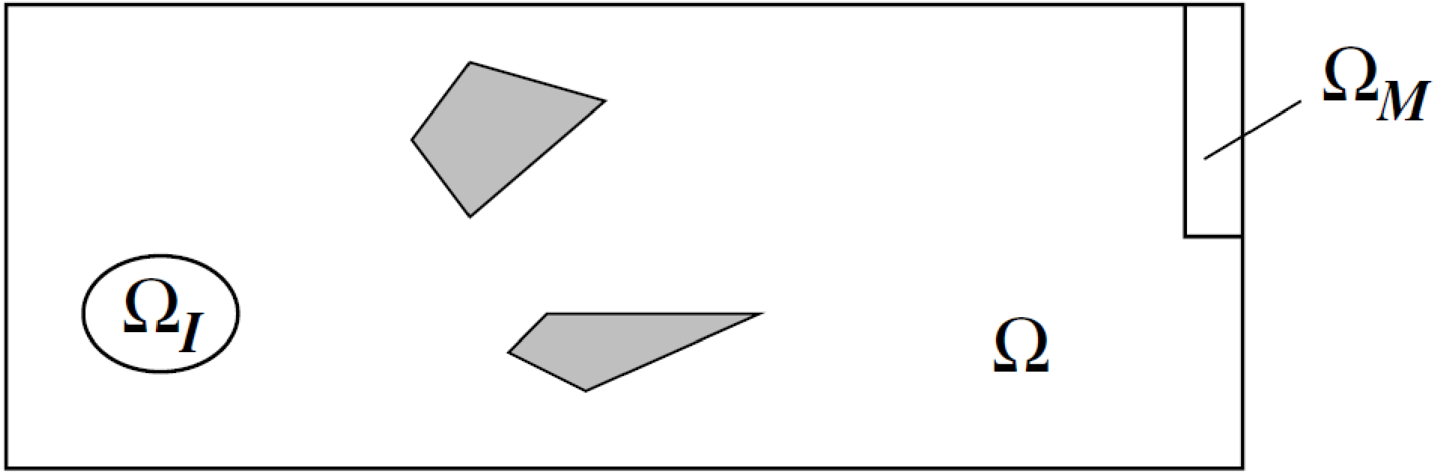
Source location through time reversal re-focusing



Input: Obstacle locations, $u(x,T), u_t(x,T),\ x \in \Omega_M$.

Goal: find Ω_I, the unknown source location

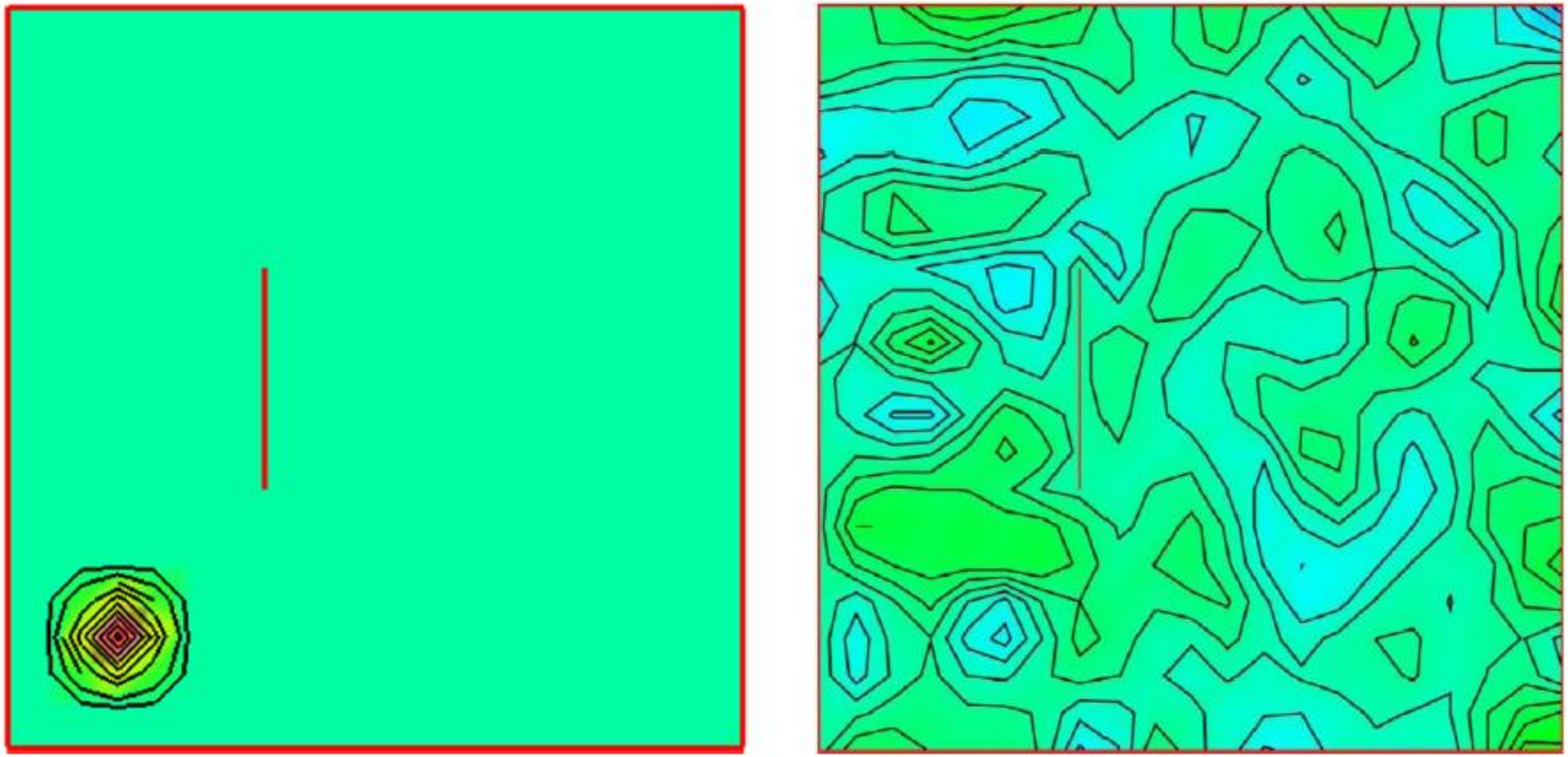
Source location through time reversal re-focusing



Method (*): Reversed finite difference scheme for the wave equation backwards from time T to time 0 based only on information from Ω_M at time T. This means initializing $u(x,T)=0$ for $x \in \Omega \setminus \Omega_M$.

