



Physics-informed neural networks with hard and soft boundary conditions for linear free surface waves

Downloaded from: <https://research.chalmers.se>, 2025-09-25 13:42 UTC

Citation for the original published paper (version of record):

Sheikholeslami, M., Salehi, S., Mao, W. et al (2025). Physics-informed neural networks with hard and soft boundary conditions for linear free surface waves. *Physics of Fluids*, 37(8). <http://dx.doi.org/10.1063/5.0277421>

N.B. When citing this work, cite the original published paper.

RESEARCH ARTICLE | AUGUST 13 2025

Physics-informed neural networks with hard and soft boundary conditions for linear free surface waves

Mohammad Sheikholeslami  ; Saeed Salehi ; Wengang Mao ; Arash Eslamdoost ; Håkan Nilsson 



Physics of Fluids 37, 087158 (2025)

<https://doi.org/10.1063/5.0277421>



View
Online



Export
Citation

Articles You May Be Interested In

Interfacial conditioning in physics informed neural networks

Physics of Fluids (July 2024)

Physics-informed neural network (PINNs) for convection equations in polymer flooding reservoirs

Physics of Fluids (March 2025)

Physics-informed neural networks for fully non-linear free surface wave propagation

Physics of Fluids (June 2024)



Physics of Fluids

Special Topics Open
for Submissions

[Learn More](#)

Physics-informed neural networks with hard and soft boundary conditions for linear free surface waves

Cite as: Phys. Fluids **37**, 087158 (2025); doi: [10.1063/5.0277421](https://doi.org/10.1063/5.0277421)

Submitted: 23 April 2025 · Accepted: 14 July 2025 ·

Published Online: 13 August 2025








View Online



Export Citation



CrossMark

Mohammad Sheikholeslami,^{1,a)}  Saeed Salehi,²  Wengang Mao,¹  Arash Eslamdoost,¹ 
and Håkan Nilsson² 

AFFILIATIONS

¹Division of Marine Technology, Department of Mechanics and Maritime Sciences, Chalmers University of Technology, Gothenburg SE-412 96, Sweden

²Division of Fluid Dynamics, Department of Mechanics and Maritime Sciences, Chalmers University of Technology, Gothenburg SE-412 96, Sweden

^{a)} Author to whom correspondence should be addressed: mohshe@chalmers.se

ABSTRACT

Physics-informed neural networks (PINNs) are introduced to solve the linear wave problem described by potential flow theory. In the proposed PINN framework, both soft and hard enforcing of boundary conditions (BCs) at the bottom and sides of the wave domain are proposed. Two scenarios for solving the linear wave problem are investigated to find suitable PINN architectures. In the first scenario, the free surface wave is considered to be completely defined, and in the second scenario, the wave angular frequency is considered an unknown parameter. With a completely defined free surface wave and incorporating both the free surface and bottom BCs as soft constraints, the average velocity distribution error is less than 3%. Incorporation of a periodic BC (PBC) as a soft constraint reduces the average error to 0.10%. A hard constraint PBC gives an average error of 0.16%, with a strict representation of the PBC. This study also explores the design of trial functions to impose the kinematic bottom BC (KBBC) as a hard constraint. While these trial functions strictly satisfy the KBBC, they limit the solution space, increasing the average error up to almost 15 times. When the angular frequency of the wave is considered as an unknown parameter, to be estimated by the PINN, its value is estimated with an average error of 0.03%. Since linear wave theory underpins many wave simulation approaches, the results of this study lay the groundwork for extending the PINN framework to more complex wave models.

© 2025 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 4.0 International (CC BY-NC-ND) license (<https://creativecommons.org/licenses/by-nc-nd/4.0/>). <https://doi.org/10.1063/5.0277421>

I. INTRODUCTION

Accurate and cost-effective modeling of free surface waves can help to improve the design of ships and offshore structures. Fluid flows are governed by the Navier-Stokes equations, which offer the most comprehensive description of fluid dynamics. A state-of-the-art method to study free surface waves is to employ CFD simulations with an interface tracking method.^{1,2} However, this comes with a high computational cost. Alternatively, solutions based on potential flow theory are often regarded as sufficiently accurate to model different types of free surface waves,³ with fewer challenges.

Although these methods excel in many applications, they are not inherently designed to incorporate experimental data, which could address potential gaps in the chosen physical model. Additionally, they encounter significant challenges when addressing inverse problems,

such as determining wave characteristics from partially known fluid flow features, such as pressure and velocity fields, which remain computationally demanding.

The above-mentioned challenges highlight the need for more versatile approaches, such as physics-informed neural networks (PINNs), which can combine data and physical laws. PINNs, introduced by Raissi *et al.*,⁴ are a class of neural networks designed to solve physical problems governed by partial differential equations (PDEs). Unlike traditional machine learning models that rely heavily on large labeled datasets, PINNs incorporate the physical laws of a system directly into the training process by encoding the governing equations and boundary conditions into the loss function. This allows PINNs to make accurate predictions even with limited or no labeled data, making them especially useful in scientific and engineering contexts where data may

be sparse or expensive to obtain. Due to its widespread applicability, regularization of the training data, and flexibility, PINNs can be used to solve different scientific and engineering problems.^{5–8} Different types of problems solved via PINNs can be found in review articles.^{9–11} A challenge with PINNs is that the solution does not accurately satisfy the boundary conditions if they are applied as (soft) loss functions. A remedy to this is to implement them as (hard) constraints within the architecture of the network.^{12,13}

Chen *et al.*¹⁴ developed a PINN to determine the flow properties of the inviscid rotational flow beneath nonlinear periodic waves. Huang *et al.*² introduced a PINN methodology that incorporates initial conditions as well as boundary conditions (BCs) as hard constraints, and validated their model on free surface problems. Jagtap *et al.*¹⁵ applied a PINN to study nonlinear wave dynamics, governed by the Serre–Green–Naghdi equations in an ill-posed setting, by combining labeled data with physical laws. Duong *et al.*¹⁶ used PINNs to investigate wave-in-deck loading phenomena, where the measured data, together with Euler’s equation, were employed to reconstruct the free surface. Wang *et al.*¹⁷ demonstrated that PINNs are effective in reconstructing nearshore wave fields by utilizing wave energy balance equations, dispersion relations, and labeled data. Sheikholeslami *et al.*¹⁸ proposed a PINN framework to estimate the velocity potential beneath a sinusoidal wave based on the linear wave theory, achieving an error of less than 5%.

The majority of the research on the use of PINNs for free surface flows has focused on solving the Euler equations,^{2,14,16} Serre–Green–Naghdi equations,¹⁵ and shallow-water equations.^{19,20} The linear wave problem, known as Airy wave, has not yet been studied using PINNs in the literature. One example where its analytical solution has been used is in the development of a PINN framework for reconstructing irregular long-crested waves in offshore engineering applications.²¹ Simulating the linear wave problem remains of engineering relevance, as this theory continues to be actively investigated in various contexts. One notable application is in wave scattering analysis. For instance, Kushwaha *et al.* recently carried out a series of studies based on linear wave theory, examining wave scattering by arrays of floating circular plates²² and circular cylinders²³ over a porous seabed. Additionally, Porter *et al.*²⁴ investigated wave scattering from a bottom-mounted circular cylinder using linear wave theory. In the current study, we employ PINNs to solve for the velocity components beneath the free surface of a linear wave based on potential theory. This is a first step toward solving more complicated problems based on potential theory in many engineering applications. Particularly, the current study investigates different approaches for incorporating boundary conditions and the determination of the unknown wave angular frequency.

The contributions of this study to the application of PINNs for free surface problems based on potential theory are threefold. First, the impact of applying periodic boundary conditions on the prediction accuracy of PINNs is thoroughly investigated. Both soft and hard constraints are considered, and a customized approach for hard constraints is developed. To the best of the authors’ knowledge, this investigation has not yet been conducted for free surface problems with inviscid flows. The method developed in this paper, to impose the periodic boundary conditions as hard constraints, is based on the general methods introduced in earlier studies.^{12,13,25}

Second, the enforcement of the kinematic bottom boundary condition (KBBC) as a hard constraint is explored using two trial

functions. These trial functions have been developed based on the methods introduced in earlier works.^{12,26,27} The design process of these trial functions, including their requirements and limitations, is explained thoroughly. This offers a practical guide for formulating trial functions for different problems.

Third, this study addresses the reconstruction of the free surface profile, treating the wave angular frequency, ω , as an unknown parameter and allows the PINN to infer ω by incorporating it into two loss terms. The network relies solely on the linear wave theory governing equations and boundary conditions, without requiring labeled data. While this approach shares similarities with inverse problems, it remains a direct problem, as no external data or measurements are involved, only the fundamental equations.

As the linear wave theory serves as the foundation for many wave modeling techniques, the present study provides a stepping stone toward extending PINNs to more advanced wave models. The proposed PINN framework, which allows for strict enforcement of boundary conditions and accurate reconstruction of key wave parameters such as the angular frequency, has potential applications in various marine and coastal engineering contexts. These include the prediction of wave-induced velocities around offshore structures, wave scattering analysis, and load estimation on floating or bottom-mounted devices. Such capabilities are particularly relevant in scenarios where traditional numerical methods may be insufficient or where labeled data are scarce. Furthermore, the ability of PINNs to infer unknown parameters opens new avenues for inverse modeling tasks, including wave field reconstruction and data assimilation in oceanography. While this study focuses on linear waves, the methodology lays the groundwork for addressing more complex problems such as strongly nonlinear wave propagation and wave–body interaction, by building upon the PINN architecture and insights developed herein.

The remainder of this paper is organized as follows: Sec. II presents the theoretical background of linear wave theory and the formulation of PINNs used in this study. Section III discusses the implementation of soft and hard boundary conditions within the PINN framework. Section IV introduces the scenarios and metrics used to verify the PINN results against analytical solutions. The results and their analysis are provided in Sec. V, where different PINN configurations are evaluated. Finally, the main conclusions and recommendations for future work are summarized in Sec. VI.

All code accompanying this manuscript is available at: <https://github.com/M-Sheikholeslami/PINNs-with-soft-and-hard-constraints-for-linear-waves>.

II. FRAMEWORK OF PINNs FOR LINEAR WAVE DYNAMICS

PINNs are variants of neural networks that incorporate the governing equations, the initial conditions (ICs), and the BCs of a problem into their architecture, rather than relying solely on labeled data. In this section, we explore how PINNs can be effectively applied to solve problems related to linear waves. We outline the theoretical foundation, describe the architecture necessary to solve wave dynamics, and discuss the various challenges and procedures involved in modeling with PINNs, including the design of the computational domain, optimization techniques, and the selection of hyperparameters.

A. Theory of linear wave (airy wave)

The linear wave problem describes the velocity potential distribution induced by small-amplitude waves with a sinusoidal profile. The derivation of the linear wave problem from general potential theory can be found in the work by Lewandowski.³ A schematic view of the wave and the spatial domain, constrained between the wave and the sea bottom, is shown in Fig. 1. This domain has a depth of h , a length of L , and a sinusoidal free surface, η , with an amplitude of A .

The governing equation of this problem is the Laplace equation for the velocity potential ϕ , which is given by

$$\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial z^2} = 0. \quad (1)$$

The two boundary conditions for the free surface, i.e., dynamic free surface boundary condition (DFSBC), and kinematic free surface boundary condition (KFSBC) are linearized and defined, respectively, as

$$\frac{\partial \phi}{\partial t} = -g\eta \quad \text{at } z = 0 \quad (\text{DFSBC}), \quad (2)$$

$$\frac{\partial \phi}{\partial z} = \frac{\partial \eta}{\partial t} \quad \text{at } z = 0 \quad (\text{KFSBC}). \quad (3)$$

The parameter η is the free surface elevation, as shown in Fig. 1, and is defined to have a sinusoidal profile as

$$\eta(x, t) = A \cos(kx - \omega t), \quad (4)$$

where k is the wave number and ω is the wave angular frequency. The use of cosine in Eq. (4) is consistent with the formulation presented by

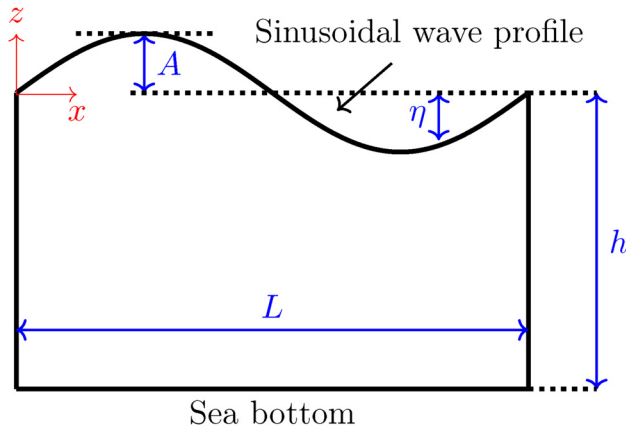


FIG. 1. Schematic representation of the spatial domain governed by linear wave theory.

Lewandowski.³ Moreover, since cosine and sine differ only by a phase shift, this choice does not affect the generality of the results. The DFSBC is the linearized Bernoulli equation and can be seen as a simplified version of the momentum conservation equation. The KFSBC guarantees that no flow crosses the free surface. The same role at the bottom of the domain is played by the kinematic bottom boundary condition (KBBC), defined as

$$\frac{\partial \phi}{\partial z} = 0 \quad \text{at } z = -h \quad (\text{KBBC}), \quad (5)$$

which essentially ensures that the bottom wall boundary is impenetrable.

