Modeling Tsunami Waves at the Coastline of Valparaiso Area of Chile with Physics Informed Neural Networks

Alicja Niewiadomska[1],
Paweł Maczuga[1][0000−0002−5111−6981],
Albert Oliver-Serra[2][0000−0002−3783−8670],
Leszek Siwik[1][0000−0003−0535−7220],
Paulina Sepulveda-Salaz[4][0000−0002−7146−2240],
Anna Paszyńska[1,3][0000−0002−0716−0619],
Maciej Paszyński[1][0000−0001−7766−6052],
Keshav Pingali[5][0000−0002−0484−4636]

[1] AGH University of Krakow, Poland
paszynsk@agh.edu.pl
[2] University of Las Palmas de Gran Canaria (ULPGC), Spain.
albert.oliver@ulpgc.es
[3] Jagiellonian University, Krakow, Poland
anna.paszynska@uj.edu.pl
[4] Pontifical Catholic University of Valparaiso, Chile
paulina.sepulveda@pucv.cl
[5] The Oden Institute for Computational Engineering, USA
pingali@cs.utexas.edu

Abstract. The Chilean coast is a very seismically active region. In the 21st century, the Chilean region experienced 19 earthquakes with a magnitude of 6.2 to 8.8, where 597 people were killed. The most dangerous earthquakes occur at the bottom of the ocean. The tsunamis they cause are very dangerous for residents of the surrounding coasts. In 2010, as many as 525 people died in a destructive tsunami caused by an underwater earthquake. Our research paper aims to develop a tsunami simulator based on the modern methodology of Physics Informed Neural Networks (PINN). We test our model using a tsunami caused by a hypothetical earthquake off the coast of the densely populated area of Valparaiso, Chile. We employ a longest-edge refinement algorithm expressed by graph transformation rules to generate a sequence of triangular computational meshes approximating the seabed and seashore of the Valparaiso area based on the Global Multi-Resolution Topography Data available. For the training of the PINN, we employ points from the vertices of the generated triangular mesh.

1 Introduction

Tsunami waves (or just tsunamis) are "high-energy" ocean waves caused by the sudden displacement of large masses of ocean water, originating most frequently

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

from (high-magnitude) (underwater) earthquakes (or, less frequently, from underwater volcanic eruptions, calving of glaciers, massive landslides or meteorite impacts). Despite the high energy and moving at speeds of up to several hundred kilometers per hour, due to their low height (up to several dozen centimeters) and long length (up to several hundred kilometers), tsunami waves may even be unnoticed in the open ocean. The situation changes, however, dramatically when they reach the coastal zone, where they accumulate (often rapidly) and, reaching heights of up to several dozen meters, break onto the land, flooding and destroying everything that comes within their range of influence.

Due to their unpredictable nature, devastating power, and huge area and range of impact, we, as humankind, can still not predict their occurrence with 100% effectiveness and in a long enough advance to get a chance to evacuate people from endangered areas. Also, it is not possible to keep all the endangered areas like a greenfield or uninhabited sites, so as important as improving the methods and tools for monitoring and warning against their occurrence, an important direction of research are methods and techniques for modeling the propagation and behavior of these waves (especially while reaching the coastal zones) to make it possible and reliable to simulate the potential impact of their occurrence and, for example, to adjust evacuation routes, procedures, and coastal areas development plans appropriately to minimize their potential devastating effects.

Due to the high seismic activity, tsunami waves are (most) frequently observed in the Pacific region, where in the so-called (Pacific) Ring of Fire, or the Circum-Pacific Seismic Belt, there are most of Earth's active volcanoes located (more than 450 in total) and approximately 90% of all earthquakes in the world occur.

One of the countries particularly exposed to the devastating effects of tsunami waves due to its location in the area of the Circum-Pacific Seismic Belt, the length of the coastline (approximately 6.5 thousand kilometers), and a large number of islands and islets located near the coastline and in the open ocean (around 3,000 in total) is south-American Chile. How real is that dangerous for the Chilean territory (and the coast in particular) let it be proven by the fact that on the list of the strongest earthquakes recorded in history, as many as five can be classified as located in the Chilean region (coastline or the interior), including so-called the Great Chilean Earthquake, on May 22, 1960, considered, with the magnitude of 9.5 Richter degrees, as the strongest earthquake recorded so far in the history of seismic measurements. So, that is the direct motivation for the research undertaken and discussed in this paper.

As for the particular region considered in our research, we focused on the Valparaiso area. It has been chosen for two reasons. First, it is located approximately halfway along Chile's coastline so that the results may be representative of the broader part of the Chilean coast. Second, Valparaiso (the city and the region) is one of Chili's most densely populated and urbanized regions. Also, it is one of the most popular tourist regions on the west coast of South Amer-

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

ica, so the topic may be even more important and interesting for the Chilean community, researchers, and authorities.

The goal of this paper is to develop the Physics Informed Neural Network (PINN) model of a tsunami based on non-linear wave equations and the topography of the seabed and seashore in the Valparaiso region of Chile. Prof. Karniadakis proposed the PINN citec1 in 2019 as a method of solving Partial Differential Equations (PDEs) by training the neural networks. PINN has been implemented in the DeepXDE library [9]. It is an extensive library supporting TensorFlow, PyTorch, JAX, and PaddlePaddle, with huge functionality, including ODEs, PDEs, complex geometries, and different initial and boundary conditions for solving forward and inverse problems. Another library for solving the wave equation, Allan-Cahn equations, Volterra integrodifferential equations, and variational minimization problems is IDRLnet [10], which uses pytorch, numpy, and Matplotlib.