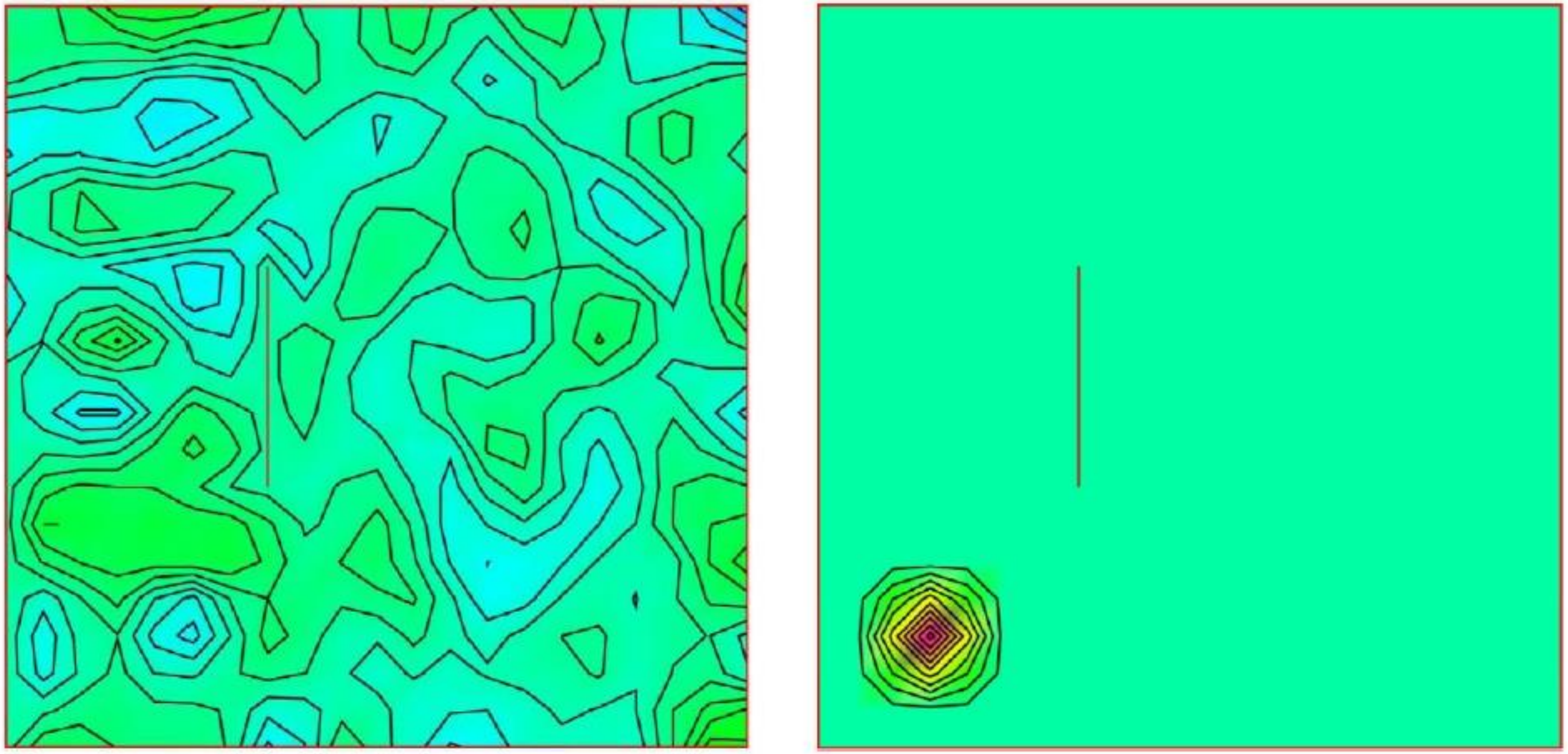
(*) D. Givoli, Time Reversal as a Computational Tool in Acoustics and Elastodynamics, Journal of Computational Acoustics, 22 (2014).

Example for forward propagation



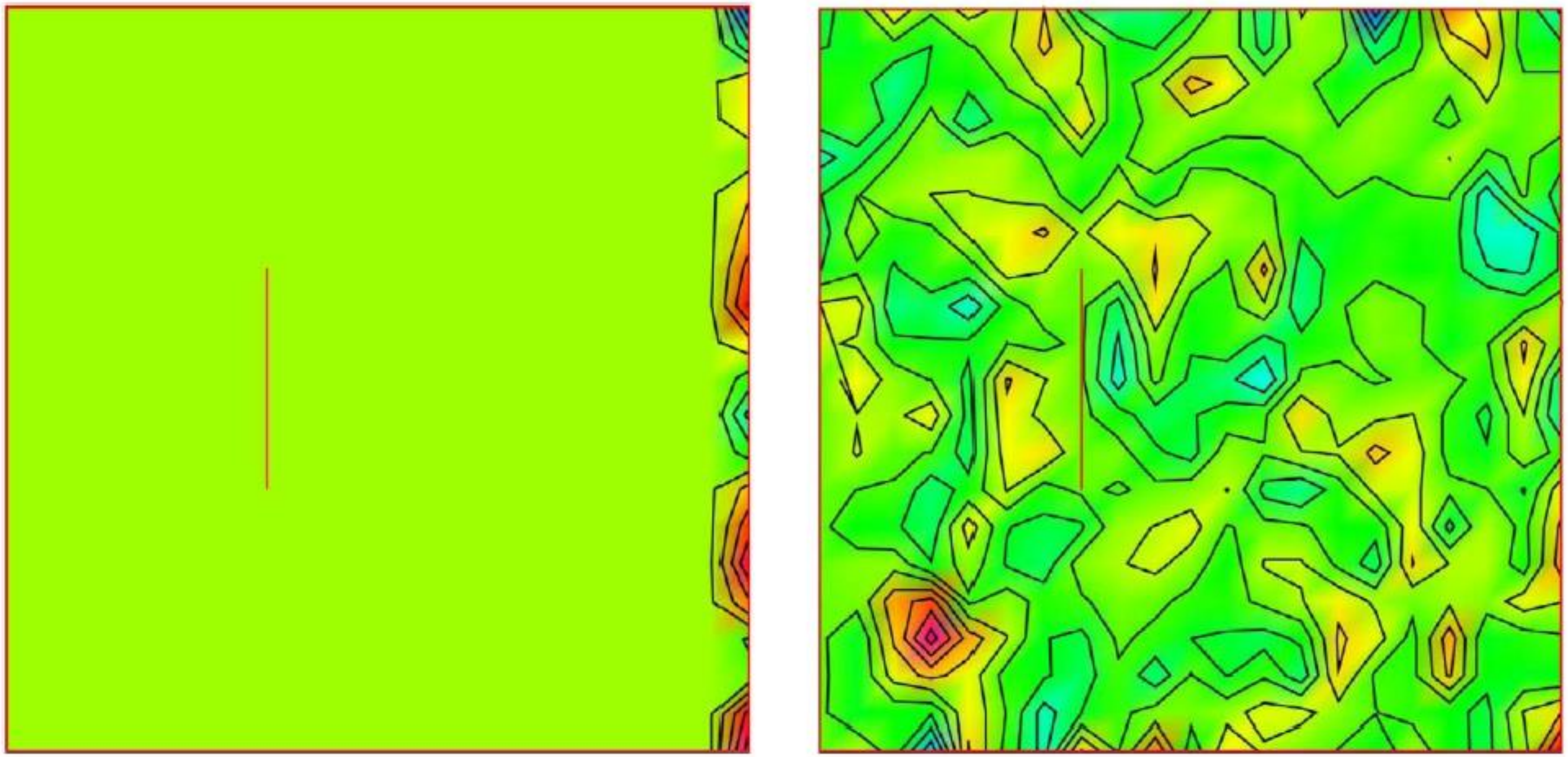
(a) (b)

Example for time reversal with $\Omega_M = \Omega$



(a) (b)

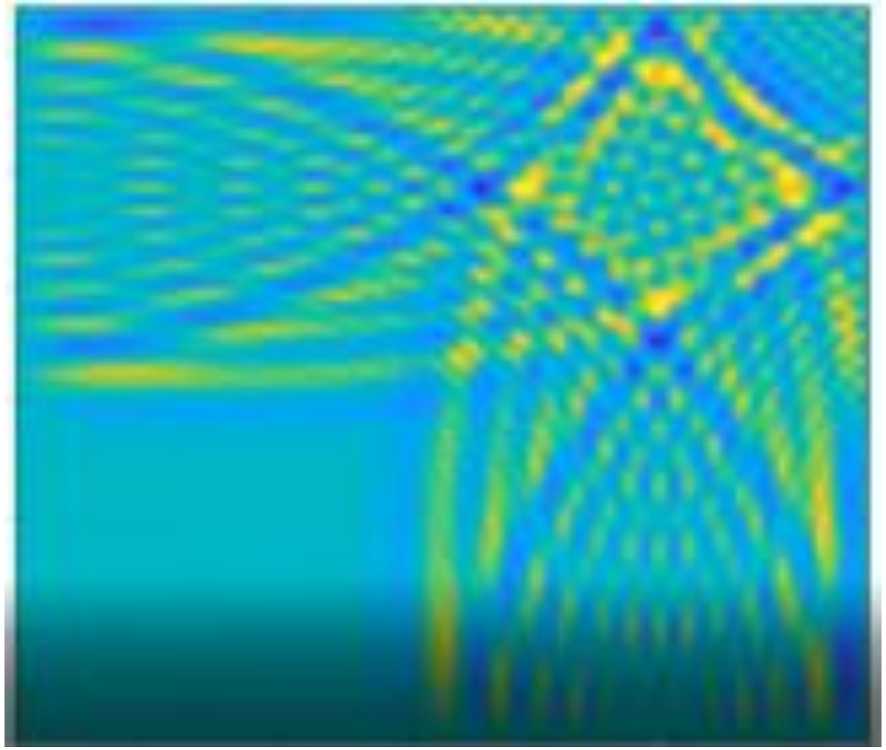
Example for time reversal with $\Omega_M \subset \Omega$



(a) (b)

Source location - AI-based toy problem

Input – <u>only</u> $u(x,T)$. $u_t(x,T)$ is missing.