B. Physics-informed neural networks (PINNs)

PINNs belong to a specific class of artificial neural networks that incorporate the physical constraints of a problem into the training process. Artificial neural networks serve as universal function approximators, capable of learning complex mappings between inputs and outputs, which can be used as a surrogate model. Once the network architecture, such as the number of layers and neurons, is designed, the remaining unknowns are the weights and biases of the network, which are determined through an optimization procedure called training. In order to design and train a PINN, we should go through three main stages, as shown in Fig. 2. These stages are explained in Subsections II B 1–II B 3, numbered as the stages.

1. Neural network as a surrogate model

Among the various types of neural networks, fully connected feed-forward neural networks are most commonly used in PINNs. These networks are composed of an input layer, one or more hidden layers, and an output layer. Each layer contains computational units called neurons. A neuron receives a set of input values, computes a weighted sum, adds a bias term, and applies a nonlinear function called the activation function. The weights determine the strength or importance of each input in the summation, while the bias allows the activation function to be shifted, enabling the model to better fit the data. The hidden layers are called hidden because their outputs are not directly observed, as they exist between the input and output layers and allow the network to learn complex, nonlinear relationships. Let the input be denoted as (\mathbf{x}, t) , where $\mathbf{x} \in \mathbb{R}^d$ represents the spatial coordinates and t is the temporal variable. The network outputs a scalar prediction $\hat{u}(\mathbf{x}, t; \theta) \in \mathbb{R}$, where θ denotes all trainable parameters of the model, including weights and biases. Each layer computes a transformation of the form

$$\mathbf{z}^{(l)} = \sigma(W^{(l)}\mathbf{z}^{(l-1)} + \mathbf{b}^{(l)}), \quad l = 1, 2, \dots, L, \quad (6)$$

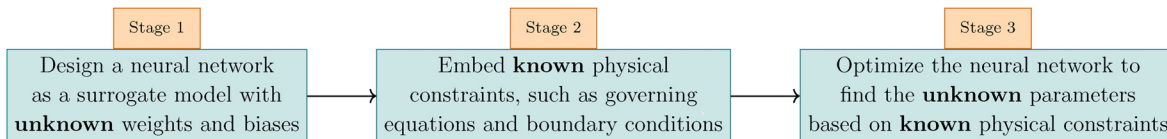


FIG. 2. Three main stages in the design and training of a PINN. The unknown parameters are the weights and biases of the network that are determined through and optimization procedure.

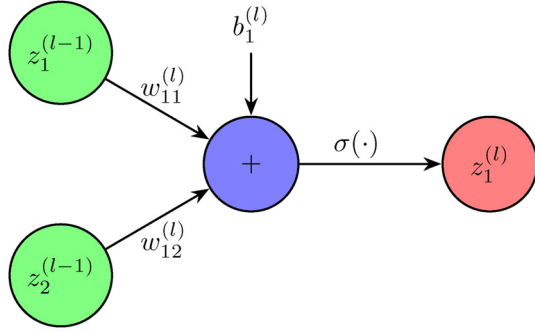


FIG. 3. Illustration of a single neuron transformation in a fully connected feed-forward neural network. The neuron (in layer l) is connected to two neurons from the previous layer ($l-1$). The inputs $z_1^{(l-1)}$ and $z_2^{(l-1)}$, which are scalar components of the vector $\mathbf{z}^{(l-1)}$, are linearly combined using scalar weights $w_{11}^{(l)}$ and $w_{12}^{(l)}$ (entries of the weight matrix $W^{(l)}$) and a scalar bias $b_1^{(l)}$ (an entry of the bias vector $\mathbf{b}^{(l)}$). The result is passed through a nonlinear activation function $\sigma(\cdot)$ to produce the output $z_1^{(l)} = \sigma(w_{11}^{(l)}z_1^{(l-1)} + w_{12}^{(l)}z_2^{(l-1)} + b_1^{(l)})$, which is the first component of the vector $\mathbf{z}^{(l)}$.

where $W^{(l)}$ and $\mathbf{b}^{(l)}$ are the weight matrix and bias vector of layer l , respectively, and $\mathbf{z}^{(0)} = (\mathbf{x}, t)$ is the input to the network. The function $\sigma(\cdot)$, called the activation function, introduces nonlinearity into the network, which is essential for approximating nonlinear mappings. The final layer produces the output $\hat{u}(\mathbf{x}, t; \theta) = \mathbf{z}^{(L)}$.

A visual representation of this transformation is provided in Fig. 3, where the weighted sum of inputs from the previous layer, along with a bias term, is passed through a nonlinear activation function to produce the neuron's output. This forms the core computational unit of a fully connected feed-forward neural network.

The trainable parameters of the network, θ (composed of W and \mathbf{b} , i.e., weights and biases), are the unknowns that we need to find. Their optimal values are not known *a priori* and must be discovered through a training process based on the known physical constraints of the problem. The training process is an optimization process involving minimizing a loss function that measures how well the network outputs $\hat{u}(\mathbf{x}, t; \theta)$ satisfies the known physical constraints of the problem. Sections II B 2 and II B 3 elaborate on how this loss function is calculated in PINNs and how the loss values are used in optimizing the weights and biases of PINNs.

2. Embedding known physical constraints

What distinguishes PINNs from purely data-driven networks is the loss function, which imposes the known governing physics of the problem as constraints for finding the unknowns θ . These constraints of the problem can be formulated as the partial differential equations (PDEs) that govern the problem, along with the initial and boundary conditions.⁴ Let the governing PDE be written as

$$N[u(\mathbf{x}, t)] = 0, \quad \mathbf{x} \in \Omega, \quad t \in (0, T], \quad (7)$$

where $N[\cdot]$ is a nonlinear differential operator, and $u(\mathbf{x}, t)$ is the true solution. Here, $\Omega \subset \mathbb{R}^d$ denotes the spatial domain of interest and T is the final time of interest. The initial and boundary conditions are given by

$$u(\mathbf{x}, 0) = g(\mathbf{x}), \quad \mathbf{x} \in \Omega, \quad (8)$$

$$u(\mathbf{x}, t) = h(\mathbf{x}, t), \quad \mathbf{x} \in \partial\Omega, \quad t \in (0, T], \quad (9)$$

where $\partial\Omega$ denotes the boundary of the spatial domain Ω . To train the network to satisfy these physical constraints, the loss function is constructed by forming the total loss L_T as

$$L_T = L_{\text{PDE}} + L_{\text{IC}} + L_{\text{BC}}, \quad (10)$$

where each term penalizes the residual of the corresponding condition

$$L_{\text{PDE}} = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| \mathcal{N}[\hat{u}(\mathbf{x}_i^f, t_i^f; \theta)] \right|^2, \quad (11)$$

$$L_{\text{IC}} = \frac{1}{N_0} \sum_{i=1}^{N_0} \left| \hat{u}(\mathbf{x}_i^0, 0; \theta) - g(\mathbf{x}_i^0) \right|^2, \quad (12)$$

$$L_{\text{BC}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \left| \hat{u}(\mathbf{x}_i^b, t_i^b; \theta) - h(\mathbf{x}_i^b, t_i^b) \right|^2. \quad (13)$$

The loss function encodes the physical constraints of the model and serves as the mathematical formulation used to infer the unknown parameters θ . In the standard formulation of PINNs, these known physical constraints are incorporated into the loss function, following the approach introduced by Raissi *et al.*⁴ However, it should be noted that known physical constraints can also be embedded directly into the neural network architecture itself. This alternative approach is referred to as the use of hard constraints, and two specific types of such constraints are described later in this paper.

3. Optimizing the neural network to find unknown parameters based on known physical constraints

The optimization process aims to determine the unknown neural network parameters θ such that the network output $\hat{u}(\mathbf{x}, t; \theta)$ approximates the solution and satisfies the known physical constraints of the problem by minimizing the loss function²⁸ as

$$\theta^* = \arg \min_{\theta} L_T(\theta). \quad (14)$$

As illustrated in Fig. 4, the process begins with the initialization of θ , followed by iterative updates. In each iteration, commonly referred to as an epoch, the network undergoes forward propagation, where the predictions $\hat{u}(\mathbf{x}, t; \theta)$ are computed at a set of collocation points (x_i, z_i, t_i) distributed across the spatiotemporal domain, including the boundaries. The required spatial and temporal derivatives of the output are also obtained during this step using automatic differentiation (AD),²⁹ which computes derivatives such as $\partial \hat{u} / \partial x$, $\partial^2 \hat{u} / \partial x^2$, and $\partial \hat{u} / \partial t$ by recursively applying the chain rule through the computational graph, as in

$$\frac{\partial \hat{u}}{\partial x} = \sum_j \frac{\partial \hat{u}}{\partial h_j} \frac{\partial h_j}{\partial x}, \quad (15)$$

where h_j are intermediate variables in the network (e.g., outputs of hidden neurons or layer activations), and the index j runs over the nodes that contribute to the computation of \hat{u} . These variables depend on the network parameters (weights and biases), but they are not independent unknown parameters themselves.

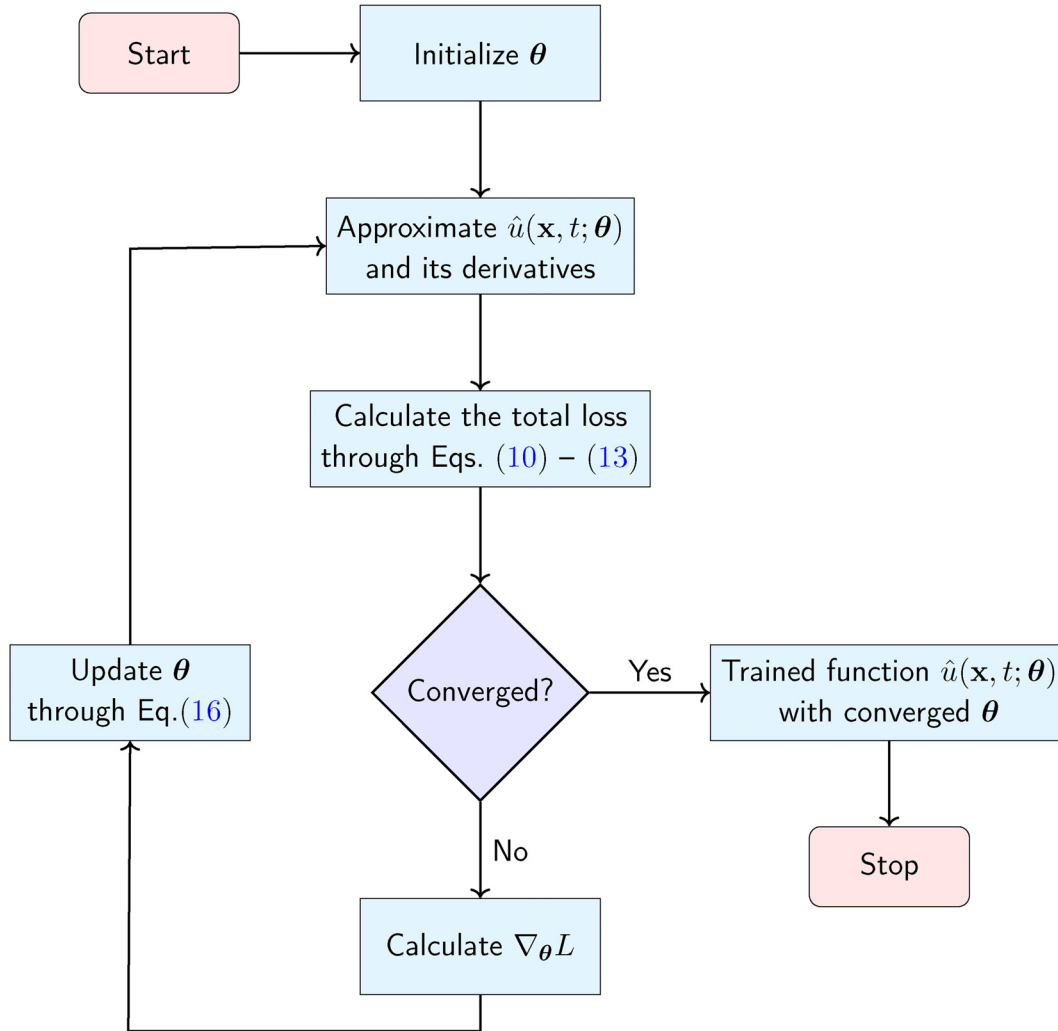


FIG. 4. The basic optimization process of an artificial neural network.

The network output and its derivatives with respect to the inputs are then substituted into the loss function defined by Eqs. (10)–(13), which quantify the extent to which the PINN approximation violates the known physical constraints. If the computed loss does not satisfy the convergence criteria, the network enters the backward propagation phase, where the gradient of the total loss with respect to each parameter in θ is computed using AD. These gradients guide the parameter updates by indicating how the weights and biases should change to reduce the loss. Specifically, the update proceeds in the direction of the negative gradient of the loss, which corresponds to the steepest descent in loss value. The update rule follows a gradient-based³⁰ optimization algorithm, typically applied as

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \nabla_{\theta} L(\theta^{(k)}), \quad (16)$$

where k denotes the iteration number in the optimization process, and α is the learning rate, a hyperparameter that controls the step size, i.e., the magnitude of each update. A larger step size can accelerate

convergence but risks overshooting minima, while a smaller step size may lead to more stable but slower convergence. The gradient $\nabla_{\theta} L$ is computed using AD. In this context, AD is also applied during the backward propagation phase to compute the gradient of the total loss with respect to the network parameters, using the chain rule²⁹ as

$$\frac{\partial L}{\partial \theta_i} = \sum_j \frac{\partial L}{\partial h_j} \frac{\partial h_j}{\partial \theta_i}, \quad (17)$$

where $\theta_i \in \theta$ is the i th parameter of the neural network (e.g., a weight or bias), and h_j is the j th intermediate variable (such as the output of a hidden neuron) that depends on θ_i . The summation runs over all computational nodes h_j influenced by θ_i . This recursive application of the chain rule enables efficient and exact computation of parameter gradients without relying on numerical approximations.