In this paper, we have implemented the PINN solver for the non-linear wave equation using our PINN-2DT library [14] implemented in PyTorch and executed in GoogleColab. It enables simple implementation and execution of the wave-equation simulations. The novelty of our computational method lies in the fact that we select the points for training PINN residuals following the adaptive mesh refinement procedure. Namely, we employ a longest-edge refinement algorithm [17] to generate an accurate triangular computational mesh approximating the seabed and seashore of the Valparaiso area. Our adaptive method employs the Global Multi-Resolution Topography Database (GMRT) [11]. We express the longest-edge refinement algorithm by graph transformation rules. For the training of the PINN, we use the points from the vertices of the generated triangular mesh. Following the ideas presented in our previous model [18, 19, 8], we express the triangular mesh refinement algorithm by a set of graph transformation rules.

Alternative available state-of-the-art simulators are based on finite volume method (GeoClaw [20]), finite difference method (COMCOT [21], CoulWAVE [22]), a hybrid of finite difference-finite volume methods (Celeris base [23]), or finite element method [8]. We verify the correctness of our PINN simulator by comparing it to the finite element method solver [8] executed on the model "pool" example. The PINN model's potential advantage over the finite element method solver lies mainly in its simplicity of implementation, employing space-time formulations, and lack of problems with time stepping and stabilization.

2 Modeling of tsunami wave with Physics Informed Neural Networks

Our basic model for simulations of the tsunami waves is based on the formulation using the wave equation described in [8]. Let us focus on a strong form of the wave equation: Find $u \in C^2(0,1)$ for $(x,y) \in \Omega = [0,1]^2$, $t \in [0,T]$ such that:

$$\frac{\partial^2 u(x,y,t)}{\partial t^2} - \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right) \cdot \left(g(u(x,y,t) - z(x,y)) \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right)\right) = 0, \quad (x,y,t) \in \Omega \times [0,T], \tag{1}$$

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

Here $z(x, y)$ stands for the bathymetry (the seabed and the seashore topography function), $g = 9,81$ is the acceleration due to the Earth's gravity, and $u(x, y, t)$ represents the water level. We start the simulation by assuming the initial tsunami wave shape:

$$u(x, y, 0) = u_0(x, y), \ (x, y) \in \Omega. \tag{2}$$

We assume the no-reflection zero-Neumann boundary condition:

$$\frac{\partial u}{\partial n} = 0, \quad \text{on } \partial\Omega \times [0, T]. \tag{3}$$

We expand the wave equation PDE by computing the partial derivatives:

$$\begin{aligned} \frac{\partial^2 u(x,y,t)}{\partial t^2} - \left(g \left(\frac{\partial u(x,y,t)}{\partial x} - \frac{\partial z(x,y)}{\partial x} \right) \frac{\partial u(x,y,t)}{\partial x} \right) \\ - \left(g \left(u(x,y,t) - z(x,y) \right) \frac{\partial^2 u(x,y,t)}{\partial x^2} \right) \\ - \left(g \left(\frac{\partial u(x,y,t)}{\partial y} - \frac{\partial z(x,y)}{\partial y} \right) \frac{\partial u(x,y,t)}{\partial y} \right) \\ - \left(g \left(u(x,y,t) - z(x,y) \right) \frac{\partial^2 u(x,y,t)}{\partial y^2} \right) = 0 \end{aligned} \tag{4}$$

We employ the Physics Informed Neural Network approach, proposed by Karniadakis [1]. The neural network is understood there as a continuous function that represents the solution:

$$u(x, y, t) = PINN(x, y, t) = A_n \sigma \left(A_{n-1} \sigma (...\sigma (A_1 \begin{bmatrix} x \\ y \\ t \end{bmatrix} + B_1)... + B_{n-1} \right) + B_n \tag{5}$$

Here, A_i represents the matrices of layers of the neural network, and B_i represents the bias vectors. Also, σ is the activation function.

We choose the sigmoid activation function after our analysis in [2], where we show it best fits for wave equation training with PINN. In our training, we employed three internal layers, each one with 300 neurons. The motivation was that a smaller number of neural networks (4 layers with 80 neurons) experience some problems with the training of the wave equations.

We define the loss function as the residual of the PDE.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

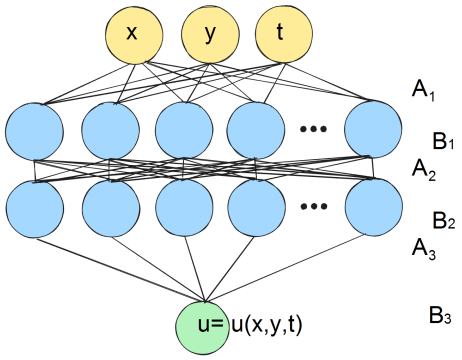

Fig. 1: Physics informed neural network as a continuous function representing a solution function. The matrix entries $A_1^{kl}, A_2^{kl}, A_3^{kl}$ represents the weight of edges connecting the neurons. The vector entries B_1^k, B_2^k, B_3^k represent the biases assigned to neurons.

$$
\begin{aligned}
&LOSS_{PDE}(x,y,t) = \\
&\left(\frac{\partial^2 PINN(x,y,t)}{\partial t^2} - \left(g\left(\frac{\partial PINN(x,y,t)}{\partial x} - \frac{\partial z(x,y)}{\partial x}\right)\frac{\partial PINN(x,y,t)}{\partial x}\right)\right. \\
&\qquad - \left(g\left(PINN(x,y,t) - z(x,y)\right)\frac{\partial^2 PINN(x,y,t)}{\partial x^2}\right) \\
&\qquad - \left(g\left(\frac{\partial PINN(x,y,t)}{\partial y} - \frac{\partial z(x,y)}{\partial y}\right)\frac{\partial PINN(x,y,t)}{\partial y}\right) \\
&\qquad \left. - \left(g\left(PINN(x,y,t) - z(x,y)\right)\frac{\partial^2 PINN(x,y,t)}{\partial y^2}\right)\right)^2 \quad (6)
\end{aligned}
$$