Output - $u(x,0)$ Source location at time 0.

Source location - AI-based toy problem

$u_t\left(x,T\right)$ is missing → problem is relatively 'ill-posed'.

Method – Train a regression convolutional network using software simulations that outputs $\left(x, y\right)$ coordinates of predicted source.

Training set – 3,000 pairs of source locations and wave images at time T

Testing set – 1,000 pairs

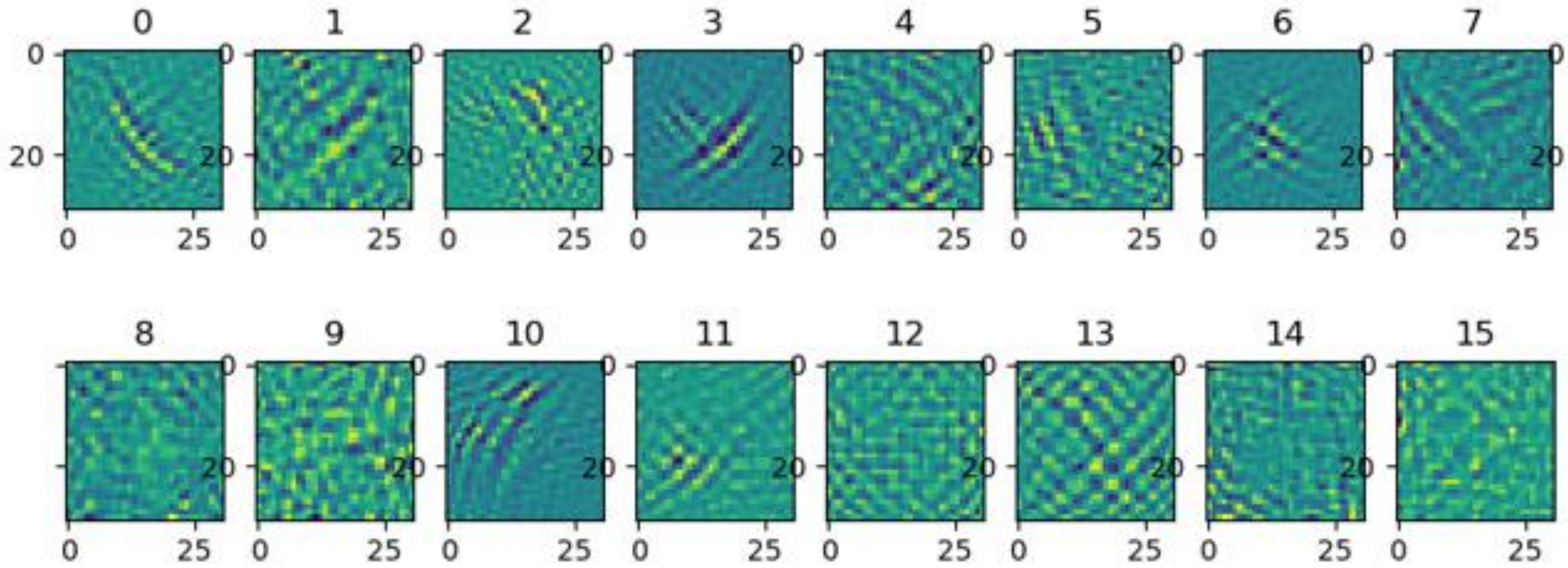
4-fold runs

Source location – Convolutional Neural Network architecture

Layer 0 (input)	128x128 image of $u(x,T)$
Layer 0 to layer 1	16 conv filters of size 31x31
Layer 1 to layer 2	32 conv filters of size 5x5
Layer 2 to layer 3	32 conv filters of size 31x31
Layer 3 to output	Fully connected
Output	predicted (x,y) coordinates of source.

Source location – trained conv filters

Visualization of the convolution filters at layer 1



Source location – results (4-fold)

Pixels range	Success rate
Exact location	36%
Up to one pixel away	72%
Up to two pixels away	87%
Up to three pixel away	93%

Obstacle segmentation

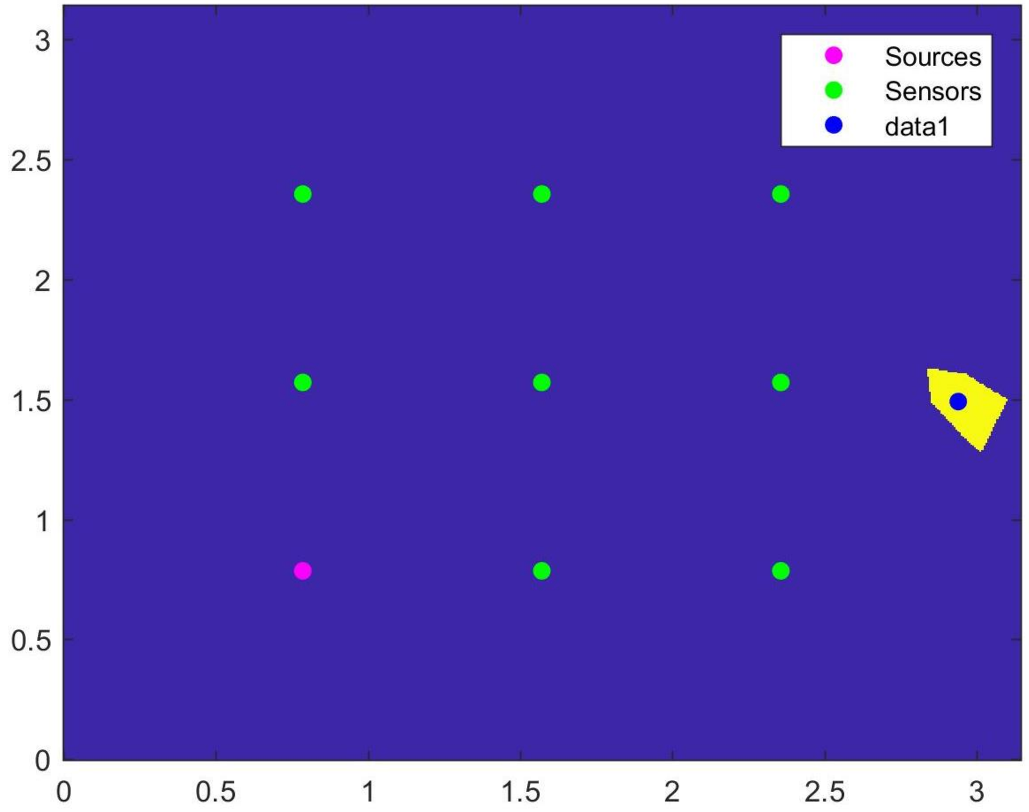
Input - sensor data $0 = t_0 < t_1 < \cdots < t_m = T$,