In this study, more advanced variants of gradient descent are employed to improve convergence speed and numerical stability. The

Adam optimizer,³¹ for example, adaptively adjusts the learning rates for each parameter based on statistical estimates of gradient moments. Alternatively, quasi-Newton methods such as L-BFGS³² use approximations of the inverse Hessian matrix to perform more informed updates of the unknown parameters, often leading to faster convergence. Both approaches are extensions of gradient descent, built to handle optimization under known physical constraints.

Each pass through this computational loop, from computing the prediction and its derivatives to updating θ , constitutes one optimization epoch. The process continues until convergence, when the network yields a solution consistent with the known physical constraints. This means the network output satisfies the governing equation [Eq. (7)] along with the initial and boundary conditions [Eqs. (8) and (9)].

C. PINNs for the linear wave problem

This section outlines the process of defining the computational domain, including the full ranges of x , z , and t , as well as the architecture and loss function of the PINNs developed to solve the linear wave problem. It also discusses the optimization algorithms and hyperparameters used in training these networks.

1. Computational domain

A three-dimensional computational domain is designed for this problem, as shown in Fig. 5. The x -length of this domain is decided to cover one wavelength, λ , of the free surface wave profile. The wave number, k , in Eq. (4), can be found through $k = 2\pi/\lambda$. The t -length of the domain is set equal to the temporal periodicity, T , of the free surface wave, which can be found through $T = 2\pi/\omega = \lambda k/\omega$. Since the linear wave theory is based on the assumption that the wave amplitude, A , should be much smaller than the wavelength, λ ,³³ a wave amplitude of $A = \lambda/(200\pi)$ is selected for all cases.

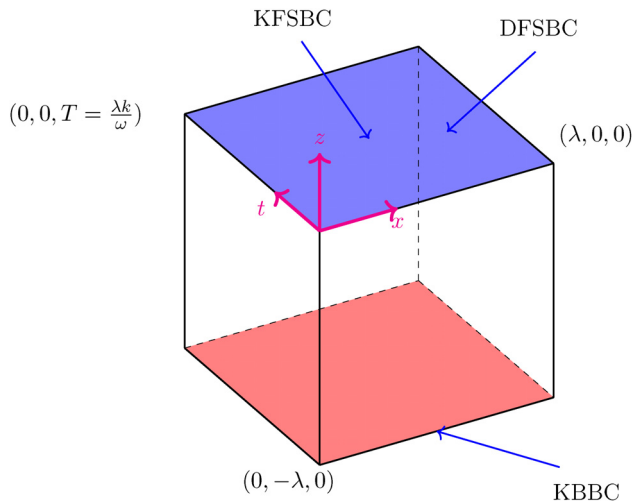


FIG. 5. Computational domain. The top boundary, where the KFSBC and DFSBC conditions are applied, is shown in blue, while the bottom boundary, where the KBBC condition is applied, is shown in red.

2. The architecture and loss function

A schematic overview of the basic PINNs, known as vanilla-PINNs,³⁴ used to solve the linear wave problem, is shown in Fig. 6. At the core of a vanilla-PINN framework, there is a fully connected artificial neural network that receives the time and coordinates of the points in the spatial-temporal computational domain as inputs and provides the unknown variable that is to be solved as outputs. The spatial-temporal points fed into the PINN are known as collocation points.⁴ The input layer of a PINN designed for the linear wave problem receives three dimensions, namely, the spatial coordinates of x and z , along with the temporal coordinate t . Unlike classical numerical methods for solving partial differential equations, PINNs treat the temporal dimension similarly to spatial dimensions.⁴ Consequently, for an unsteady two-dimensional linear wave, a three-dimensional computational domain is covered by collocation points.

The incorporation of the governing equation [Eq. (1)] and three BCs [Eqs. (2)–(5)] takes the first and second derivatives of ϕ with respect to z and t . In Fig. 6, loss functions L_T , L_{GE} , L_{KBBC} , and L_{KFSBC} stand for the total loss, and the loss functions for the governing equation, KBBC and KFSBC, respectively, which are defined as

$$L_{GE} = \sum_{i=1}^{n_x} \sum_{j=1}^{n_z} \sum_{k=1}^{n_t} \left| \frac{\partial^2 \phi(x_i, z_j, t_k)}{\partial x^2} + \frac{\partial^2 \phi(x_i, z_j, t_k)}{\partial z^2} \right|^2, \quad (18)$$

$$L_{KBBC} = \sum_{i=1}^{n_x} \sum_{k=1}^{n_t} \left| \frac{\partial \phi(x_i, -h, t_k)}{\partial z} \right|^2, \quad (19)$$

$$L_{KFSBC} = \sum_{i=1}^{n_x} \sum_{k=1}^{n_t} \left| \frac{\partial \phi(x_i, 0, t_k)}{\partial z} - \frac{\partial \eta(x_i, t_k)}{\partial t} \right|^2. \quad (20)$$

After the calculation of the different loss terms, the rest of the training procedure in a vanilla-PINN is similar to the common procedure of training classic neural networks, using specific optimization methods. The setup of the PINNs used in this study is presented in detail in Subsection II C 3.

3. Hyperparameters and optimization scheme

A sensitivity analysis was conducted on the PINN architecture for solving linear wave dynamics, with the results summarized in Table I. The study examines the influence of three key hyperparameters: the number of layers, the number of neurons per layer, and the number of collocation points. The total error represents the normalized average error of the PINN prediction compared to the analytical solution, while the periodic error quantifies discrepancies in the predicted velocity at the periodic boundaries along the x dimension. The abbreviation Std. refers to the standard deviation of the total error across repeated runs. Among the tested configurations, the model with 16 layers, ten neurons per layer, and 41^3 collocation points demonstrated the most reliable performance, achieving the lowest average error and the smallest standard deviation across all runs. Each configuration was trained 100 times with different random initializations to capture variability and ensure statistical reliability. This extensive sampling reduces the risk of overfitting to specific initialization patterns and highlights the selected configuration's robustness to randomness in parameter initialization.

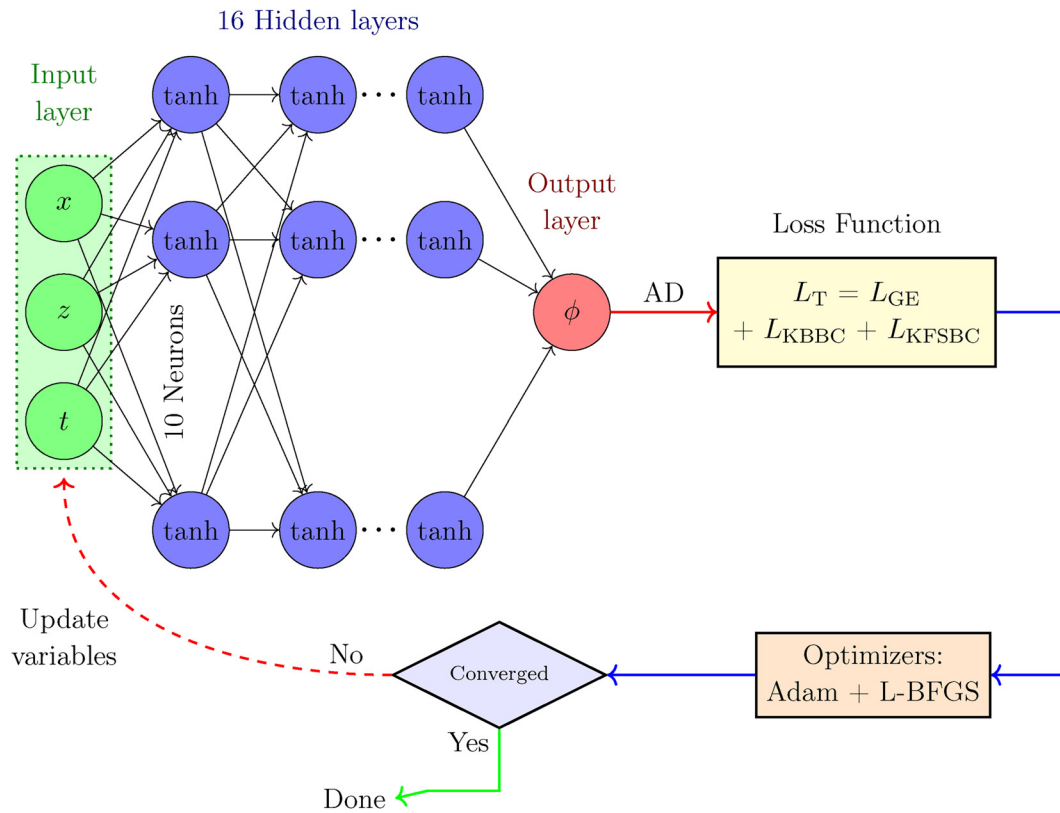


FIG. 6. Schematic of the PINN used for the linear wave problem.

TABLE I. PINN performance for linear wave problem under different configurations.

Layers	Neurons	N_{cp}	u Total error (Std) [%]	w Total error (Std) [%]	u Period. error (Std) [%]	w Period. error (Std) [%]	Time (Std) [s]
4	10	21^3	18.22 (28.04)	0.81 (0.57)	2.31 (1.53)	2.93 (2.17)	40.91 (5.11)
8	5	21^3	24.95 (90.93)	0.54 (0.40)	1.71 (1.31)	1.69 (1.54)	48.08 (6.03)
8	10	21^3	6.64 (12.19)	0.44 (0.38)	1.30 (1.05)	1.52 (1.42)	61.05 (5.82)
8	20	21^3	17.04 (10.28)	5.09 (2.28)	16.46 (6.11)	14.63 (4.47)	34.07 (1.03)
16	10	11^3	3.18 (5.20)	0.57 (1.36)	1.09 (0.95)	1.05 (0.92)	88.76 (8.75)
16	10	21^3	2.68 (6.18)	0.33 (0.46)	1.05 (1.47)	1.00 (1.14)	145.85 (19.02)
16	10	41^3	2.31 (3.37)	0.27 (0.19)	0.84 (0.56)	0.84 (0.63)	250.69 (35.90)
16	10	61^3	2.97 (10.85)	0.26 (0.20)	0.82 (0.57)	0.84 (0.73)	579.76 (94.62)
16	5	21^3	14.22 (31.17)	0.90 (1.35)	2.53 (2.73)	2.51 (2.97)	84.29 (10.10)
16	20	21^3	12.64 (10.62)	4.28 (1.64)	14.69 (4.69)	13.40 (4.77)	60.84 (1.57)
32	10	21^3	6.11 (18.40)	0.44 (0.45)	1.42 (1.46)	1.27 (1.26)	166.98 (16.69)

The distribution of collocation points in all cases in the current study is equidistant throughout the domain, as it is also the case for some other studies in the literature.^{35,36} To ensure that the trained model is not overfit to the training collocation points, the results presented in Secs. IV and V were extracted at points other than the

collocation points. To this end, an equidistant grid of 31^3 is utilized for all the post-processing analysis.

Similar to classical neural networks, PINNs use activation functions to represent different levels of non-linearity.³⁷ Hyperbolic tangent is a popular choice to be used in the design of PINNs,^{4,5,34} and

has been used in the current study. However, the output layer of the PINNs has an identity activation function, as in the work of Berrone *et al.*³⁸

Learning rates between 10^{-3} and 10^{-4} have been used to solve a wide range of problems^{5,11,12,39,40} using PINNs. Here, a learning rate of 10^{-3} is chosen. A two-stage optimization procedure is employed for training the model variables. In the first stage, the Adam optimizer³¹ is used for 1000 epochs. Subsequently, the optimization continues with the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm, a quasi-Newton, gradient-based method.³² The L-BFGS optimization terminates automatically based on the increment tolerance. This two-stage optimization approach has been widely employed in the literature.^{5,12,14,39,41}

III. KINEMATIC BOTTOM BC AND PERIODIC BC FOR PINN MODELING

An accurate imposition of BCs is crucial for ensuring the physical fidelity of the model. Boundary conditions can be enforced using various methods, generally categorized into soft and hard approaches. Sections III A–III C explore these approaches in detail for imposing the periodic and kinematic bottom boundary conditions in the PINN framework.

A. Soft and hard approach to imposing boundary conditions in PINNs

The soft approach for imposing BCs involves adding a separate loss term for the imposed BCs to the total loss function as in Eq. (10). During optimization, the total loss function is minimized as the sum of all loss terms, which can result in imperfect satisfaction of the boundary condition. In contrast, the hard constraint approach embeds the boundary condition directly into the architecture of the PINN, eliminating the need for a dedicated loss term. This ensures that the neural network output inherently satisfies the boundary conditions. In this study, two different types of hard BCs for the periodic BC (PBC) and KBBC are investigated. These two different hard constraints target two different parts of a PINN, which is denoted schematically in Fig. 7.

B. Hard periodic boundary condition

Imposing a periodic BC using a periodic layer is a robust method to satisfy the periodic BC to machine precision. In the problem under investigation, the flow field is periodic in the x direction. Thus, the

periodic layer, placed between the input layer and the first hidden layer of the PINN, transforms the input, x , into a sum of periodic functions with the same periodicity, α . The periodic layer is composed of several neurons, each representing a periodic function (P) of the input of interest, x , as

$$P_i(x) = \tanh(\sin(\alpha x + B_i)), \quad (21)$$

where the phase angle, B_i , is trainable, and α , which is the periodicity of the solution with respect to x , is not trainable. For the linear wave theory, α is equal to the periodicity of the free surface profile with respect to x , i.e., $\alpha = 2\pi/\lambda = 1$. It should be noted that the tanh operator in Eq. (21) is a nonlinear activation function and does not necessarily need to be the hyperbolic tangent function. This function is used to be consistent with the activation functions in the other layers of the PINN.