On top of the loss function related to the residual of the PDE, we also need to define the loss function for training the initial condition $u_0(x,y)$:

$$LOSS_{Init}(x,y,0) = \left(PINN(x,y,0) - u_0(x,y)\right)^2 \quad (7)$$

We also define the loss of the residual of the Neumann boundary condition:

$$LOSS_{BC}(x,y,t) = \left(\frac{\partial PINN(x,y,t)}{\partial n} - 0\right)^2. \quad (8)$$

During the training, we adjust the weights of the matrices representing the neural network layers. We also adjust entries of the vector representing the neural network biases. This is done by using the derivatives of the neural network with respect to particular entries. The sketch of the gradient descent procedure is the following:

The sketch of the training procedure is the following

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

– Repeat
  - Select points $(x, y, t) \in \Omega \times [0, T]$ from vertices of the computational mesh (excluding its boundary edges) approximating the bathymetry, update the weights: $A^k_{i,j} = A^k_{i,j} - \eta \frac{\partial LOSS_{PDE}(x,y,t)}{\partial A^k_{i,j}}$, $B^k_i = B^k_i - \eta \frac{\partial LOSS_{PDE}(x,y,t)}{\partial B^k_i}$.
  - Select point $(x, y) \in \partial\Omega$ from vertices of the computational mesh located on its boundary, approximating the seabed and the seashore, and update the weights: $A^k_{i,j} = A^k_{i,j} - \eta \frac{\partial LOSS_{BC}(x,y,t)}{\partial A^k_{i,j}}$, $B^k_i = B^k_i - \eta \frac{\partial LOSS_{BC}(x,y,t)}{\partial B^k_i}$.
  - Select point $(x, y, 0) \in \Omega \times \{0\}$ from vertices of the computational mesh located on its boundary at time moment 0, update the weights $A^k_{i,j} = A^k_{i,j} - \eta \frac{\partial LOSS_{Init}(x,y,0)}{\partial A^k_{i,j}}$, $B^k_i = B^k_i - \eta \frac{\partial LOSS_{Init}(x,y,0)}{\partial B^k_i}$.
– Until $w_{PDE} LOSS_{PDE} + w_{BC} LOSS_{BC} + w_{Init} LOSS_{Init} \leq \delta$

As for the parameters, we run our experiments with the following setup:

– Maximum number of epochs: 15000,
– $\eta = 0.00015$ (lines 4,5 and 6).

The selection of the training rate came from experimenting with smaller and larger values. The weights w_{PDE}, w_{BC}, w_{Init} are selected by the adaptive loss weighting of Neural Networks with multi-part loss functions method.

During the training procedure, we employ the ADAM algorithm [3], storing the derivatives computed in previous iterations and taking the weighted average to modify the neural network's weights. In this way, it avoids being trapped in local minima. There are also several modern modifications to the ADAM algorithm available [6, 4, 5], but ADAM seems to be the most popular one.

3 Verification of the code

To verify the correctness of the PINN, we compare with finite element method [8] executed on the "pool" scenario, using the non-linear wave equation

$$\frac{\partial^2 u}{\partial t^2} - \nabla \left(g(u - z) \nabla u \right) = 0, \tag{9}$$

We employ a Bubnov-Galerkin [24] finite element method for spatial discretization and an explicit finite difference time-stepping scheme for temporal discretization. Namely, we introduce a finite difference approximation of the second time derivative $\frac{\partial^2 u}{\partial t^2} \approx \frac{u_t - 2u_{t-1} + u_{t-2}}{dt^2}$:

$$\underbrace{u_t}_{\text{Next state}} = \text{`} \quad \underbrace{u_{t-1}}_{\text{Previous state}} + \underbrace{u_{t-1} - u_{t-2}}_{\text{States difference}} + \underbrace{\Delta t^2}_{(\text{Time step})^2} \underbrace{\nabla \left(g(u_{t-1} - z) \nabla u_{t-1} \right)}_{\text{Physics}} \tag{10}$$

We multiply by test functions v, integrate by parts, and apply the boundary conditions $\nabla u_t \cdot n = 0$. We solve: Find $u \in V$:

$$\begin{aligned} (u_t, v) = &(u_{t-1}, v) + C \left(u_{t-1} - u_{t-2}, v \right) \\ &- \Delta t^2 \left(g(u_{t-1} - z) \nabla u_{t-1}, \nabla v \right) \quad \forall v \in V \end{aligned} \tag{11}$$

Following [8], we have introduced a damping constant C in front of the wave propagation term to stabilize the finite element formulation.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

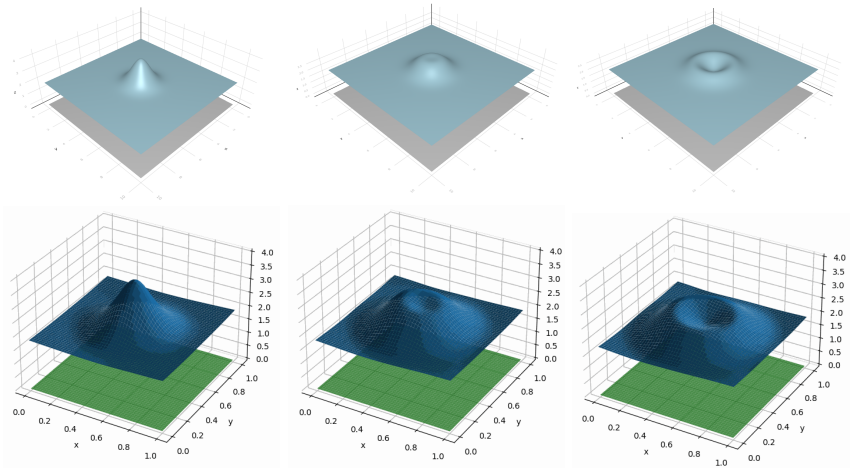

Fig. 2: Model simulation of the wave equation in a "pool" using the finite element method code (first row) and PINN solver (second row).