- source location
- boundary conditions

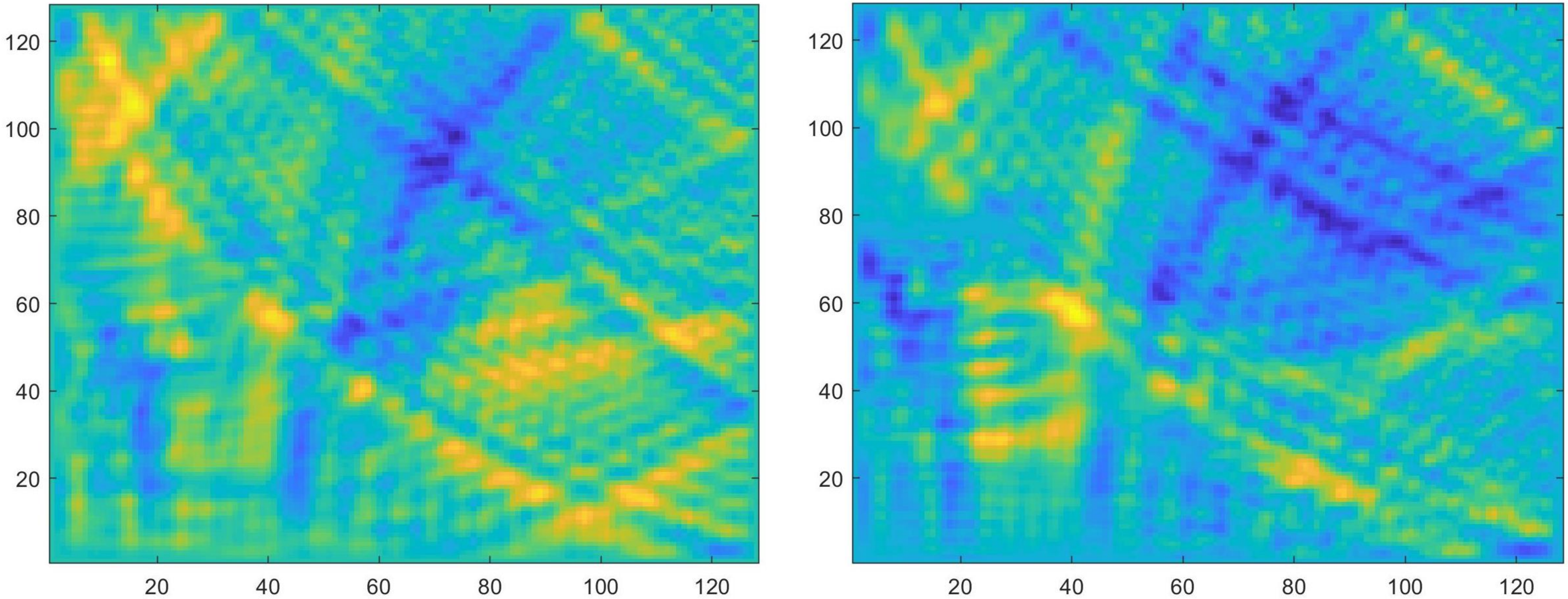
Output - obstacle segmentation map

State-of-the-art using "classical" numerical methods + optimizations: <u>unknown single circular obstacle</u>

Here we assume $c = const$ away from boundary and obstacles.



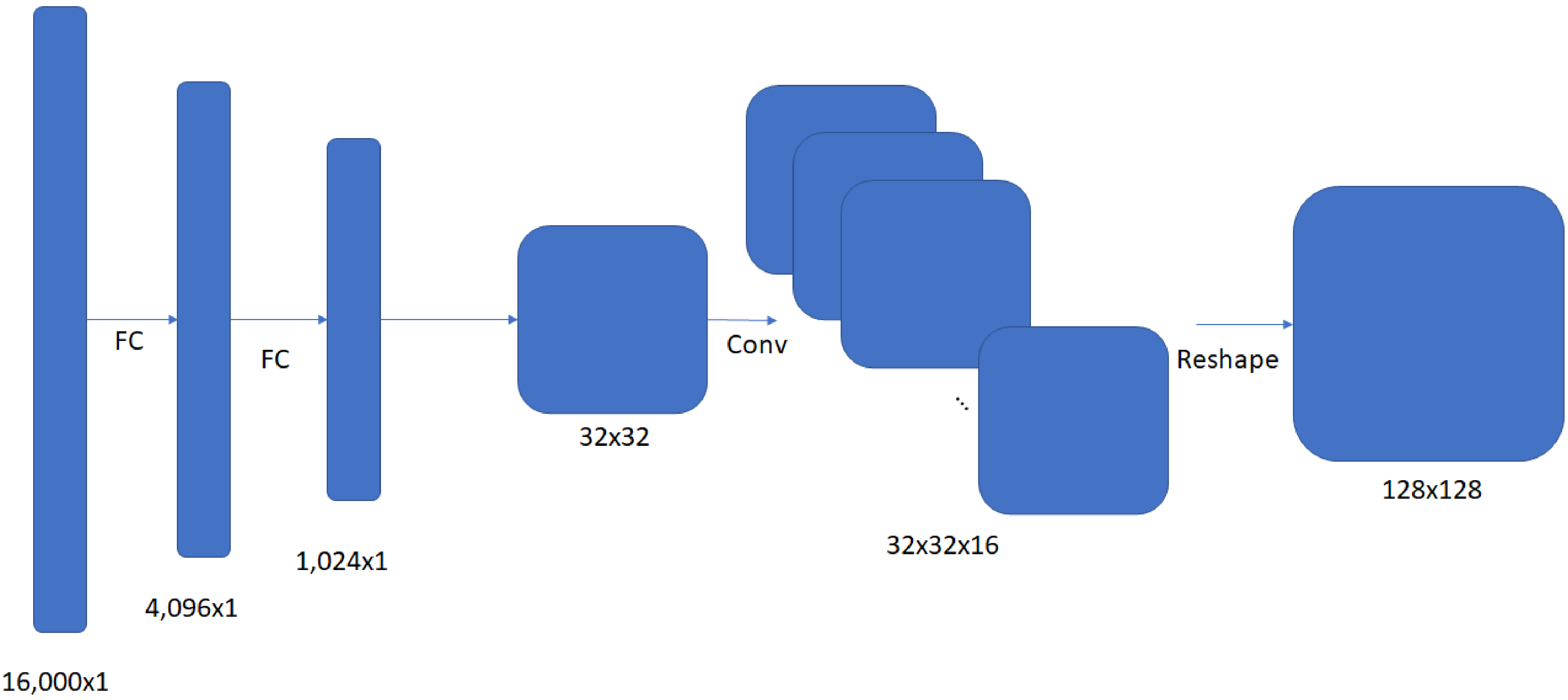
Waves at $t = 500$ with different obstacles



Obstacle segmentation – network architecture

Input: 2,000 time samples for each of the 8 sensors

Output: segmentation map of obstacle. Each pixel is logistic regression variable with value [0,1]



Obstacle segmentation – Training

Dataset of 20,000 random polygonal obstacles. Recall that during training we may use the ground truth obstacles.

For each obstacle sample I and each location $x \in [0,127]^2$

- $p_I(x) \in \{0,1\}$- ground truth.
- $p_I(x) = 1$ if location x is part of the obstacle in sample I.

$s_I \in \mathbb{R}^n$ - vector representation of sample I at the layer before last,

$$\tilde{p}_I(x) = \frac{1}{1 + e^{-(\langle W_x, s_I \rangle + b_x)}}, \qquad W_x \in \mathbb{R}^n \text{ , } b_x \in \mathbb{R},\ x \in [0,127]^2.$$

So $0 \le \tilde{p}_I(x) \le 1$, approximation for obstacle at location x.