As shown in Fig. 8, the periodic layer can consist of several neurons that apply Eq. (21) to the input dimension with respect to which the output is periodic, specifically x in Fig. 8. The other dimensions go through an identity function, I , which keeps them unchanged as no periodicity is imposed in the z or t dimensions. The periodic layer in this study has 12 neurons, of which ten are connected to the x -dimension input neuron applying the P_i functions. The neurons of the z and t dimensions in the input layer are connected to a single neuron of identity function (I), keeping them unchanged. This configuration, with ten neurons connected to the x -dimension, was chosen to align with the neuron count used in the hidden layers, ensuring architectural consistency across the network.

C. Hard kinematic bottom boundary condition

This section develops trial functions for imposing the KBBC as a hard constraint. The logic behind using trial functions to enforce constraints has already been discussed in the literature^{12,26,27} and is not reiterated here. Incorporating a trial function strictly enforces the BCs of interest. A trial function can be incorporated into the PINN architecture, as shown in Fig. 9. As an example, consider a PINN with output $N(x)$ designed to solve a one-dimensional (1D) partial differential equation $\psi(x)$. A trial function such as $\psi_t(x) = A + xN(x)$ can impose the Dirichlet BC of $\psi(0) = A$.

A proper trial function should have the following two requirements:

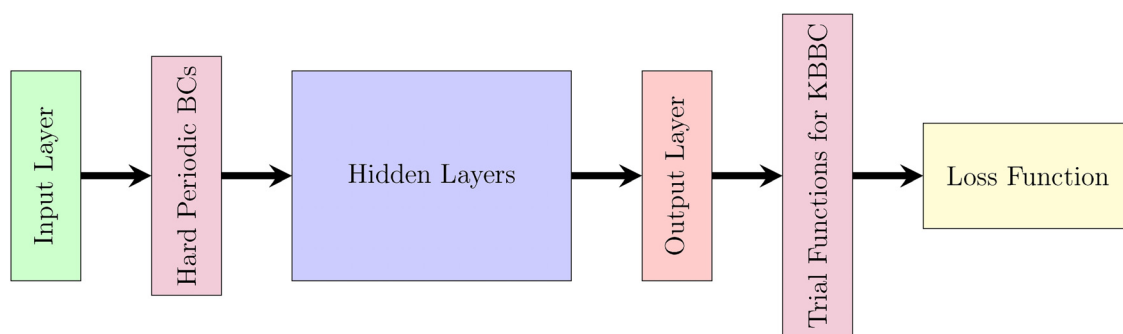


FIG. 7. Positions in the PINN where different hard BCs should be applied.

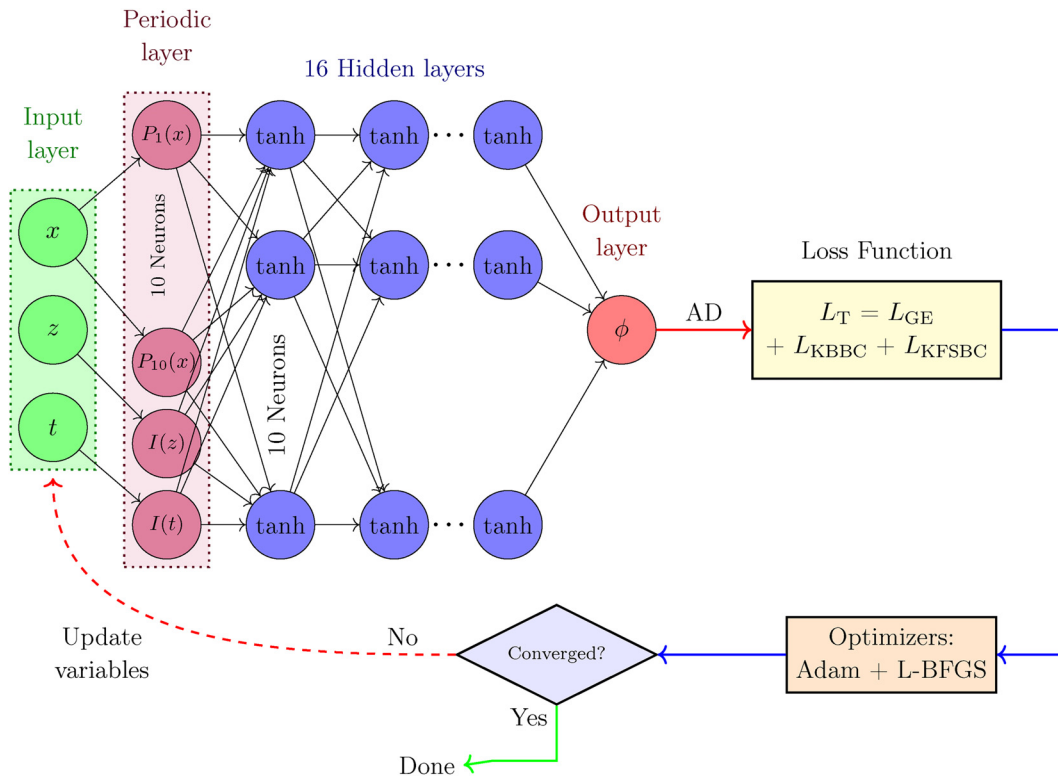


FIG. 8. Schematic of the PINN used for the linear wave problem, having a periodic layer to meet the periodic BC of the problem.

Requirement 1: At the boundaries of interest, the trial function must satisfy the boundary conditions, and all terms, including the output of the neural network, must zero out.

Requirement 2: Throughout the rest of the domain, the trial function must include the output of the PINN and/or its derivatives.

For the development of a trial function tailored for enforcing a specific constraint, the outcome has to fulfill the two requirements mentioned before. As a starting point to develop a trial function tailored for enforcing a homogeneous Neumann BC, i.e., KBBC, we can begin with a similar case. In this case, it is the trial function provided by Lagaris *et al.*²⁶ for the Poisson problem, i.e.,

$$\frac{\partial^2}{\partial x^2} \psi(x, y) + \frac{\partial^2}{\partial y^2} \psi(x, y) = f(x, y) \quad (22)$$

with mixed BCs of $\psi(0, y) = f_0(y)$, $\psi(1, y) = f_1(y)$, $\psi(x, 0) = g_0(x)$, and $\frac{\partial}{\partial y} \psi(x, 1) = g_1(x)$. The suggested trial function by Lagaris *et al.*²⁶ for imposing these four BCs is

$$\Psi_t(x, y) = B(x, y) + x(1-x)y \left[N(x, y, \vec{p}) - N(x, 1, \vec{p}) - \frac{\partial}{\partial y} N(x, 1, \vec{p}) \right], \quad (23)$$

$$B(x, y) = (1-x)f_0(y) + xf_1(y) + g_0(x) - [(1-x)g_0(0) + xg_0(1)] + y\{g_1(x) - [(1-x)g_1(0) + xg_1(1)]\}, \quad (24)$$

where N is the output of the neural network and \vec{p} are the weights and biases of the neural network. In the suggested trial function [Eq. (23)] for the Poisson problem [Eq. (22)], the multipliers of x , $(1-x)$, and y in Eq. (23) relate, respectively, to the three Dirichlet conditions of $\psi(0, y) = f_0(y)$, $\psi(1, y) = f_1(y)$, and $\psi(x, 0) = g_0(x)$. Since the trial function we are looking for is supposed to only impose the KBBC, and KBBC is a homogeneous Neumann BC, we do not need those three multipliers. The term $B(x, y)$ in Eq. (23) is also responsible to hold the value of the Poisson function or its derivatives at the boundaries, which is zero in our case. Therefore, Eq. (23) is updated accordingly and turns to

$$\phi_t(x, z, t) = N(x, z, t) - N(x, -h, t) - \frac{\partial}{\partial z} N(x, -h, t), \quad (25)$$

where $\Psi_t(x, y)$ in Eq. (23) has turned to $\phi_t(x, z, t)$ in order to conform with the linear wave problem terminology, and \vec{p} is not mentioned, since it is apparent that the output of the PINN is also a function of weights and biases. The derivative of Eq. (25) with respect to z on $z = -h$ is

$$\frac{\partial}{\partial z} \phi_t(x, z, t) \Big|_{z=-h} = \frac{\partial}{\partial z} N(x, z, t) \Big|_{z=-h}, \quad (26)$$

which includes a derivation of the PINN output and is not necessarily zero, which violates requirement 1. One way to resolve this problem is multiplying the bracket via a function of z , whose derivatives with respect to z equal the original function. The only function with this

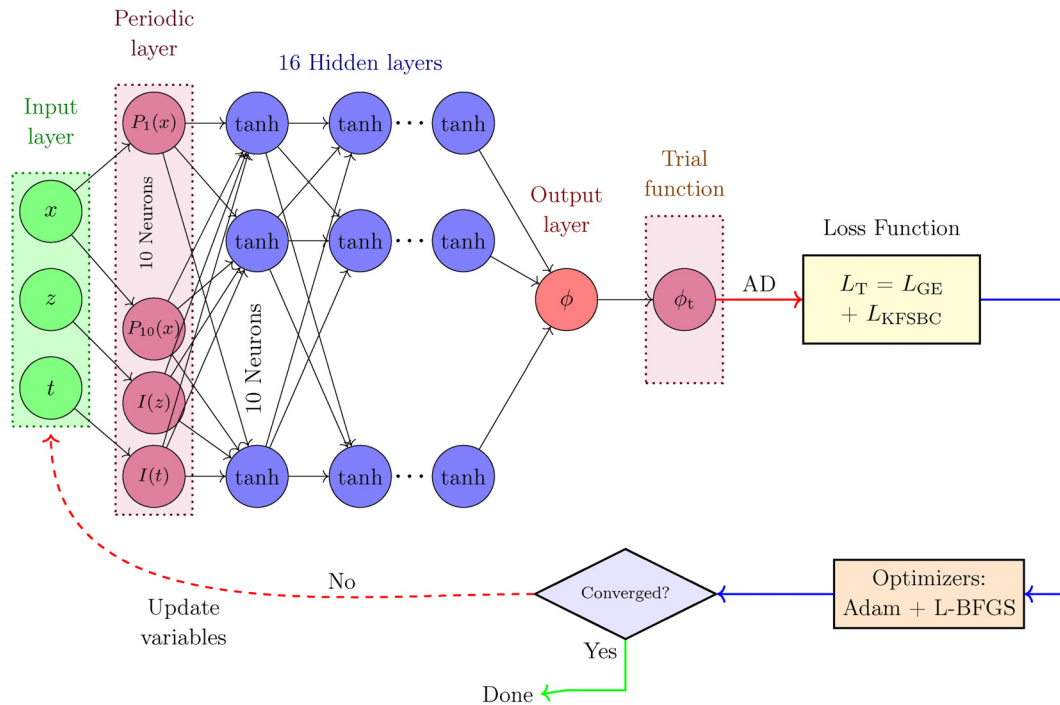


FIG. 9. Schematic of the PINN used for the linear wave problem, having a periodic layer to meet the periodic BC of the problem, and a trial function to ensure that the neural network output adheres to the required physical constraints (enforcing KBBC in this study).

characteristic is the exponential function.⁴² The trial function in Eq. (25) turns to

$$\phi_t(x, z, t) = e^z \left[N(x, z, t) - N(x, -h, t) - \frac{\partial}{\partial z} N(x, -h, t) \right]. \quad (27)$$

The derivative of Eq. (27) with respect to z on $z = -h$ is

$$\left. \frac{\partial}{\partial z} \phi_t(x, z, t) \right|_{x=-h} = 0, \quad (28)$$

which satisfies KBBC. Since the exponential function never becomes zero, the PINN output always exists in the function, and no unwanted Dirichlet BC is being imposed on the problem. Therefore, the trial function in Eq. (27) meets both requirements to be a trial function for imposing KBBC on the linear wave problem in the PINN.

For the second candidate for strictly imposing KBBC on the linear wave problem, the trial function for an initial value problem, with Cauchy initial condition introduced by Lagaris *et al.*,²⁶ has been chosen as a starting point. By adopting the trial function developed by Lagaris *et al.*²⁶ for enforcing KBBC, we can end up with

$$\phi_t(x, z, t) = (z + h)^2 N(x, z, t). \quad (29)$$

Therefore, two trial functions, i.e., Eqs. (27) and (29), are developed for imposing the KBBC. The trial function in Eq. (27) satisfies both previously mentioned requirements. Although the trial function in Eq. (29) meets requirement 1, it fails in satisfying requirement 2 at $z = -h$, where it becomes zero at the bottom of the domain. This condition is not necessarily true for potential flow. However, the values of ϕ at the bottom are not expected to vary significantly across x and t , and the level of ϕ generally does not influence its spatial derivatives

with respect to x and z , i.e., the velocity components u and w , which are the focus of this study.

IV. SCENARIOS AND METRICS FOR PINNs WAVE VERIFICATION

In this section, the analytical solution for the linear wave is presented as a benchmark, and the evaluation metrics are defined. The vanilla PINN is described as a baseline for comparison.