4 Generation of the computational mesh describing the seashore and seabed near Valparaiso region

For the training of the Physics Informed Neural Network model, we select the points as vertices of mesh triangles of the most refined mesh from the adaptive procedure towards the GMRT [11] database data. As the refinement criterion we estimate the error between the triangular elements approximation of the topography and the GMRT topography data. For the adaptation, we employ the longest-edge refinement algorithm [8]. We have executed 22 iterations of the longest-edge refinement algorithm starting from a rectangle partitioned into two triangles. The adaptive mesh refinements were continued until the required topography accuracy was reached. The overview of the refinement process is presented in Figure 3. The resulting mesh is shown in Figure 4. The mesh is employed as a barymetry function $z(x, y)$. On top of that mesh, we introduce the sea level $u(x, y)$. Table 1 presents the number of triangles and vertices of the generated meshes. The final mesh has 91,458 triangles and 45,957 vertices. We employ these 45,957 vertices to train the PINN model.

The longest-edge refinement algorithm employed for the generation of the computational mesh approximating the seashore and seabed of the Valparaiso region of Chile has been expressed by graph transformations. Some representative transformations are presented in Figures 5-6. The rules presented in this paper are the transformation of the hyper-graph grammar implementation of the longest-edge refinement algorithm described in [16] into the composite graph grammar, employed previously for *hp* adaptive mesh refinements [15]. They express the rules for the creation of the new triangular elements in a way that

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

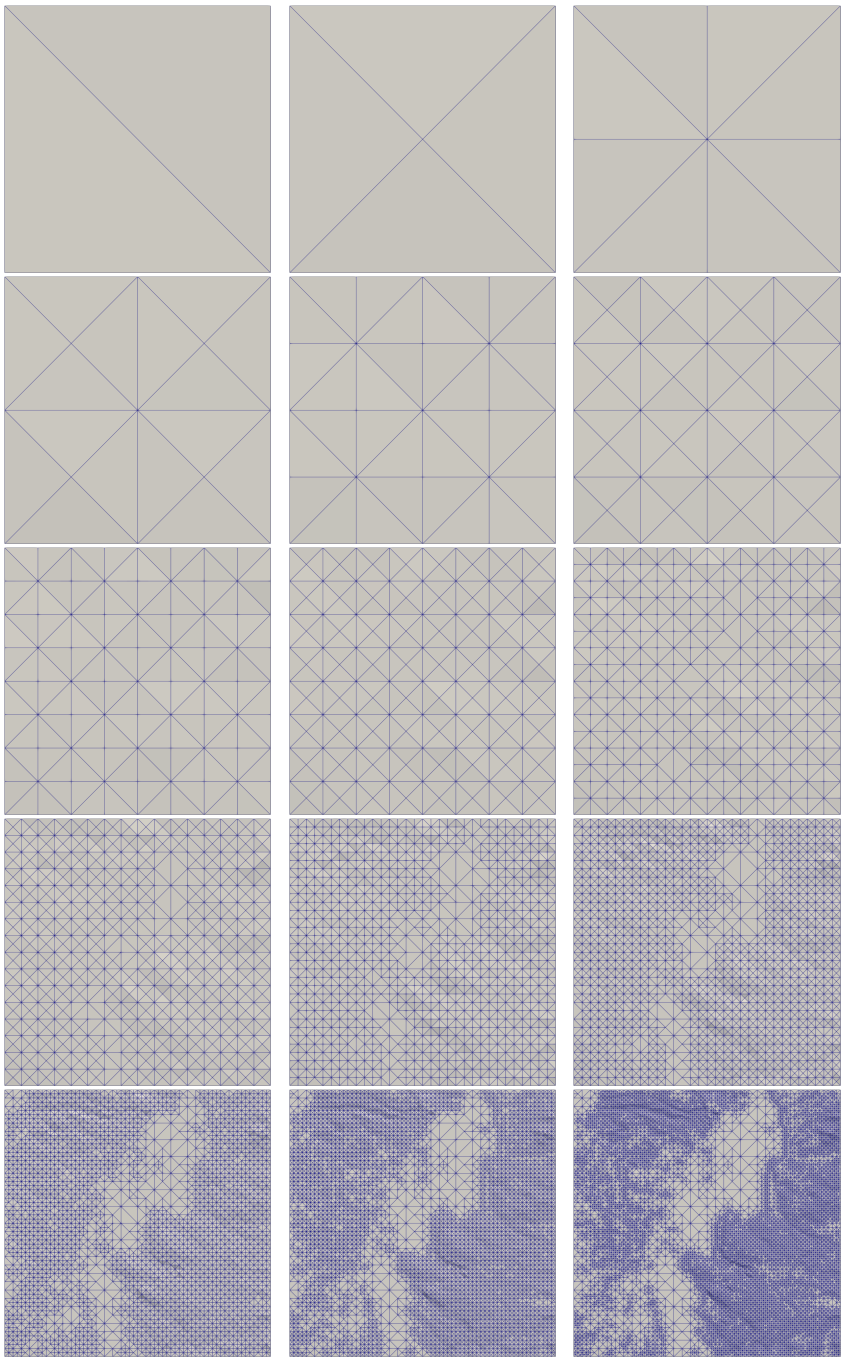

Fig. 3: The sequence of computational meshes approximating the seabed and seashore at the Valparaiso area of Chile, obtained by executing the longest edge refinement algorithm starting from the rectangular domain partitioned into two triangles.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