Minimization of Negative Log-Likelihood loss

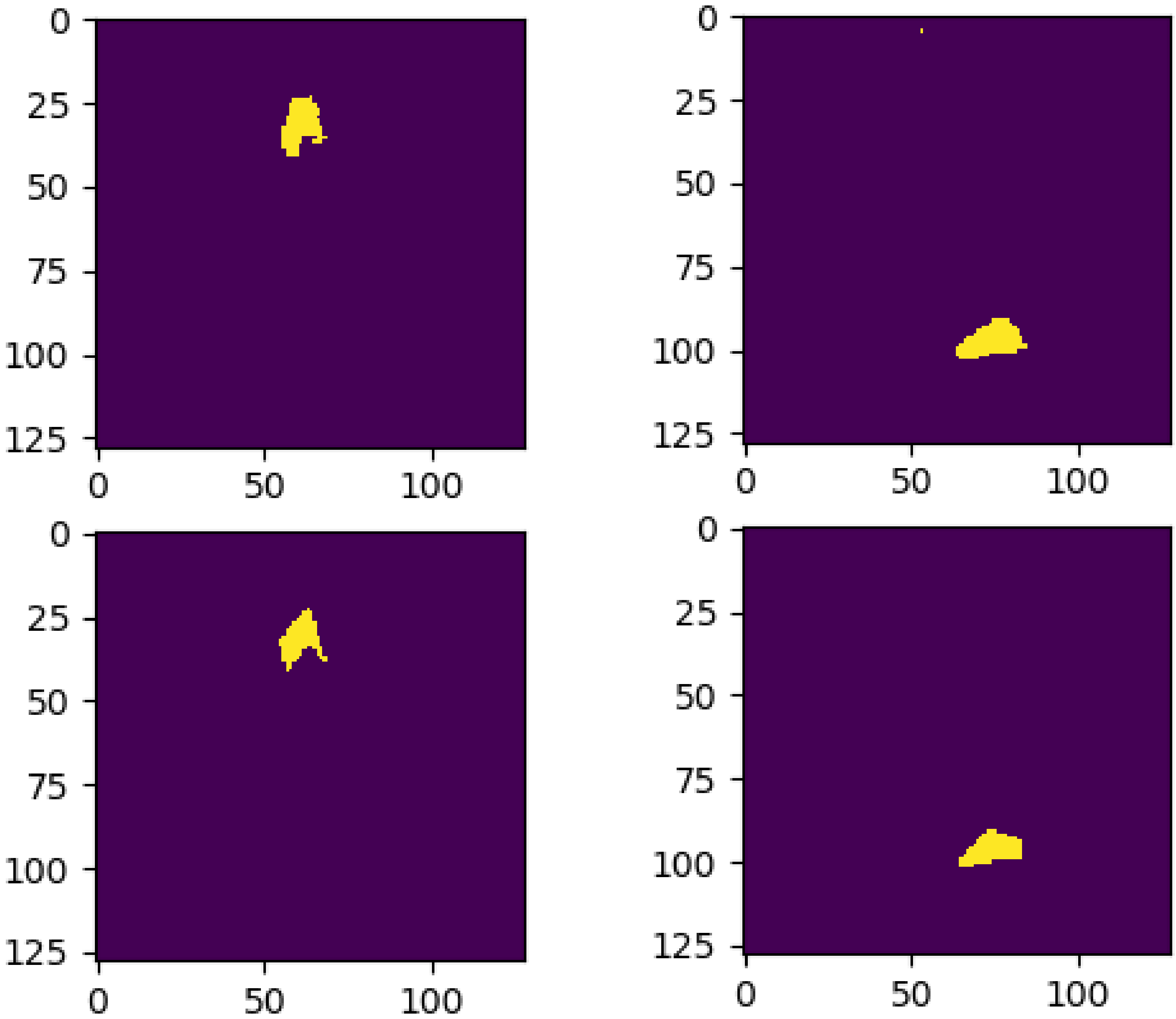
$$NLL = -\frac{1}{\# I}\frac{1}{128^2}\sum_{I}\sum_{x\in[0,127]^2} p_I(x)\log\tilde{p}_I(x) + \left(1-p_I(x)\right)\log\left(1-\tilde{p}_I(x)\right)$$

Minimization of Jaccard loss (approximates minus of Intersection Over Union (IoU))

$$-\frac{\langle p_I, \tilde{p}_I\rangle}{|p_I|_1 + |\tilde{p}_I|_1 - \langle p_I, \tilde{p}_I\rangle}$$

Obstacle segmentation – results

Prediction



Ground Truth

Obstacle segmentation – results

$$0 \leq IOU(A,B) := \frac{|A \cap B|}{|A \cup B|} \leq 1$$

Mean IOU	Median IOU
0.621	**0.66**

Physics-informed obstacle segmentation

- Regularization of the solution using the wave equation.
- The known source, sensors and output obstacle image $\{\tilde{p}(x)\}$, $0 \le \tilde{p}(x) \le 1$, provide a solution $\tilde{u}$ (function of $\tilde{p}$) to

$$\frac{\partial^2 \tilde{u}}{\partial t^2}(x,t) = c^2 div\left(\left(1-\tilde{p}(x)\right)^2 \nabla \tilde{u}(x,t)\right), \ 0 \le t \le T, \ x \in [0,127]^2.$$

- So, if the sensors are located at $\{x_k\}_{k=1}^{8}$, then

$$MSE := \frac{1}{8}\frac{1}{T}\sum_{k=1}^{8}\sum_{j=1}^{T}\left(u(x_k, j) - \tilde{u}(x_k, j)\right)^2 .$$

- Training loss - linear combination of NLL and physics informed MSE.

Physics-informed obstacle segmentation

- $\{\tilde{u}(x_k, j)\}$ are a determined by obstacle map $\tilde{p}(x)$.
- But…how do we pass this information to the loss function?
- We add to the architecture of the neural network, one additional layer per time stamp, $\tilde{u}(x, j)$, $1 \le j \le T$.
- We apply finite difference forward passes

$$\tilde{u}(x, j+1) = F(x, c, \tilde{p}, \tilde{u}(\cdot, j), \tilde{u}(\cdot, j-1)).$$

- We collect from each layer j the values $\{\tilde{u}(x_k, j)\}_{k=1}^{8}$ and pass to the loss function.

Journal of Computational Physics 442 (2021) 110493

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Beyond the Courant-Friedrichs-Lewy condition: Numerical methods for the wave problem using deep learning

Oded Ovadia *, Adar Kahana, Eli Turkel, Shai Dekel

Department of Applied Mathematics, Tel-Aviv University, Tel-Aviv 69978, Israel

ABSTRACT

We investigate a numerical method for approximating the solution of the one dimensional acoustic wave problem, when violating the numerical stability condition. We use deep learning to create an explicit non-linear scheme that remains stable for larger time steps and produces better accuracy than the reference implicit method. The proposed spatio-temporal neural-network architecture is additionally enhanced during training with a physically-informed term, adapting it to the physical problem it is approximating and thus more accurate.

3.1. Mathematical model

The general formulation of the wave problem is given by -

$$\begin{cases} \ddot{u}(\overrightarrow{x},t) = \nabla \cdot (c^2(\overrightarrow{x})\nabla u(\overrightarrow{x},t)) & \overrightarrow{x} \in \Omega,\ t \in (0,T], \\ u(\overrightarrow{x},0) = u_0(\overrightarrow{x}) & \overrightarrow{x} \in \Omega, \\ \dot{u}(\overrightarrow{x},0) = v_0(\overrightarrow{x}) & \overrightarrow{x} \in \Omega, \\ u(\overrightarrow{x},t) = f(\overrightarrow{x},t) & \overrightarrow{x} \in \partial\Omega_1,\ t \in [0,T], \\ \nabla u(\overrightarrow{x},t) = g(\overrightarrow{x},t) & \overrightarrow{x} \in \partial\Omega_2,\ t \in [0,T], \qquad \partial\Omega_1 \cup \partial\Omega_2 = \partial\Omega, \end{cases} \tag{1}$$

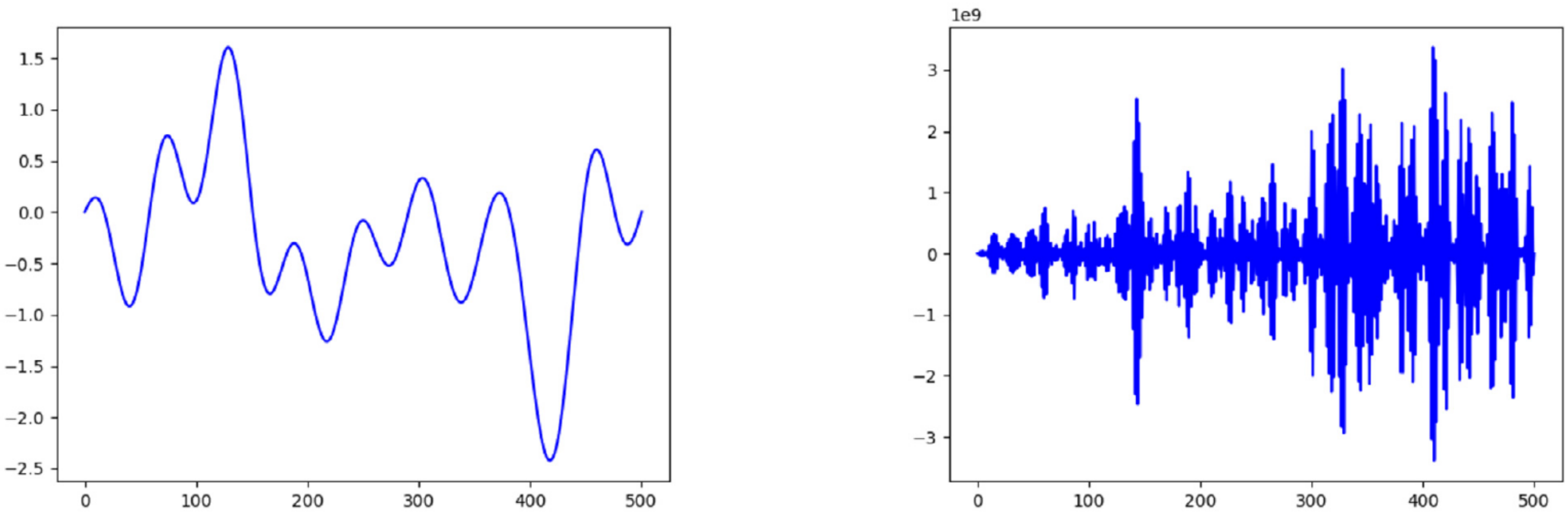
where $u(\overrightarrow{x},t)$ is the wave amplitude or acoustic pressure, c is the wave propagation speed, u_0 and v_0 are the initial pressures and velocities respectively, and f and g are boundary condition of types Dirichlet and Neumann respectively. In this work we investigate the one-dimensional case. Therefore, throughout the paper x is treated as a scalar so $x \in \Omega = [a,b]$. We also assume that c is constant. The specific system we solve is given in section 5.1 by (5). Problem (1) is well-posed and thus small changes in the problem conditions result in small changes in the solution and there exists a unique continuous solution to the problem inside the domain.