A. Analytical solution for the linear wave

In order to analyze different approaches to solving the linear wave problem, we compare the results with the analytical solution. The free surface wave is defined in Eq. (4). With this free surface profile, the wave problem stated in Eqs. (1)–(5) can be linearized and solved analytically to achieve the velocity potential distribution³ as

$$\phi(x, z, t) = \frac{Ag \cosh(k(h+z))}{\omega \cosh(kh)} \sin(kx - \omega t). \quad (30)$$

The application of the DFSBC condition, i.e., Eq. (2), provides the relation between the wave angular frequency, ω , and the wave number, k , which is called the dispersion relation, as

$$\omega^2 = gk \tanh kh. \quad (31)$$

The actual values of the velocity potential, ϕ , do not play any role in the governing equation and BCs. The values can end up at any level while still satisfying the governing equations and boundary conditions. It is only the gradients of ϕ , i.e., flow velocity components, that contribute

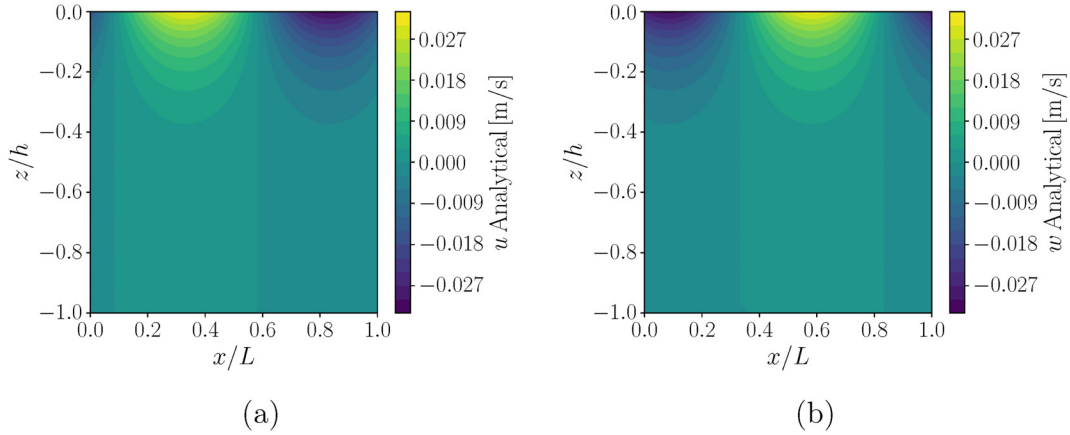


FIG. 10. Distributions of u (a) and w (b) obtained by the analytical solution at time $t = T/3$.

to the linear wave problem. Therefore, in this study, only the distributions of $d\phi/dx$ and $d\phi/dz$, which are equivalent to the velocity components in the x and z directions, i.e., u and w , respectively, are reported.

The derivatives of the analytical solution in Eq. (30), with respect to x and z , are the reference solutions that we compare to the PINN results to examine the accuracy of the calculations. The analytical solution of u and w at time $t = T/3$, with $k = 1$, is shown in Fig. 10.

B. Metrics to evaluate accuracy

The trainable variables of a network are usually initialized randomly before training. To mitigate the risk of dependence of the PINN results on such initialization, all the PINN models in Secs. IV C and V are trained 100 times with random initializations. All the errors in the PINN results are then calculated as the average error of those 100 training sessions. The total error for u , ϵ_u , is defined as

$$\epsilon_u[\%] = \frac{1}{n_x n_z n_t \max[u_{\text{Analytical}}(x, z, t)]} \sum_{k=1}^{n_t} \sum_{j=1}^{n_z} \sum_{i=1}^{n_x} \left| u_{\text{PINN}}(x_i, z_j, t_k) - u_{\text{Analytical}}(x_i, z_j, t_k) \right| \times 100, \quad (32)$$

where n_x , n_z , and n_t represent the number of collocation points in the x , z , and t directions, respectively. The total error for w , ϵ_w , can be found by replacing u with w in Eq. (32). The periodicity error of u , ϵ_{Pu} , is defined as

$$\epsilon_{\text{Pu}}[\%] = \frac{1}{n_z n_t \max[u_{\text{Analytical}}(x, z, t)]} \sum_{k=1}^{n_t} \sum_{j=1}^{n_z} \left| u_{\text{PINN}}(0, z_j, t_k) - u_{\text{PINN}}(L, z_j, t_k) \right| \times 100, \quad (33)$$

and the periodicity error of w , ϵ_{Pw} , can be found by replacing u with w in Eq. (33).

C. The vanilla PINN for the linear wave

In this subsection, the results of a vanilla PINN for the linear wave problem are presented. These results serve as reference values

that will be improved by imposing hard boundary conditions in Sec. V. We put the governing equation [Eq. (1)] and the kinematic boundary conditions, i.e., Eqs. (3) and (5), in the loss function to find the velocity potential distribution below the sinusoidal wave, described by Eq. (4). As the governing equation and boundary conditions are imposed as soft constraints, the PINN can be called vanilla PINN. Additionally, ω , calculated using Eq. (31), is assumed as an input in the vanilla PINN. The settings of this PINN are shown in Fig. 6.

The contours of u and w at time $t = T/3$, when $k = 1$ and $\omega = 3.13$, are presented in Fig. 11. This figure shows that the general distributions of u and w are captured relatively well by the PINN, compared to the analytic solution seen in Fig. 10. The average errors of the calculated u and w of the 100 training sessions of the vanilla PINN are 2.31% and 0.27%, with standard deviations of 3.37% and 0.19%, respectively. The distributions of the errors, or the discrepancies between the exact data (Fig. 10) and the PINN results, at $t = T/3$ for both u and w are also shown in Fig. 11. The distributions of the errors in these figures reveal that the regions of elevated error tend to form near the boundaries of x . This pattern was consistently observed across the majority of results from 100 sessions of training. Although the x -dimension in Fig. 11 spans one full wavelength, the distributions of u and w , as well as their error distributions, do not exhibit periodicity. This issue will be addressed in Sec. V.

V. RESULTS AND DISCUSSION

This section is divided into three subsections. In Secs. V A and V B, the free surface wave is considered to be fully known, and the values of the wave number, k , and the wave angular frequency, ω , are set so they satisfy the dispersion relation [Eq. (31)]. The impacts of the soft and hard periodic boundary conditions are studied in Secs. V A 1 and V A 2, respectively. Hard imposing of the KBBC is investigated in Sec. V B. In Sec. V C, the wave angular frequency, ω , is considered unknown. A PINN architecture is designed to find the distributions of u and w and the value of ω . Finding the correct value of ω by the PINN can be viewed as the solution of the dispersion relation in the analytical solution [Eq. (31)]. The calculated distributions of u and w in these subsections are compared to the analytical results (Fig. 10) to find the total error. The improvements caused by imposing the soft

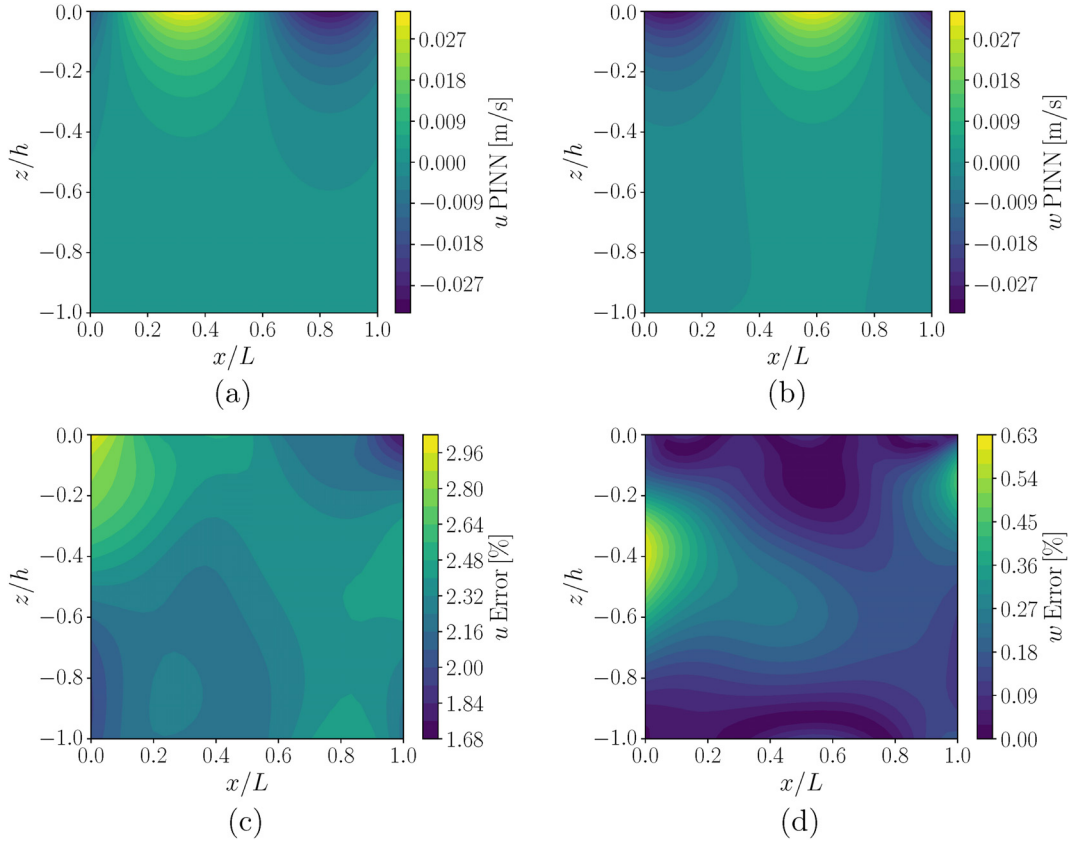


FIG. 11. Distributions of u (a) and w (b) and error distributions of u (c) and w (d) obtained by the vanilla PINN at time $t = T/3$.

and hard periodic BCs and the hard KBBC BC are examined by comparing the results with those of the vanilla PINN in Fig. 11.

A. Periodic boundary conditions

The analytical results in Fig. 10 show that the distributions of u and w are perfectly periodic in the x -direction. However, the distributions of the u and w velocity components obtained by the vanilla PINN (Fig. 11) do not preserve such perfect periodicity. The average ϵ_{Pu} and ϵ_{Pw} in the x -direction are 0.84% and 0.84%, with standard deviations of 0.56% and 0.63%, respectively.

To address the imperfect periodicity of the PINN results, we need to revisit the linear wave problem and how periodic BCs are imposed in the theoretical solution. The solution's periodicity in the x -direction is not explicitly enforced by any BCs. In the analytical approach, the periodicity of the wave profile imposed by the KFSBC automatically extends to the solution, velocity potential ϕ , and its derivatives. However, PINN transforms the problem into an optimization process, which does not necessarily maintain strict periodicity in the x -direction. Although the periodicity errors of the results in Sec. IV C might be acceptable for some applications, achieving more consistent periodicity may be crucial for other applications. Sections V A 1 and V A 2 explore and compare two methods for imposing periodic BCs on the linear wave problem, i.e., the soft and hard approaches.

1. Soft periodic BC

To impose the periodic BC in the x -direction as a soft constraint, additional loss terms are incorporated into the total loss function of the neural network. Three periodic loss terms are used at the lower and upper boundaries in the x direction to enforce periodic equality of ϕ and its spatial derivatives in the x and z directions (the u and w velocity components). Hence, the periodic loss terms are defined as

$$L_{P,1} = \sum_{k=1}^{n_t} \sum_{j=1}^{n_z} |\phi(0, z_j, t_k) - \phi(L, z_j, t_k)|^2, \quad (34)$$

$$L_{P,2} = \sum_{k=1}^{n_t} \sum_{j=1}^{n_z} \left| \frac{\partial \phi}{\partial z}(0, z_j, t_k) - \frac{\partial \phi}{\partial z}(L, z_j, t_k) \right|^2, \quad (35)$$

$$L_{P,3} = \sum_{k=1}^{n_t} \sum_{j=1}^{n_z} \left| \frac{\partial \phi}{\partial x}(0, z_j, t_k) - \frac{\partial \phi}{\partial x}(L, z_j, t_k) \right|^2. \quad (36)$$

Even though these loss terms are designed to enforce the periodicity of the solution, the resulting periodicity depends on how effectively the PINN minimizes these loss terms.

When these terms are added to the total loss function of the vanilla PINN, the loss function expands to six components. This new PINN has been trained 100 times with random initializations. The

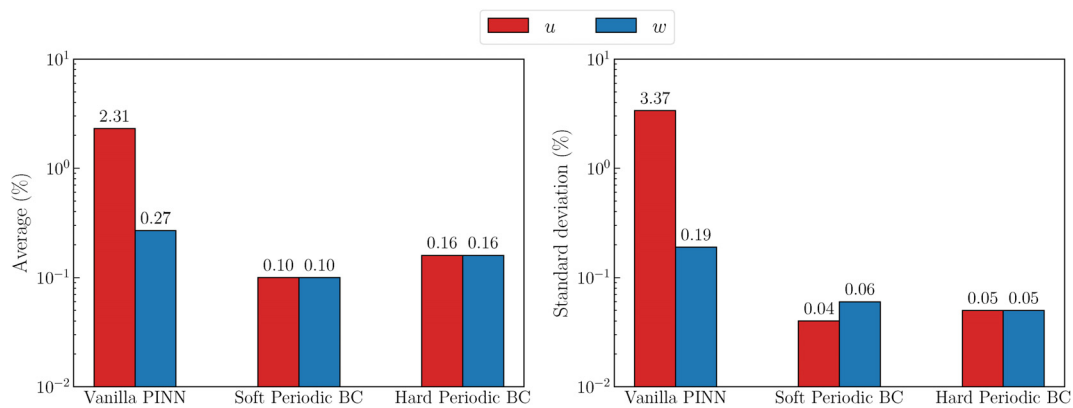


FIG. 12. Comparison of average and standard deviation of the total errors for different methods of dealing with periodic BCs.

results show that the incorporation of the periodicity-related loss terms reduces the average periodicity errors for u and w to 0.10% and 0.10%, respectively. The average training time was 220.28 s with a standard deviation of 41.01 s, which is lower than that of the vanilla PINN configuration reported in Table I. This reduction in training time can be attributed to the role of periodic boundary conditions, which simplify the optimization process.