Iteration	Number of vertices	Number of triangles
0	4	2
1	5	4
2	9	8
3	13	16
4	25	32
5	41	64
6	81	128
7	145	256
8	284	502
9	519	972
10	1008	1890
11	1834	3540
12	3443	6658
13	6050	11869
14	10126	19904
15	15492	30619
16	21144	41866
17	25205	49974
18	28103	55763
19	30909	61372
20	34409	68369
21	38808	77165
22	45957	91458

Table 1: The number of vertices and triangles generated by an adaptive longest-edge refinement algorithm executed from two triangles.

no hanging nodes are generated in the mesh. They employ the predicates of applicability, telling when the transformation can be executed.

5 Numerical results

After initial experiments with the ADAM optimization algorithm [3], due to the fact that we have residual, boundary and initial loss functions, thus, the multi-objective optimization problem, we switched to the SoftAdapt algorithm [13] executing ADAM algorithm with adaptive weighting of three loss functions. The convergence of these three loss functions, as well as the convergence of the total loss function, is presented in Figure 7.

The residual loss first learns the trivial zero solution, but when the model learns the initial and boundary conditions, it adjusts and maintains a high accuracy of the order of 0.001. The initial loss learns the initial state with an accuracy of almost 0.003. The boundary loss is also trained with an accuracy of 0.001. As a result, the PINN has learned the solution with a total loss function accuracy of 0.003. The initial condition is given by:

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

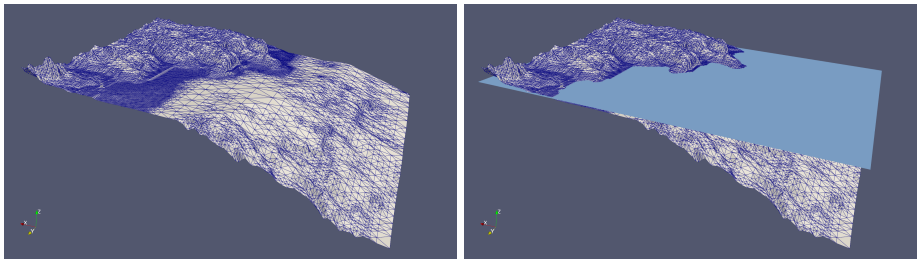

Fig. 4: The final computational mesh representing the seashore and seabed near the Valparaiso area obtained after 22 iterations of the longest-edge refinement algorithm. The topographic mesh represents the bathymetry function $z(x, y)$; the sea level represents the undisturbed water level $u(x, y, t)$.

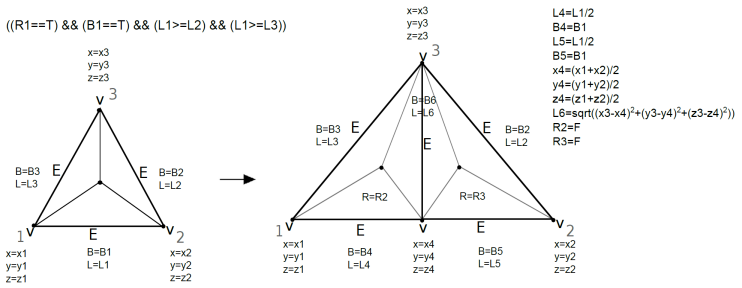

Fig. 5: Production (P1).

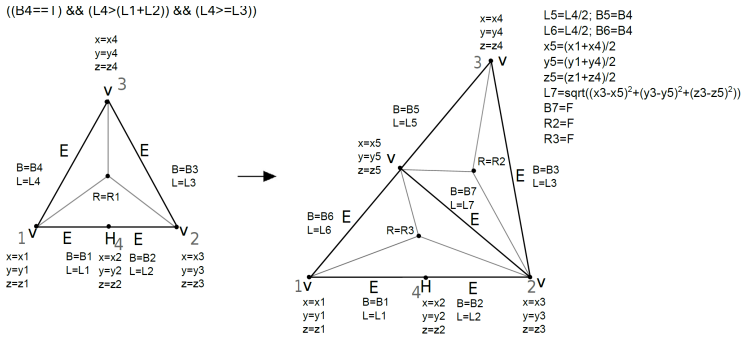

Fig. 6: Production (P2).

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

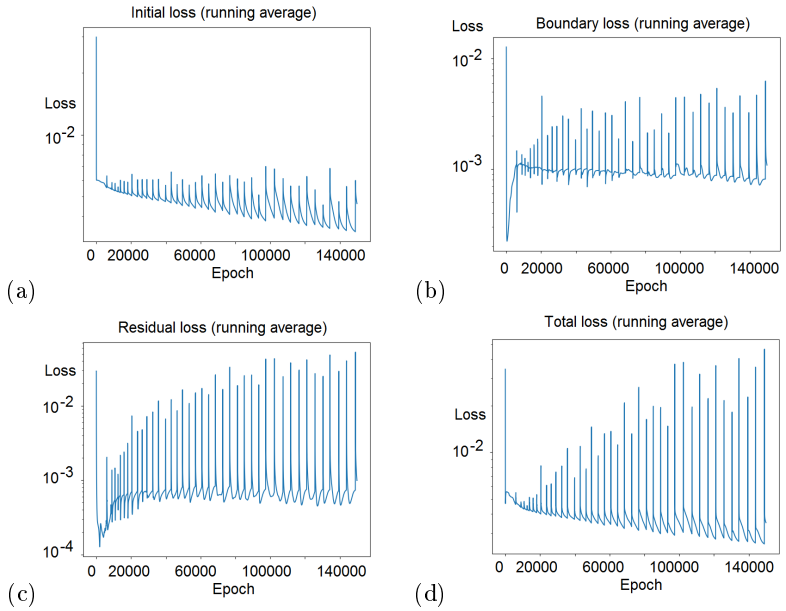

Fig. 7: The convergence of the training for the initial (a), boundary (b), residual (c), and total (d) loss.