We approximate the spatial domain Ω using N_x nodes. We also approximate the temporal direction $[0, T]$ using N_t nodes. Both are uniform divisions. We denote Δx and Δt as the grid spacing. We use the notation u_i^n for the solution at point $(i \cdot \Delta x, n \cdot \Delta t)$ where $i = 0, \ldots N_x - 1$ and $n = 0, \ldots N_t - 1$. A commonly used method to approximate the solution is the FD method [5]. We use it to approximate both in time and space as follows:

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\Delta t^2} = c^2 \left(\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \right). \tag{2}$$

3.3. *Numerical stability*

Denote $\alpha = c\frac{\Delta t}{\Delta x}$. This is the Courant-Friedrichs-Lewy (CFL) number. The CFL criterion $\alpha = c\frac{\Delta t}{\Delta x} \leq 1$ is a necessary and sufficient condition for the convergence of FD method [6]. When we choose a grid (selection of $N_x, N_t, \Delta x$ and Δt) that does not satisfy the CFL condition, $\alpha > 1$, after a few iterations in time the solution "blows up" as shown in Fig. 1.



(a) $\alpha = 0.875$

(b) $\alpha = 8.75$

Fig. 1. Approximation of a solution to the wave problem after 10 iterations using the FD method with different CFL numbers.

Table 1
Network layers types, sizes and number of weights.

Layer type	Number of weights	Number of biases	Activation	Layer output shape	Number of activation weights
Convolution	32	16	PReLU	$N_x \times 1 \times 16$	8016
Convolution	6,144	128	PReLU	$N_x \times 128$	64,128
Convolution	16,384	128	PReLU	$N_x \times 128$	64,128
Convolution	128	1	-	$N_x \times 1$	0
Dense	251,001	501	-	N_x	0

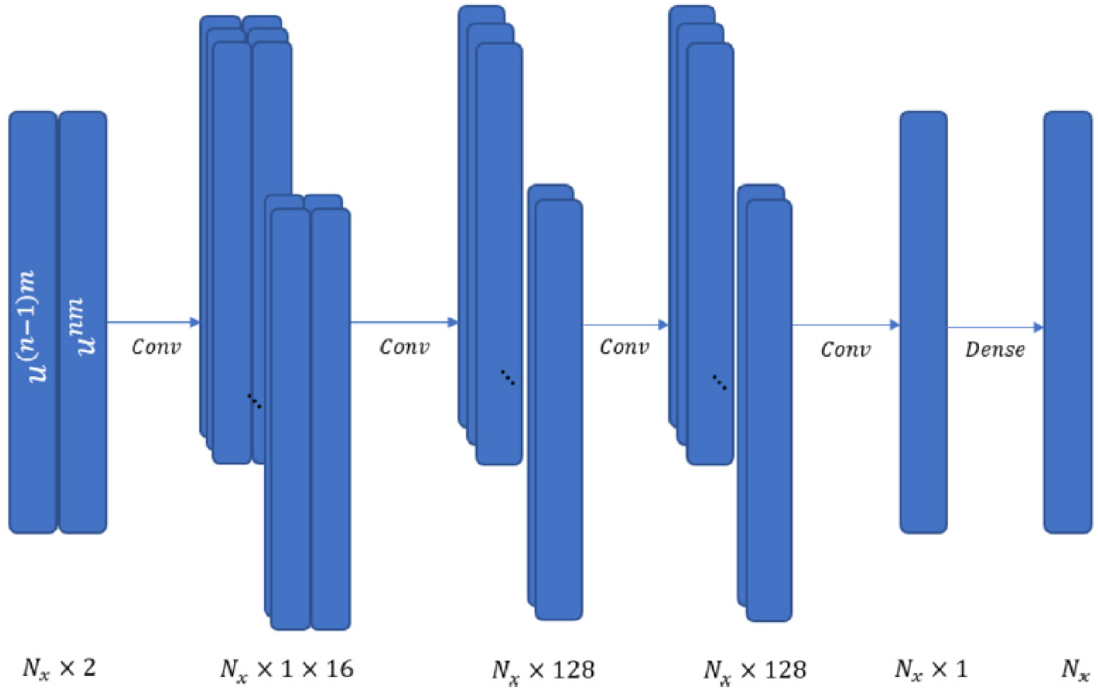
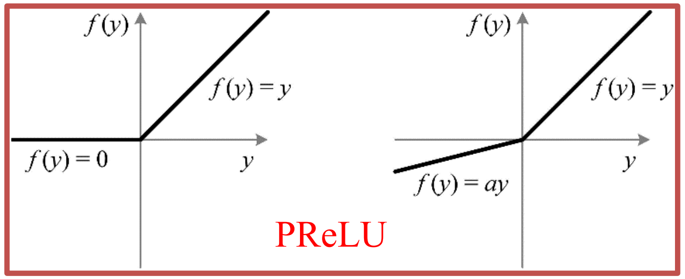


Fig. 2. The network architecture.