Figure 12 compares the total errors and standard deviations of this case (*soft periodic*) with the case studied in Sec. IV C, named as *vanilla PINN*. The total average errors were reduced to almost 1/23 of the reference value for u and to less than 2/5 of the reference value of w , with significantly lower standard deviations. These results demonstrate that enforcing periodicity constraints effectively reduces the periodicity error, which leads to a more accurate prediction of the overall results.

2. Hard periodic BC

A hard periodic BC is applied in this section, as outlined in Sec. III B and depicted in Fig. 8. The PINN includes a periodic layer that

ensures a perfect periodicity of the solution ϕ . In fact, the periodicity errors in these cases are on the order of machine precision. This configuration is referred to as *hard periodic BC*.

The average and standard deviation of the total errors for different periodic BC constraints are compared in Fig. 12. The PINN with soft PBC is labeled as soft periodic BC. The results of the hard periodic BC PINN demonstrate a lower error than vanilla PINN, although it has slightly higher errors than the results of the soft periodic PINN, probably due to its higher computational demands. All cases used the same collocation points and optimization scheme, but the hard periodic BC PINN has ten more trainable variables in the periodic layer and needs more computational resources to reach the same accuracy as the soft periodic BC PINN. The average training time was 319.41 s with a standard deviation of 57.25 s, which is higher than that of both the vanilla PINN and the PINN with soft periodic boundary conditions. This increase can be attributed to the additional complexity introduced by the periodic layer, which makes the optimization process more computationally demanding. The conclusions drawn from the results in Fig. 12 are summarized in the flow chart presented in Fig. 13.

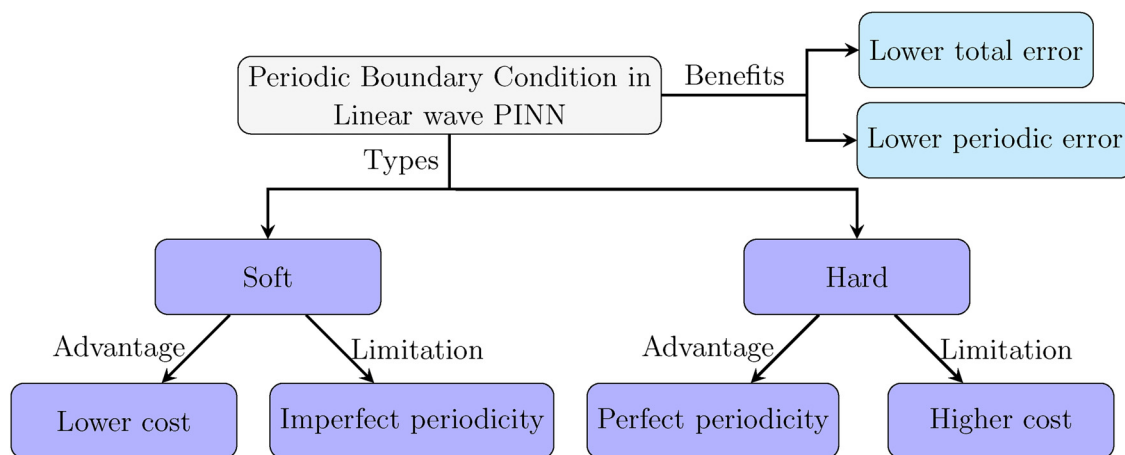


FIG. 13. Benefits of using the periodic BCs for modeling linear waves, and the advantages and limitations of each implementation approach.

For the remainder of the paper, the periodic boundary condition is imposed using the hard method. This approach is used to assess how other adjustments to the PINN perform when the periodic layer is included. Figure 14 shows the distributions of u and w at time $t = T/3$ of hard periodic. Comparing these results with the analytical distributions in Fig. 10 reveals that the general behavior of u and w is captured well.

B. Hard kinematic bottom BC

So far, the KBBC has been imposed using a dedicated loss term. This method, referred to as the soft approach, permits some error in satisfying the BCs, depending on the success of the optimization problem. Although this level of error may be acceptable in certain applications, strict enforcement of BCs is sometimes required. To reach a perfect satisfaction of the KBBC, this boundary condition is imposed as a hard constraint by adding a trial function to the PINN architecture, as explained in Sec. III C. Two different trial functions, introduced in Eqs. (27) and (29), are examined in this section to investigate the accuracy of the solution while the KBBC was imposed as a hard constraint.

A PINN architecture as illustrated in Fig. 9 was designed using the trial function of Eq. (27). The model was trained, and the results showed that the KBBC was satisfied to the machine precision, with average total errors of 0.30% for u and 0.48% for w . The average

training time was 571.53 s with a standard deviation of 119.74 s, which is significantly higher than that of the models employing the soft KBBC. This increase reflects the added difficulty in optimizing the network when this specific trial function is imposed. The same PINN was then trained to utilize the trial function from Eq. (29), also achieving perfect KBBC satisfaction, though with higher average errors of 2.36% for u and 2.10% for w . The average training time was 300.50 s with a standard deviation of 82.36 s, indicating that this trial function for the KBBC was more computationally efficient. However, this efficiency came at the cost of a higher average error. The results in Fig. 15 compare three cases: hard KBBC 1 using the trial function from Eq. (27), hard KBBC 2 using the trial function from Eq. (29), and hard periodic BC, discussed in Sec. V A 2, which incorporates the KBBC boundary condition as a loss term.

While the trial functions in Eqs. (27) and (29) successfully satisfied the KBBC, the results in Fig. 15 show that using these trial functions increased the overall average and standard deviation of the total error. This decline in accuracy may be due to the added complexity of the trial function. Additionally, trial functions can introduce unintended constraints that reduce accuracy, which can be difficult to identify but may be revealed by examining the trial function and error distribution.

The results of the u and w velocity components of the hard KBBC 1 case are shown in Fig. 16, and the corresponding results for the hard KBBC 2 case are presented in Fig. 17. Both cases capture the

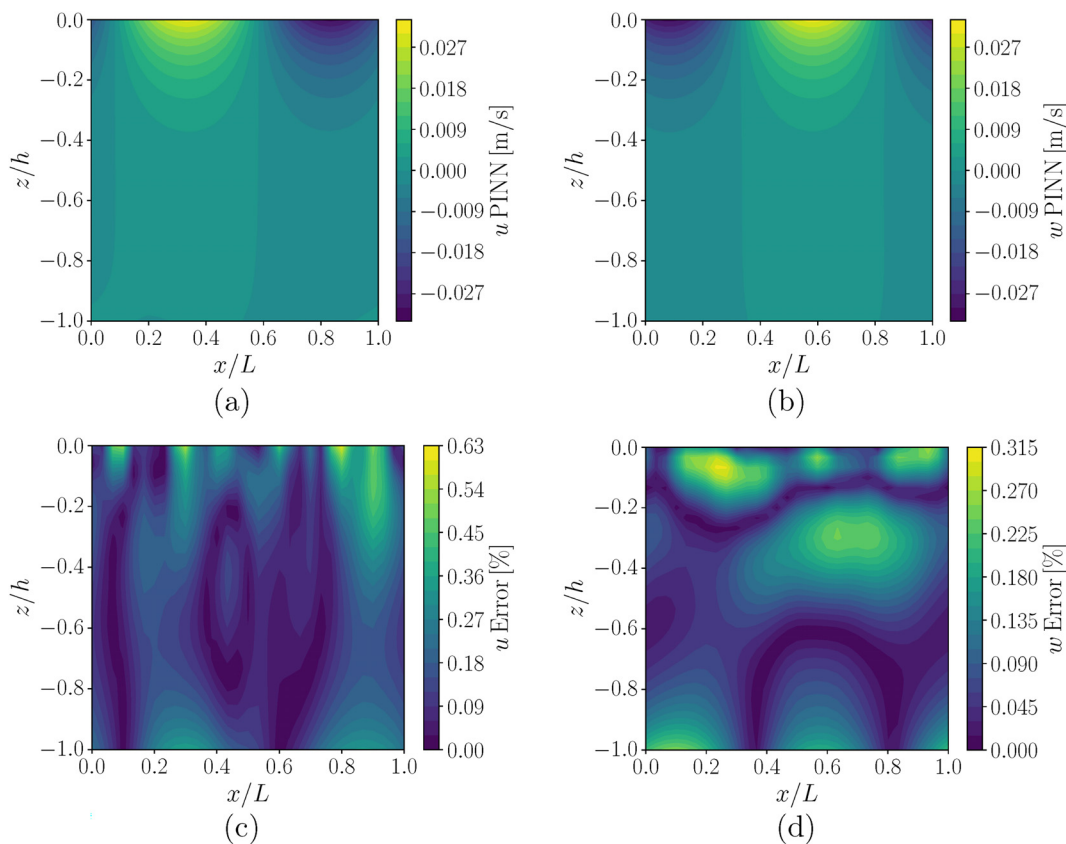


FIG. 14. Distributions of u (a) and w (b) and error distributions of u (c) and w (d) obtained by the Hard Periodic BC PINN at time $t = T/3$.

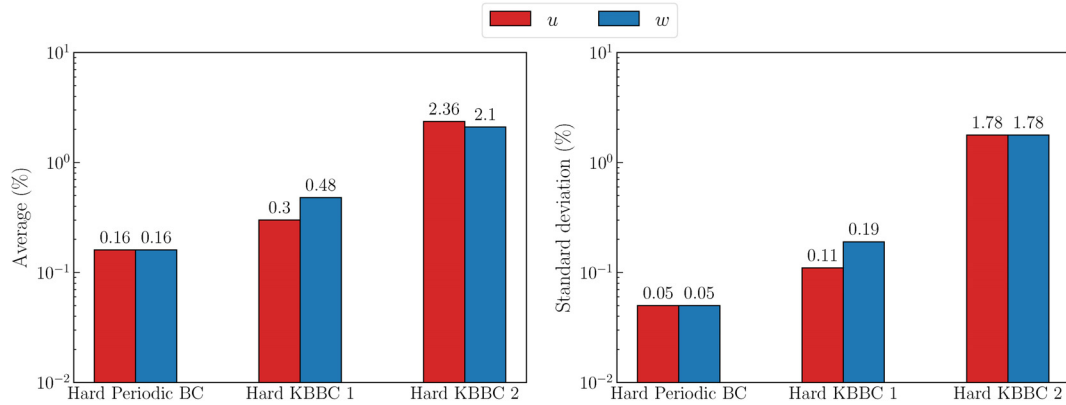


FIG. 15. Comparison of average and standard deviations of total errors for different methods of dealing with the KBBC.

overall distributions of u and w , aligning with the analytical results (Fig. 10). The hard KBBC 1 results reveal a noticeable error pattern in the bottom part of the domain for u , denoted by a red rectangle. Since this pattern was not observed in the error distribution of previous cases, it can likely be attributed to the application of the trial function defined in Eq. (27). Although this trial function satisfies the two main

requirements of a proper trial function, pointed out in Sec. VB, it probably introduces unintended constraints that are not immediately apparent, limiting the solution space in ways that are difficult to recognize.

To discover the cause of the error increase in the hard KBBC 2 case, the error distribution in the hard KBBC 2 case can be examined.

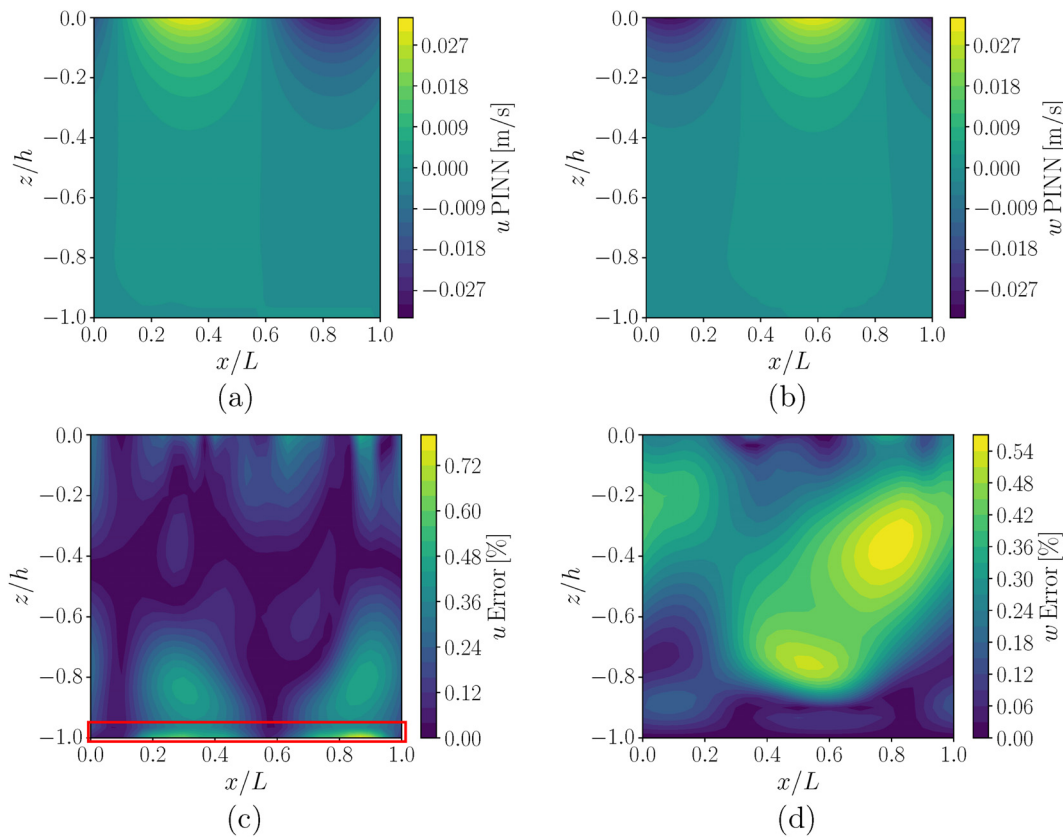


FIG. 16. Distributions of u (a) and w (b) and error distributions of u (c) and w (d) obtained by the hard KBBC 1 PINN at time $t = T/3$.