$$u_0(x, y, 0) = 0.04 \exp\left(-120\sqrt{(x - x_0)^2 + (y - y_0)^2}\right) + 0.5 \tag{12}$$

in the relative units of $[0, 1]^3$ of the computational domain. This setup represents the following interpretation. The ocean level is set at 0.5 (the middle of the horizontal height of the domain). The seashore spans between 0.5 and 0.55, so the initial tsunami wave height is set to 0.55 (see first panel in Figure 5). The center of the initial tsunami wave is located in $1/5$ of the domain in the x direction and $1/2$ of the domain in the y direction (see the first panel in Figure 5). The snapshots of the simulations are presented in Figure 9. We can predict the propagation of the tsunami waves along the coastline as well as check the water elevation in the coastal area.

6 Conclusions

In this paper, we showed how to use physics-informed neural networks to simulate tsunami wave propagation for an assumed seafloor and coastal topography. Our application consisted of the following building blocks. The PINN model used a nonlinear wave propagation equation [8]. Neural network learning was based on the residuum of the equation, boundary condition, and initial condition using the ADAM algorithm [3], adaptive loss function weight selection [13]. The topography of the seabed and waterfront was approximated using an

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

adaptive generation algorithm for computational grids built from triangular elements. Adaptation was performed using the longest-edge refinement method on the basis of the Global Multi-Resolution Topography Database [11]. The points selected for learning the neural network, in particular for sampling the equation residuum, boundary, and initial condition, were chosen based on the nodes of the generated computational grid. Having the computational meshes generated for the coastline of Chile, it takes a couple of days for a master's degree student to implement and run the PINN simulator from scratch, employing the mesh points for training. Implementing the finite element method solver [8] requires weeks of a very careful development and debugging, including issues with the stabilization of the simulation, proper selection of time iteration schemes, and dealing with simulator behavior at the coastline area. The PINN solver provides similar accuracy results as a linear finite element [8] solver. The future work may involve comparison to higher-order finite element method solvers [25, 26].

Acknowledgements The work of Albert Oliver Serra was supported by "Ayudas para la recualificación del sistema universitario español" grant funded by the ULPGC, the Ministry of Universities by Order UNI/501/2021 of 26 May, and the European Union-Next Generation EU Funds The authors are grateful for support from the funds the Polish Ministry of Science and Higher Education assigned to AGH University of Krakow. The visit of Maciej Paszyński at ODen Institute was supported by J. T. Oden Research Faculty Fellowship.

References

1. M. Raissi, P.Perdikaris, 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 (2019) 686-707
2. Paweł Maczuga, Maciej Paszyński, Influence of Activation Functions on the Convergence of Physics-Informed Neural Networks for 1D Wave Equation, Computational Science – ICCS 2023: 23rd International Conference, Prague, Czech Republic, July 3-5, 2023, Proceedings, Part I, (2023) 74–88,
3. D. P. Kingma, J. Lei Ba, ADAM: A method for stochastic optimization, arXiv:1412.6980 (2014)
4. Ilya Loshchilov, Frank Hutter, Decoupled Weight Decay Regularization arxiv.org/abs/1711.05101 (2019)
5. Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, Quoc V. Le, Symbolic Discovery of Optimization Algorithms, arxiv.org/abs/2302.06675 (2023)
6. Yichuan Deng, Hang Hu, Zhao Song, Omri Weinstein, Danyang Zhuo, Training Overparametrized Neural Networks in Sublinear Time, arxiv.org/abs/2208.04508 (2022)
7. Yang Chen, Yongfu Xu, Lei Wang, Tianyi Li, Modeling water flow in unsaturated soils through physics-informed neural network with principled loss function, Computers and Geotechnics, 161 (2023) 105546,
8. Paweł Maczuga, Albert Oliver-Serra, Anna Paszyńska, Eirik Valseth, Maciej Paszyński, Graph-grammar based algorithm for asteroid tsunami simulations, Journal of Computational Science, 64 (2022) 101856,