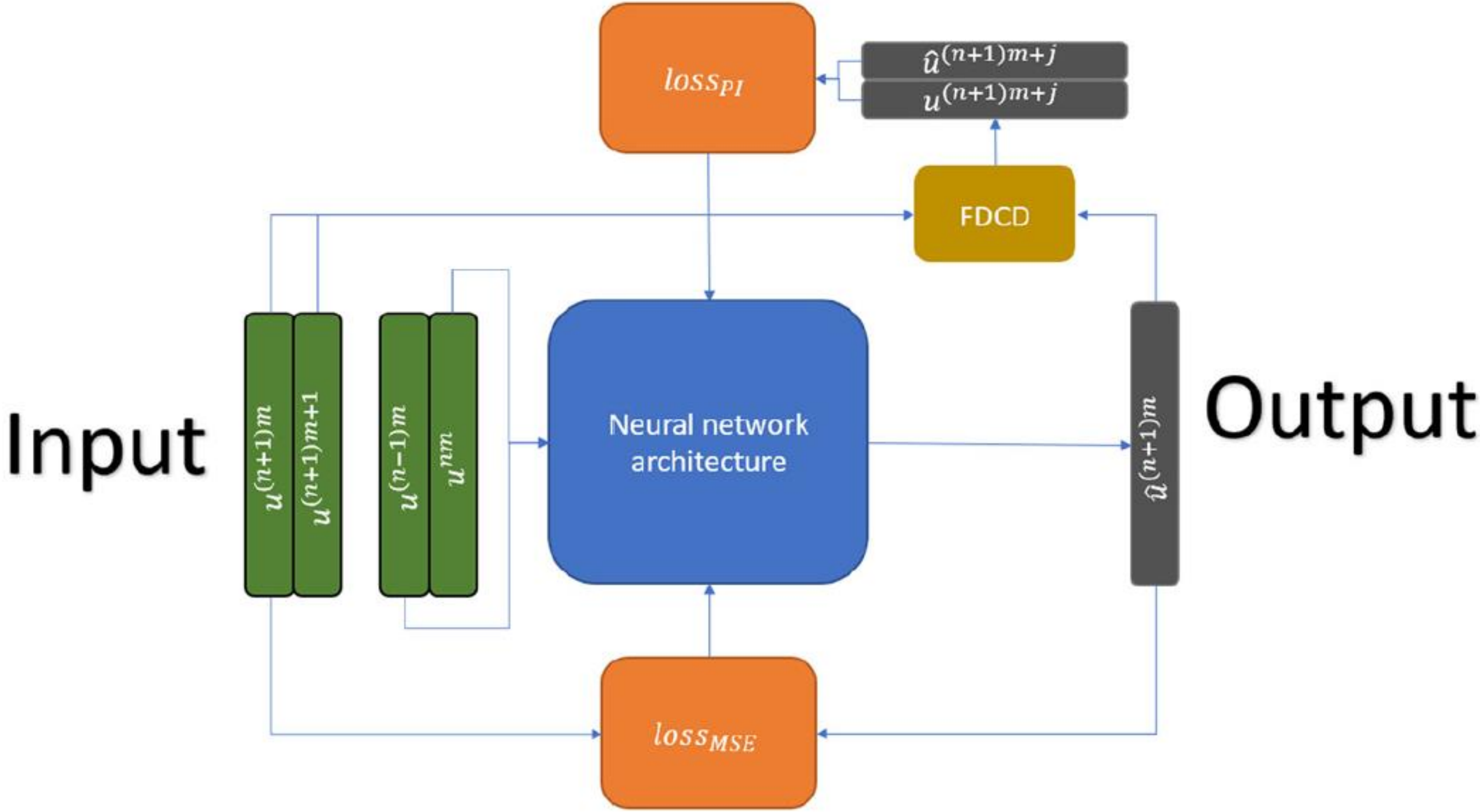


Fig. 4. Physically-informed loss calculation process.

5.1. Experiment setup

We tested the method on problem (1) setting $\Omega = [0, 1]$, $T = 1$, $c = 7$ and $v_0(x) = 0$. Problem (1) then becomes:

$$\begin{cases} \ddot{u}(x,t) = c^2 u_{xx}(x,t) & 0 \leq x \leq 1,\ 0 \leq t \leq T, \\ u(x,0) = u_0(x) & 0 \leq x \leq 1, \\ \dot{u}(x,0) = 0 & 0 \leq x \leq 1, \\ u(0,t) = u(1,t) = 0 & 0 \leq t \leq T. \end{cases} \tag{5}$$

For the numerical grid we chose $N_t = 4000$ and $N_x = 500$ on the domain $[0, 1]$. For these conditions the CFL number is:

$$\alpha = c\frac{\Delta t}{\Delta x} = 7 \times \frac{\frac{1}{400}}{\frac{1}{500}} = 8.75 > 1,$$

which violates the CFL condition. Multiplying Δt by $m = 10$ we get a CFL number of 0.875 which satisfies the stability condition.

The orthonormal basis functions we use are $\{sin(\pi kx)\}_{k=1}^{20}$. We use these basis functions to create the initial conditions that form the training, validation and testing data-sets. Each initial condition is a linear combination of the basis functions $\sum_{n=1}^{20} a_n sin(\pi nx)$. The coefficients are randomly generated and satisfy $\sum_{n=1}^{20} |a_n|^2 = 1$. Fig. 5 presents 5 examples of randomly generated initial conditions.

We generate 1,250 initial conditions and use the FD method with the fine discretization $(m\Delta t)$ as described in section 3.4 to create the data-set consisting of $1,250 \cdot 397 = 496,250$ samples. Each sample is of the form:

$$Input:\ (u^{n-10}, u^n) \quad \rightarrow \quad Label:\ (u^{n+10}, u^{n+11}), \quad n = 10, 20, ..., 3980$$

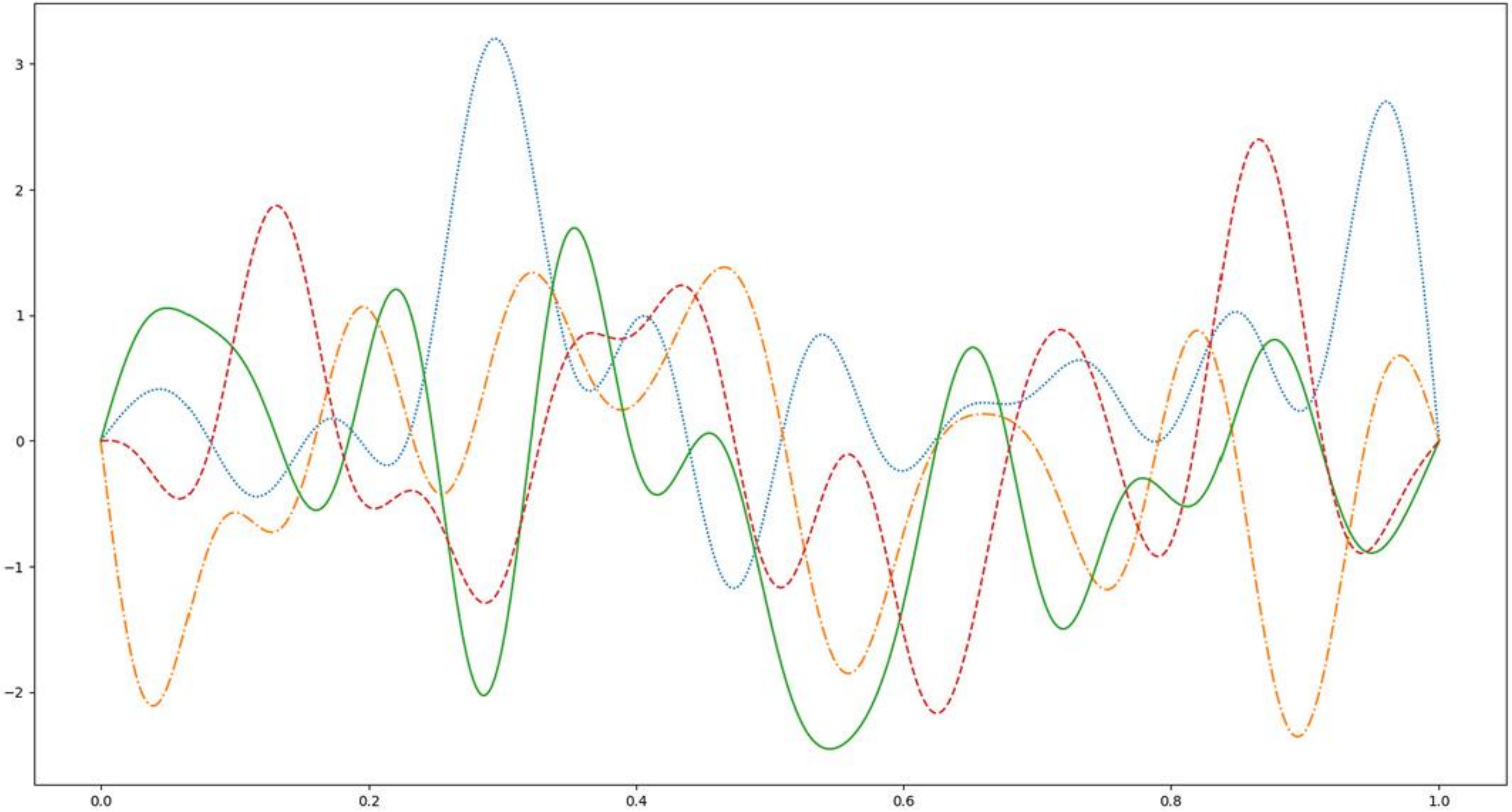


Fig. 5. 5 Randomly sampled initial conditions that were created for the data-set.