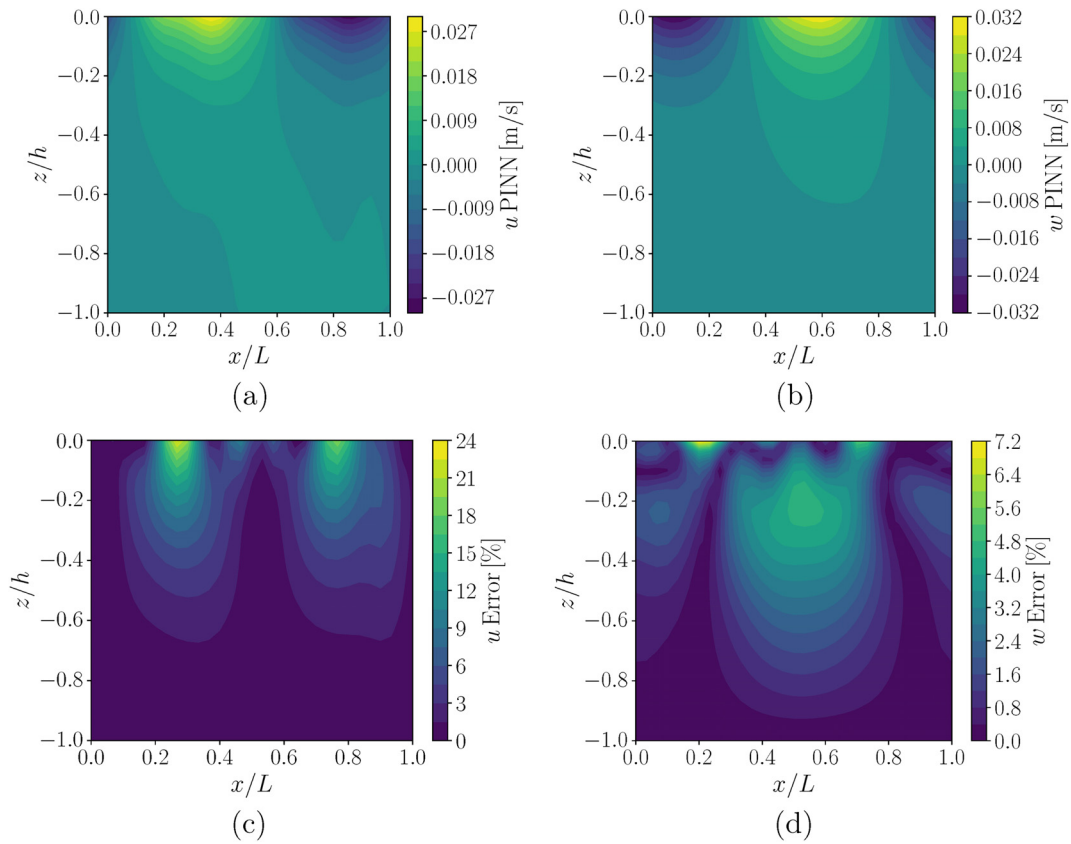


FIG. 17. Distributions of u (a) and w (b) and error distributions of u (c) and w (d) obtained by the hard KBBC 2 PINN at time $t = T/3$.

Figure 17 displays error distribution patterns similar to those in the previous cases, indicating minimal to no unintended constraints caused by the trial function. However, the trial function used in this case [Eq. (29)] fails to meet requirement 2 by setting $\phi_i(x, -h, t) = 0$ at $z = -h$, which can be a significant cause of the error increase in the hard KBBC 2 case. As the error distribution in Fig. 17 does not reveal any significant anomalies, it can be concluded that the error patterns may not fully reflect the issues caused by the trial function in the PINN.

Considering the process of deriving the trial functions defined in Eqs. (27) and (29) and their corresponding results, limitations in using the hard method based on trial functions can be articulated as follows:

1. Finding a suitable trial function for KBBC, which is a homogeneous Neumann boundary condition, is not always as straightforward as for Dirichlet BCs.³⁴
2. A practical trial function is not necessarily unique, and finding the best trial function demands efforts to find and compare different candidates.
3. Even if the trial function theoretically satisfies the two requirements mentioned in Sec. III C, there is no guarantee for convergence of the solution in practice.
4. The use of trial functions for imposing BCs gives a risk of introducing unwanted constraints for the PINN. The more complex

the function, the harder it becomes to detect these unwanted constraints, either in the trial function or in the error distribution.

The requirements, advantages, and limitations of specifying BCs via trial functions are summarized in Fig. 18.

C. Finding the unknown wave angular frequency

So far, the DFSBC has not been incorporated into the PINN. Instead, we have assumed the wave angular frequency, ω , as an input to the problem with the value from the analytical solution of DFSBC, namely, the dispersion relation [Eq. (31)]. However, ideally, ω should be obtained as a part of the solution. Thereby, in this section, ω is considered unknown, and the PINN is expected to find this parameter at the free surface by having another BC, which is DFSBC, introduced in Eq. (2). Finding ω by the PINN can be interpreted as solving the dispersion relation, via PINN.

For this purpose, the PINN, schematically shown in Fig. 8, has been used. A new term is added to the total loss function, dedicated to the DFSBC as

$$L_{\text{DFSBC}} = \sum_{i=1}^{n_x} \sum_{k=1}^{n_t} \left| \frac{\partial \phi(x_i, 0, t_k)}{\partial t} + g\eta(x_i, t_k) \right|^2. \quad (37)$$

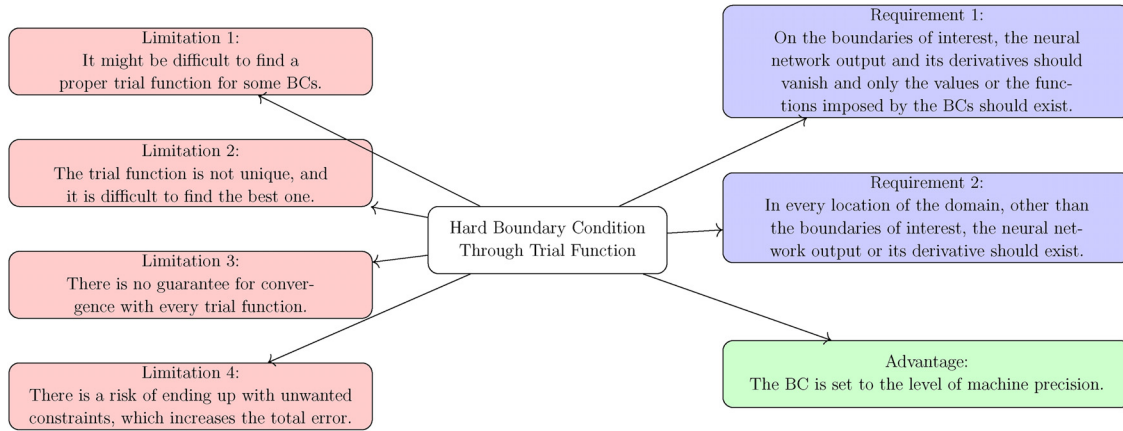


FIG. 18. Summary of limitations, requirements, and advantage of imposing boundary conditions through trial functions.

The trainable variables in this PINN include the weights and biases of the neural network, the phase angles of the neurons in the periodic layer, and ω in KFSBC and DFSBC. The number of Adam epochs is increased to 20 000 as the optimization task has become more complex by adding the trainable variable ω in the KFSBC and

DFSBC loss terms. As before, this is followed by L-BFGS. The wave number is $k = 1$ as in the previous cases.

The distributions of the u and w velocity components, as well as the error distributions of u and w , at time $t = T/3$ are illustrated in Fig. 19. A comparison of the results in Fig. 19 with the corresponding

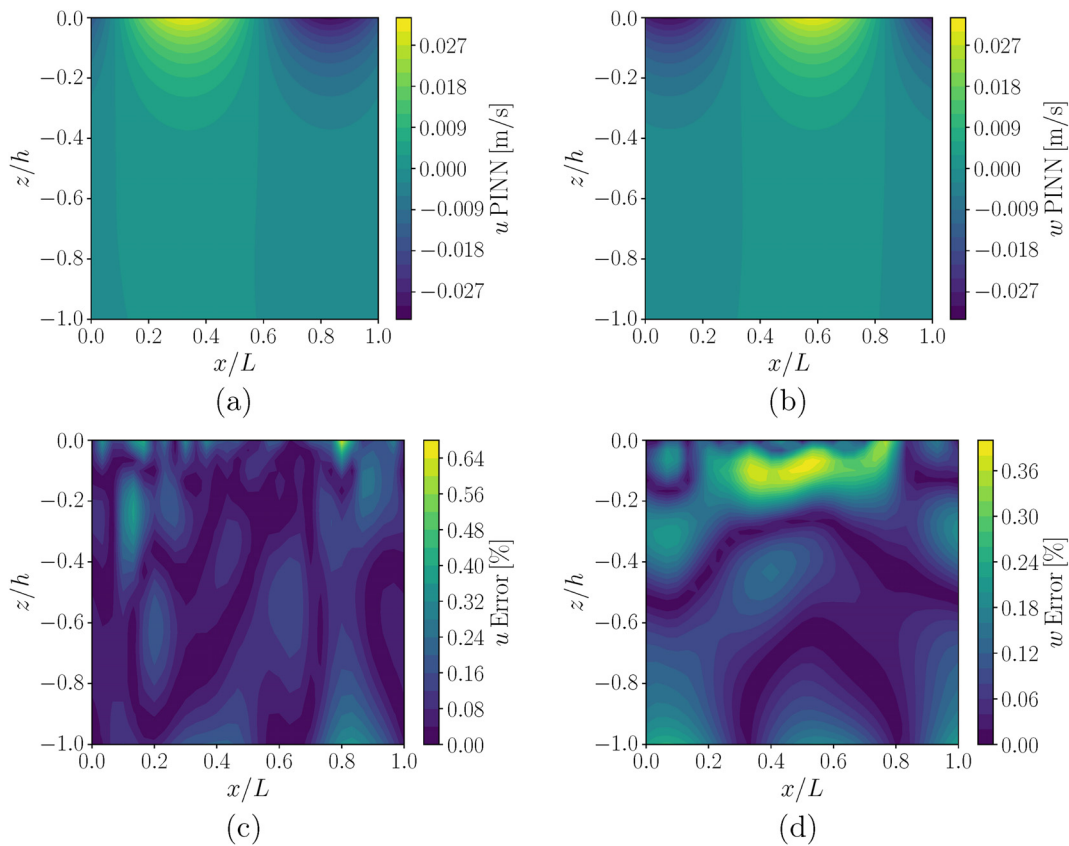


FIG. 19. Distributions of u (a) and w (b) and error distributions of u (c) and w (d) obtained by the PINN with unknown ω at time $t = T/3$.

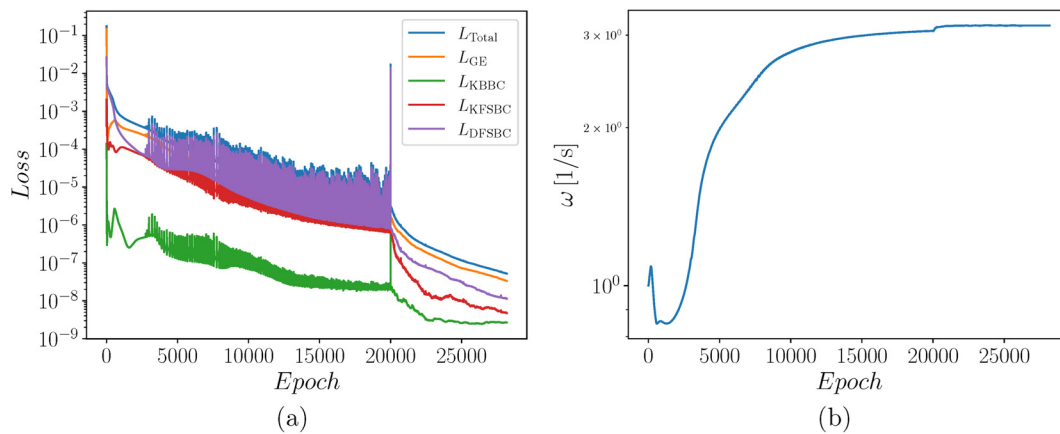


FIG. 20. Evolutions of loss values (a) and wave angular frequency (b) during the training.

analytical results in Fig. 10 indicates that the general distributions of u and w are captured relatively well. The error distributions of both u and w do not indicate any dominant pattern and look randomly dispersed, as was the case in most of the results Secs. IV, V A, and V B. The evolutions of the total loss, as well as the losses of the governing equation, KBBC, KFSBC, and DFSBC are shown in Fig. 20(a). The convergence of ω is shown in Fig. 20(b). During the L-BFGS optimization in Fig. 20, we observe a sudden increase in the values of the loss terms and ω during the final epochs. These jumps might be associated with the optimizer's attempt to refine the solution near local minima,

resulting in abrupt changes in the solution space. However, the values subsequently return to their previous range, consistent with the overall convergence trend.

The average total errors of the calculated u and w are 0.16% and 0.14% with standard deviations of 0.05% and 0.05%, respectively. The wave angular frequency, ω , has been found by the average error of 0.03% and a standard deviation of 0.03% across 100 simulations. The results suggest that the PINN has been successful in finding u and w under the free surface [Eq. (30)] and also solving the dispersion relation [Eq. (31)].