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

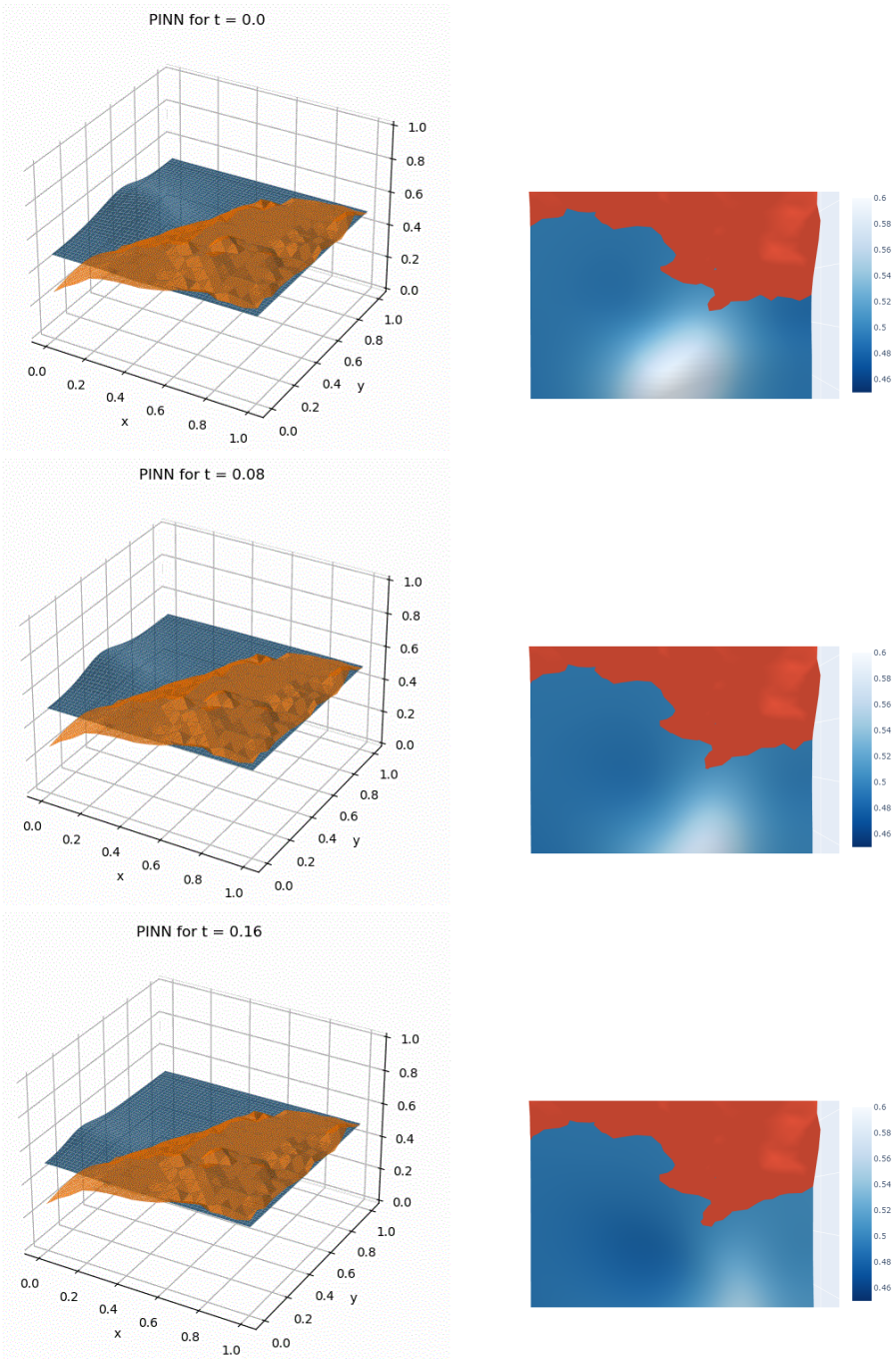

Fig. 8: Snapshots from tsunami simulation near the Valparaiso region of Chile (left panels). Changes of the coastline caused by the tsunami near the Valparaiso region of Chile (right panels).