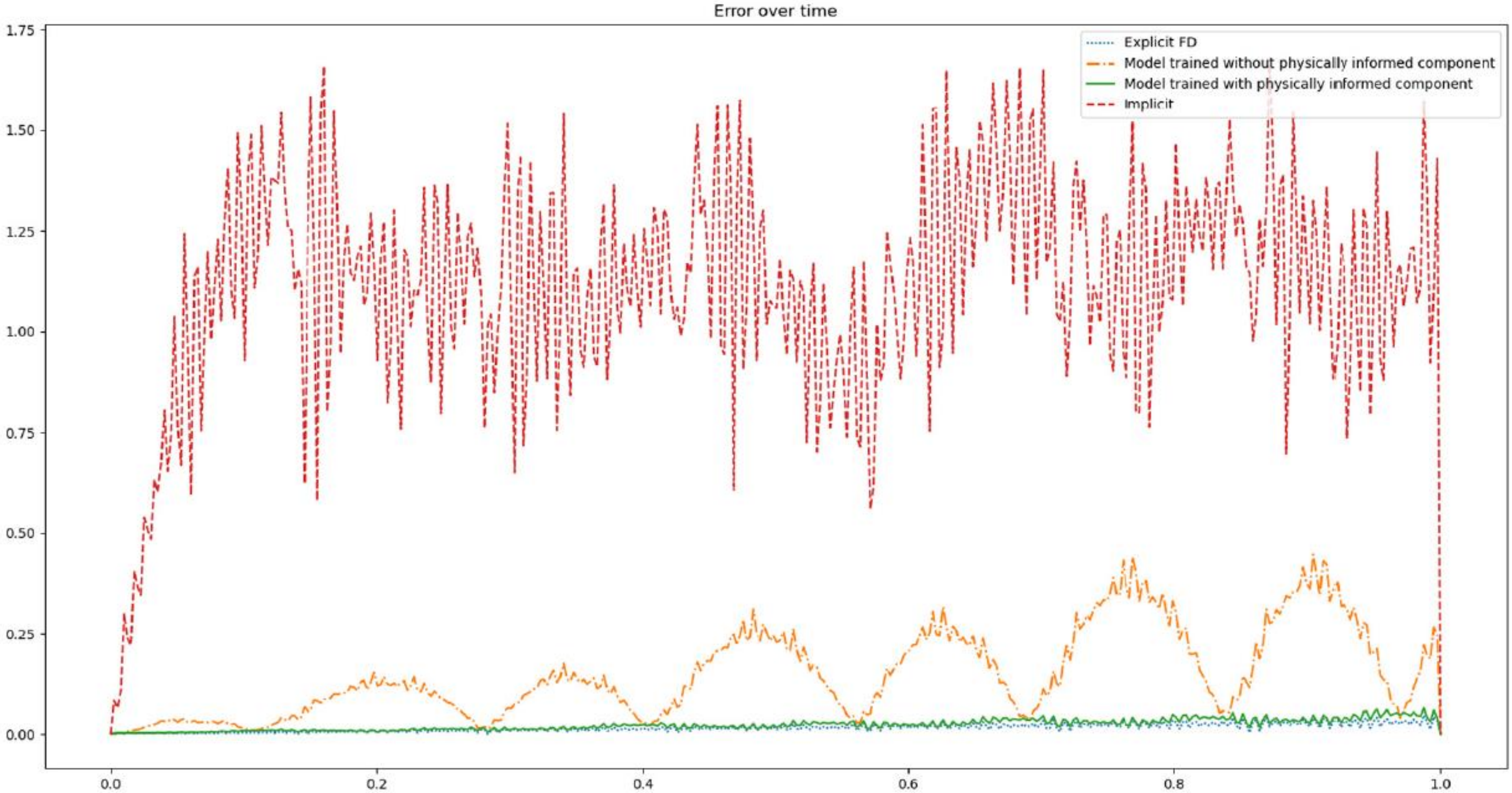


Fig. 6. Error over time comparing four different approximation methods.

5.3. Generalization

Using sines as basis functions for training and testing the model raises the question whether the model can perform well given an initial condition that was not formed using a finite linear combination of sine functions. We explore the ability of the model to generalize - predict solutions of initial conditions it was not trained on with high accuracy. We define a set of testing random initial conditions using:

$$u(x,0)=\sum_{k=1}^{20} a_k e^{-(x-b_k)^2+c_k} sin(\pi kx), \tag{6}$$

where $\sum_{k=1}^{20}|a_k|^2=\sum_{k=1}^{20}|b_k|^2=\sum_{k=1}^{20}|c_k|^2=1,\ a_k, b_k, c_k \in [-1,1]$ are generated randomly. Using separation of variables, we find the analytic solution of the wave problem given such an initial condition by finding its Fourier coefficients. The coefficients are calculated using:

$$A_l=\int_0^1\left(\sum_{j=1}^{20} a_j e^{-(x-b_j)^2+c_j} sin(\pi jx) sin(\pi lx)\right)dx, \quad \forall l:\ A_l \neq 0 \ almost\ surely,$$

and the solution is given by:

$$u(x,t)=\sum_{l=1}^{\infty} A_l sin(\pi lx) cos(c\pi lt).$$

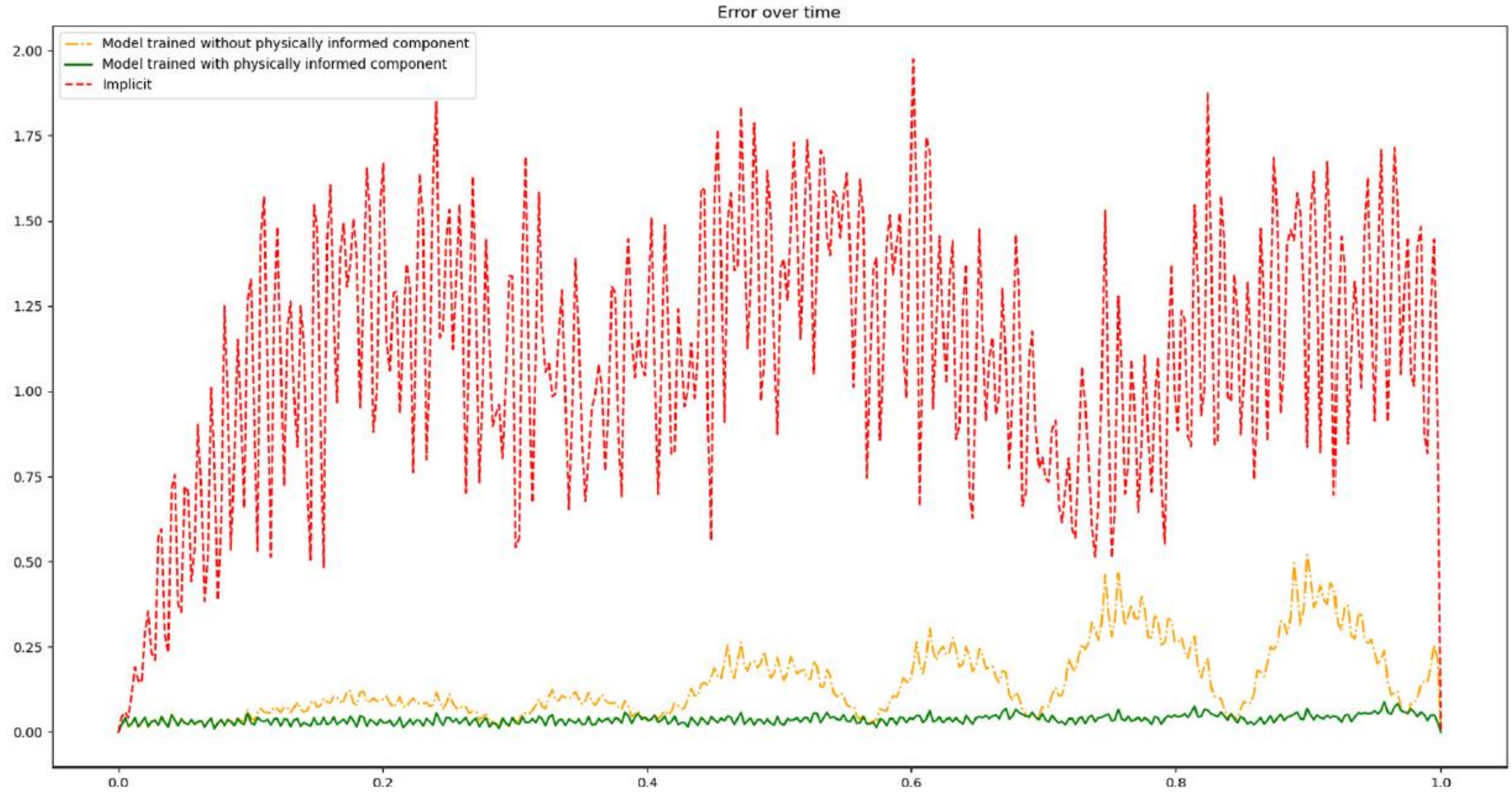


Fig. 7. Error over time comparing the implicit method and the model with physically-informed term, with an initial condition of the type (6).