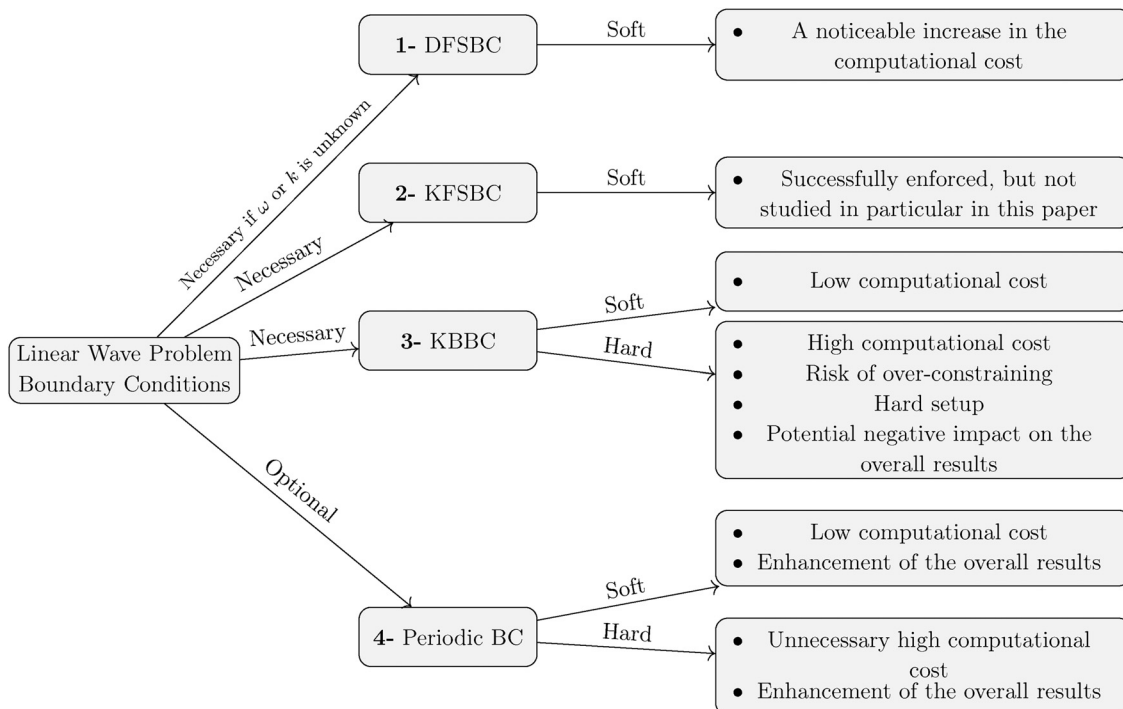


FIG. 21. Important features of the different enforcing methods of BCs of the linear wave theory.

VI. CONCLUSION

The current study demonstrates that the PINN framework is capable of calculating the wave-induced velocity field, based on the linear wave theory, both with known and unknown wave angular frequencies. The impacts of imposing the soft and hard approaches for different boundary conditions (BCs) of the problem are investigated and summarized in Fig. 21. In all cases, the soft approach cannot perfectly satisfy the constraints, and its effectiveness in enforcing them depends significantly on the optimization process. On the other hand, the hard approach meets the constraints with machine precision accuracy.

The results showed that the inclusion of the periodic BC (PBC) strongly contributes to the enhancement of the overall result, although there is no separate PBC in the problem definition. By enforcement of the PBC, the average error for u and w decreased from 2.31% and 0.27% to 0.10% and 0.10%, respectively. We found that the hard PBC provides a good trade-off between satisfying the periodicity of the results, enhancing overall results, and maintaining a reasonable computational cost. Two different trial functions were designed for enforcing KBBC. Unlike using PBCs, enforcing KBBC via trial functions presented several challenges, as are briefly mentioned in Fig. 21.

The PINN was also capable to converge to the solution with an average error of less than 0.16% and 0.14% for u and w , respectively, when the angular frequency, ω , of the free surface wave was considered an unknown. The value of ω was found in this case with an error of 0.03%.

Since linear wave theory forms the basis of wave simulations, the insights from this study will pave the way for modeling more complex wave theories within the PINN framework. Higher-order terms can be introduced to the linear wave theory via the Stokes' expansion approach. To further improve the results of this study, in addition to general measures such as increasing the number of collocation points and performing sensitivity analyses on various aspects of the PINN (e.g., the type of activation function and learning rate), adopting an adaptive distribution for the collocation points can potentially enhance the outcomes. This suggestion stems from the observation that the error distribution across the domain is often non-uniform. Therefore, strategically populating collocation points in different regions during training might be beneficial.

ACKNOWLEDGMENTS

The work was financed by the Department of Mechanics and Maritime Sciences at Chalmers University of Technology. It was carried out as a part of the "Swedish Centre for Sustainable Hydropower—SVC". SVC has been established by the Swedish Energy Agency, Energiforsk and Svenska kraftnät together with Luleå University of Technology, Uppsala University, KTH Royal Institute of Technology, Chalmers University of Technology, Karlstad University, Umeå University and Lund University.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Mohammad Sheikholeslami: Conceptualization (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Software

(equal); Validation (equal); Visualization (equal); Writing – original draft (equal). **Saeed Salehi:** Conceptualization (equal); Formal analysis (equal); Funding acquisition (equal); Investigation (equal); Methodology (equal); Project administration (equal); Resources (equal); Supervision (equal); Writing – review & editing (equal). **Wengang Mao:** Conceptualization (equal); Formal analysis (equal); Funding acquisition (equal); Methodology (equal); Project administration (equal); Resources (equal); Supervision (equal); Writing – review & editing (equal). **Arash Eslamdoost:** Conceptualization (equal); Formal analysis (equal); Funding acquisition (equal); Methodology (equal); Project administration (equal); Resources (equal); Supervision (equal); Writing – review & editing (equal). **Håkan Nilsson:** Conceptualization (equal); Formal analysis (equal); Funding acquisition (equal); Methodology (equal); Project administration (equal); Resources (equal); Supervision (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that supports the findings of this study are available within the article and its supplementary material.

REFERENCES

- ¹S. N. Elgeti and H. Sauerland, "Deforming fluid domains within the finite element method: Five mesh-based tracking methods in comparison," *Arch. Comput. Methods Eng.* **23**, 323–361 (2016).
- ²Y. H. Huang, Z. Xu, C. Qian, and L. Liu, "Solving free-surface problems for non-shallow water using boundary and initial conditions-free physics-informed neural network (bif-PINN)," *J. Comput. Phys.* **479**, 112003 (2023).
- ³E. M. Lewandowski, *The Dynamics of Marine Craft: Maneuvering and Seakeeping* (World Scientific, 2004).
- ⁴M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.* **378**, 686–707 (2019).
- ⁵H. Eivazi, M. Tahani, P. Schlatter, and R. Vinuesa, "Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations," *Phys. Fluids* **34**, 075117 (2022).
- ⁶H. Eivazi, Y. Wang, and R. Vinuesa, "Physics-informed deep-learning applications to experimental fluid mechanics," *Meas. Sci. Technol.* **35**, 075303 (2024).
- ⁷H. Baty and V. Vigon, "Modelling solar coronal magnetic fields with physics-informed neural networks," *Mon. Not. R. Astron. Soc.* **527**, 2575–2584 (2023).
- ⁸P.-H. Chiu, J. C. Wong, C. Ooi, M. H. Dao, and Y.-S. Ong, "CAN-PINN: A fast physics-informed neural network based on coupled-automatic-numerical differentiation method," *Comput. Methods Appl. Mech. Eng.* **395**, 114909 (2022).
- ⁹T. D. Ryck and S. Mishra, "Numerical analysis of physics-informed neural networks and related models in physics-informed machine learning," *arXiv:2402.10926 [math.NA]* (2024).
- ¹⁰D. Kim and J. Lee, "A review of physics informed neural networks for multi-scale analysis and inverse problems," *Multiscale Sci. Eng.* **6**, 1–11 (2024).
- ¹¹S. Cai, Z. Mao, Z. Wang, Y. Minglang, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," *Acta Mech. Sin.* **37**, 1727–1738 (2021).
- ¹²L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, "Physics-informed neural networks with hard constraints for inverse design," *SIAM J. Sci. Comput.* **43**, B1105–B1132 (2021).
- ¹³S. Dong and N. Ni, "A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks," *J. Comput. Phys.* **435**, 110242 (2021).
- ¹⁴L. Chen, B. Li, C. Luo, and X. Lei, "WaveNets: Physics-informed neural networks for full-field recovery of rotational flow beneath large-amplitude periodic water waves," *Eng. Comput.* **40**, 2819 (2024).

- ¹⁵A. D. Jagtap, D. Mitsotakis, and G. E. Karniadakis, “Deep learning of inverse water waves problems using multi-fidelity data: Application to Serre–Green–Naghdi equations,” *Ocean Eng.* **248**, 110775 (2022).
- ¹⁶T. T. Duong, K. H. Jung, G. N. Lee, and S. B. Suh, “Physics-informed neural network for the reconstruction of velocity and pressure of wave-in-deck loading from particle image velocimetry data,” *Appl. Ocean Res.* **153**, 104193 (2024).
- ¹⁷N. Wang, Q. Chen, and Z. Chen, “Reconstruction of nearshore wave fields based on physics-informed neural networks,” *Coastal Eng.* **176**, 104167 (2022).
- ¹⁸M. Sheikholeslami, S. Salehi, W. Mao, A. Eslamdoost, and H. Nilsson, “Physics-informed neural networks for modeling linear waves,” in ASME 43rd International Conference on Ocean, Offshore and Arctic Engineering, 2024.
- ¹⁹R. Leiteritz, M. Hurler, and D. Pflüger, “Learning free-surface flow with physics-informed neural networks,” in *20th IEEE International Conference on Machine Learning and Applications (ICMLA)* (IEEE, 2021), Vol. 378, pp. 1668–1673.
- ²⁰X. Qi, G. A. de Almeida, and S. Maldonado, “Physics-informed neural networks for solving flow problems modeled by the 2d shallow water equations without labeled data,” *J. Hydrol.* **636**, 131263 (2024).
- ²¹Y. Liu, X. Zhang, Q. Dong, G. Chen, and X. Li, “Phase-resolved wave prediction with linear wave theory and physics-informed neural networks,” *Appl. Energy* **355**, 121602 (2024).
- ²²A. K. Kushwaha, V. K. Gupta, H. Behera, and T.-W. Hsu, “Wave scattering by multiple floating flexible circular plates over a porous bed,” *Ocean Eng.* **314**, 119663 (2024).
- ²³A. K. Kushwaha, H. Behera, and V. K. Gupta, “A temporal study of wave scattering by multiple circular cylinders over a porous bed,” *Phys. Fluids* **37**, 037117 (2025).
- ²⁴R. Porter, S. Zheng, and H. Liang, “Scattering of surface waves by a vertical truncated structured cylinder,” *Proc. R. Soc. A* **478**(2258), 20210824 (2022).
- ²⁵D. Zhang, L. Guo, and G. E. Karniadakis, “Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks,” *arXiv:1905.01205* (2019).
- ²⁶I. E. Lagaris, A. Likas, and D. I. Fotiadis, “Artificial neural networks for solving ordinary and partial differential equations,” *IEEE Trans. Neural Networks* **9**, 987–1000 (1998).
- ²⁷P. L. Lagari, L. H. Tsoukalas, S. Safarkhani, and I. E. Lagaris, “Systematic construction of neural forms for solving partial differential equations inside rectangular domains, subject to initial, boundary and interface conditions,” *Int. J. Artif. Intell. Tools* **29**, 2050009 (2020).
- ²⁸M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations* (Springer, 2024).
- ²⁹A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, “Automatic differentiation in machine learning: A survey,” *arXiv:1502.05767* [cs.SC] (2018).
- ³⁰S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv:1609.04747* [cs.LG] (2017).
- ³¹D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv:1412.6980* [cs.LG] (2017).
- ³²D. C. Liu and J. Nocedal, “On the limited memory BFGS method for large scale optimization,” *Mathematical Programming* **45**, 503–528 (1989).
- ³³O. Faltinsen, *Sea Loads on Ships and Offshore Structures* (Cambridge University Press, 1993), Vol. 1.
- ³⁴H. Baty, “A hands-on introduction to physics-informed neural networks for solving partial differential equations with benchmark tests taken from astrophysics and plasma physics,” *arXiv:2403.00599* [physics.comp-ph] (2024).
- ³⁵H. Baty and L. Baty, “Solving differential equations using physics informed deep learning: A hand-on tutorial with benchmark tests,” *arXiv:2302.12260* [cs.LG] (2023).
- ³⁶L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, “DeepXDE: A deep learning library for solving differential equations,” *SIAM Rev.* **63**, 208–228 (2021).
- ³⁷I. Chatterjee, *Machine Learning and its Application: A Quick Guide for Beginners* (Bentham Science Publishers, 2021).
- ³⁸S. Berrone, C. Canuto, M. Pintore, and N. Sukumar, “Enforcing Dirichlet boundary conditions in physics-informed neural networks and variational physics-informed neural networks,” *Heliyon* **9**, e18820 (2023).
- ³⁹X. Jin, S. Cai, H. Li, and G. E. Karniadakis, “NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations,” *J. Comput. Phys.* **426**, 109951 (2021).
- ⁴⁰A. D. Jagtap and G. E. Karniadakis, “Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations,” *Commun. Comput. Phys.* **28**, 2002–2041 (2020).
- ⁴¹P. Roy and S. Castonguay, “Exact enforcement of temporal continuity in sequential physics-informed neural networks,” *arXiv:2403.03223* [cs.LG] (2024).
- ⁴²J. D. Fehribach, *Sequences and Series in Calculus*, De Gruyter Textbook (De Gruyter, 2023).