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

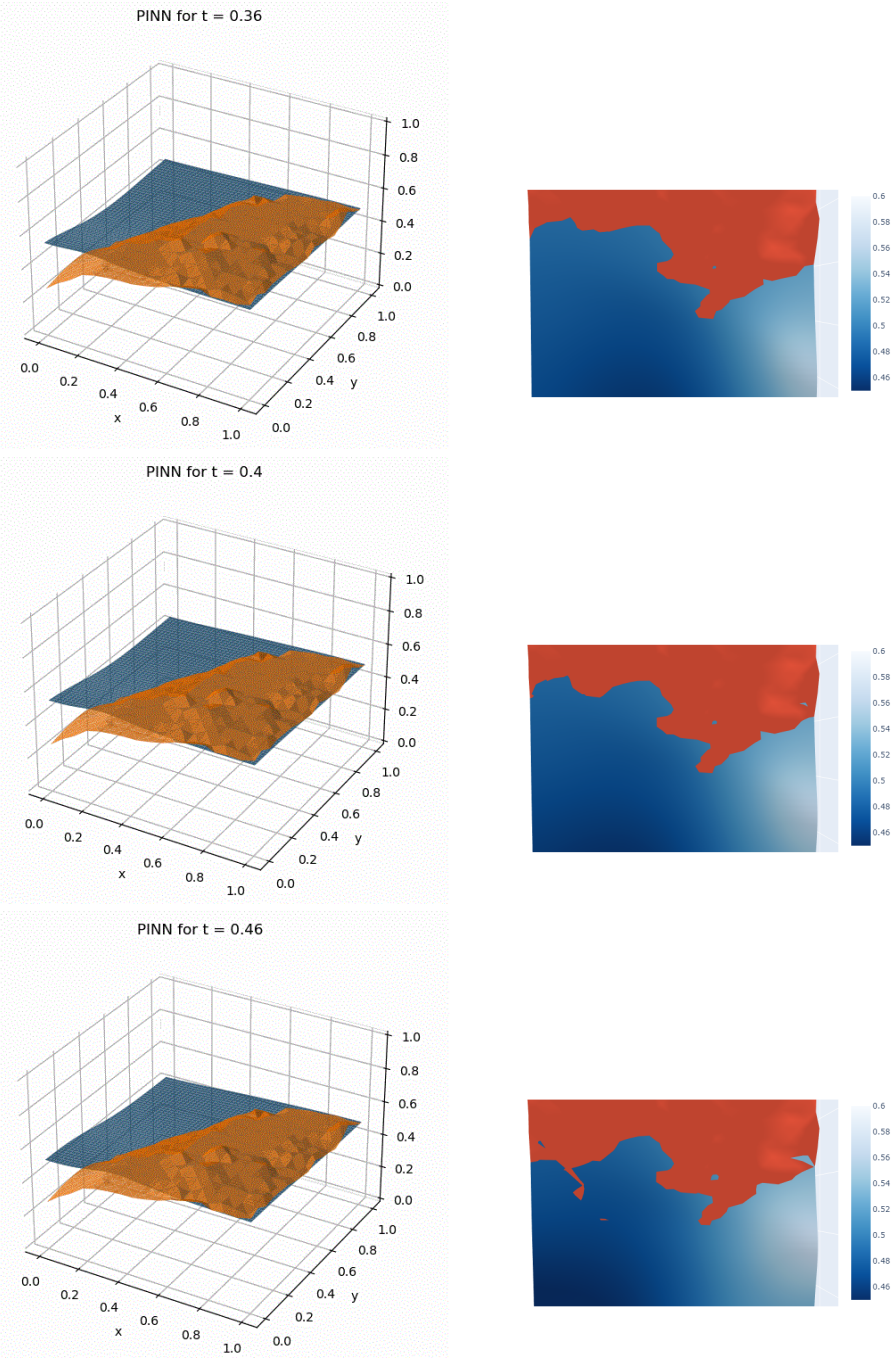

Fig. 9: Snapshots from tsunami simulation near the Valparaiso region of Chile (left panels). Changes of the coastline caused by the tsunami near the Valparaiso region of Chile (right panels).

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14

9. Lu Lu, Xuhui Meng, Zhiping Mao, George Em Karniadakis, DeepXDE: A deep learning library for solving differential equations, SIAM Review, 63(1), (2021) 208-228.
10. Wei Peng and Jun Zhang and Weien Zhou and Xiaoyu Zhao and Wen Yao and Xiaoqian Chen, IDRLnet: A Physics-Informed Neural Network Library, arxiv.2107.04320 (2021)
11. Global Multi-Resolution Topography Data Synthesis *https://www.gmrt.org/*
12. https://en.wikipedia.org/wiki/List_of_earthquakes_in_Chile
13. A. Ali Heydari, Craig A. Thompson, Asif Mehmood, SoftAdapt: Techniques for Adaptive Loss Weighting of Neural Networks with Multi-Part Loss Functions, arXiv:1912.12355v1 (2019)
14. P. Maczuga, M. Sikora, M. Skoczeń, P. Rożnawski, F. Tłuszcz, M. Szubert, M. Łoś, W. Dzwinel, K. Pingali, M. Paszyński, Physics Informed Neural Network Code for 2D Transient Problems (PINN-2DT) Compatible with Google Colab arXiv:2310.03755v2 (2024)
15. A. Paszyńska, M. Paszyński, E. Grabska, Graph Transformations for Modeling hp-Adaptive Finite Element Method with Triangular Elements. In: Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds) Computational Science – ICCS 2008. ICCS 2008. Lecture Notes in Computer Science, vol 5103. Springer, Berlin, Heidelberg (2008)
16. K. Podsiadło, A. O. Serra, A. Paszyńska, R. Montenegro, I. Henriksen, M. Paszyński, K. Pingali, Parallel graph-grammar-based algorithm for the longest-edge refinement of triangular meshes and the pollution simulations in Lesser Poland area, Engineering with Computers 37, 3857–3880 (2021)
17. M. C. Rivara, New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations, International Journal of Numerical Methods in Engineering 40 (1997) 3313–3324.
18. A. Paszyńska, M. Paszyński, E. Grabska, Graph Transformations for Modeling hp-Adaptive Finite Element Method with Triangular Elements. Computational Science–ICCS 2008: 8th International Conference, Krakow, Poland (2008) 604-613.
19. M. Paszyński, A. Paszyńska, Graph Transformations for Modeling Parallel hp-Adaptive Finite Element Method, Parallel Processing and Applied Mathematics: 7th International Conference, Gdansk, Poland (2007) 1313-1322.
20. R. J. LeVeque, D. L. George, M. J. Berger, Tsunami modelling with adaptively refined finite volume methods. Acta Numerica. 20 (2011) 211-289.
21. X. Wang, User manual for COMCOT version 1.7. Cornel University (2009)
22. P. Lynett, P.L.F. Liu, K.I. Sitanggang, D. Kim, Modeling Wave Generation, Evolution, and Interaction with Depth-Integrated, Dispersive Wave Equations COULWAVE Code Manual Cornell University Long and Intermediate, Wave Modeling Package V. 2.0, Cornell University, Itacha, New York (2008)
23. S. Tavakkol, P. Lynett, Celeris base: An interactive and immersive Boussinesq-type nearshore wave simulation software Compututer & Physics Communication. 248 (2020) Article 106966
24. E. B. Becker, G. F. Carey, J. T. Oden, *Finite elements: an introduction*, Vol. 1, Prentice Hall (1981)
25. M. Woźniak, M. Łoś, M. Paszyński, L. Dalcin, V. M. Calo, Parallel fast isogeometric solvers for explicit dynamics. Computing and Informatics 36 (2) (2017s) 423-448.
26. M. Łoś, J. Munoz-Matute, I. Muga, M. Paszyński, Isogeometric Residual Minimization Method (iGRM) with direction splitting for non-stationary advection–diffusion problems, Computers & Mathematics with Applications 79 (2) (2020) 213-229.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63751-3_14