Exploring Waves Propagation with a Newly Developed High-Performance Code in OpenFoam using Free-Surface Capturing Approach

A. Mehmood[1], D. I. Graha[2], K. Langfeld[3], D. M. Greaves[4], K. Akhtar[5], A. Naveed[6]

[1, 5, 6]*Department of Mechanical Engineering, University of Engineering and Technology, Peshawar, Pakistan.*
[2, 3, 4]*University of Plymouth, Drake Circus, Plymouth PL4 8AA, United Kingdom*

[1]arshadmehmood@uetpeshawar.edu.pk

Abstract- To address the demands of contemporary research and industry, it is crucial to reduce computational costs and rely on open-source codes to simulate real-world wave propagation phenomena. As a response, we have developed an efficient solver for modeling free surfaces within the OpenFOAM computational fluid dynamics (CFD) software package. Our approach involves employing the finite volume method for discretization to solve the Laplacian of the velocity potential. Moreover, we have devised the necessary kinematic and dynamic boundary conditions to depict fluid behavior at the computational domain's boundaries and the free surface's behavior. The convergence analysis demonstrated the expected first-order accuracy of the solver as the error decreased with a reduction in grid size. Time discretization study showed good temporal convergence as decreasing the time step size resulted in lower error. Wave period comparison validated the numerical solution by comparing it with 2nd-order Stokes theory, and a close agreement was observed between the simulated and analytical wave periods. We also validated our model by comparing them with experimental data from previous studies. The comparisons showed good agreement between the numerical simulations and experimental measurements, confirming the reliability of the developed numerical model. It is pertinent to note that our solver and boundary conditions are novel, exclusive to our research and were not present in the standard OpenFOAM distribution.

Keywords- CFD; OpenFOAM; Numerical modelling; Potential flow

I. INTRODUCTION

Waves propagation phenomena and their interaction with structures have received considerable attention in marine and coastal engineering [1-3]. There are three main methods for analyzing waves: analytical, experimental, and numerical. Analytical analysis involves using mathematical equations to determine how waves behave. Although this approach can provide insight into the underlying physics of waves, its results are often restricted to idealized conditions and may not consider real-world complexities. On the other hand, experimental analysis involves measuring actual wave behavior in the real world. While this approach can provide a wealth of data, it is often limited to specific wave conditions and conducting experiments for various scenarios can result in high computational cost[4-7]. Finally, numerical analysis involves using computer simulations to model wave behavior. This approach is capable of accounting for complex real-world conditions and providing highly accurate predictions. However, it requires specialized software and can be computationally intensive. Despite this, numerical solutions are often more practical in terms of cost and complexity [8-10]. In fact, with the availability of powerful computational resources today, numerical modeling has become an increasingly popular tool for researchers to replicate experimental observations[3, 9, 11-15].

The natural evolution of waves depends on various factors such as their amplitude, wave period, and water depth [16-19]. Typically, these waves are generated offshore and propagate over long distances. In most cases, the effects of fluid viscosity are negligible throughout the propagation of waves. Therefore, potential flow theory is used to model free surface waves, assuming that the fluid flow is inviscid and incompressible, and the fluid is irrotational [20]. This theory assumes that pressure and gravitational forces dominate the motion of fluid particles. Potential flow theory is considered computationally efficient, as the method relies on solving Laplace's equation to find the velocity potential, which is a scalar function that describes

the fluid flow and is the only unknown variable in the model. The current research work aims to simulate nonlinear water waves with minimum numerical cost and to make the developed code available to the research community as an open-source tool.

The developed solver is implemented in the OpenFOAM-Extend environment, which is an open-source computational fluid dynamics (CFD) solver. OpenFOAM, which stands for Open Field Operation and Manipulation, is a software library that offers a range of numerical methods and solvers for simulating various physical phenomena, including fluid flow and heat transfer. OpenFOAM is widely used for CFD simulations and other multi-physics simulations due to its open-source nature and versatility [21-24]. OpenFOAM is a versatile software library that incorporates various numerical methods and solvers for simulating physical phenomena. It supports both grid-based and particle-based methods, such as the finite volume method, the finite element method, and the lattice-Boltzmann method. The software also provides a diverse range of pre-built solvers for common simulation problems, including turbulent flow, multiphase flow, and free surface flow. The software's flexibility is one of its main advantages, allowing it to be applied to a wide range of geometries, including complex and irregular shapes. Implementing boundary conditions is also straightforward with OpenFOAM, and it can be easily coupled with other numerical methods, such as boundary element methods. Another advantage of OpenFOAM is its open-source nature, making it freely available to users, and the source code can be modified and improved upon by the community. To learn more details about how OpenFOAM integrates space and time, it is suggested to read reference [2]. In various fields, such as aerospace, automotive, energy, and environmental engineering, OpenFOAM has become a popular software library for simulating fluid flow problems [21-23, 25, 11]. Hydrodynamic groups have also been actively using it for coastal-related applications. Researchers have developed different solvers for simulating free surface waves, including waves2Foam [11] and IHFOAM [1]. Waves2Foam actively generates waves and absorbs them using wave relaxation zones, which extends the computational domain over a few wavelengths, increasing computational expenses. IHFOAM, on the other hand, generates and absorbs waves actively, thus reducing computational costs. However, both models use the built-in Volume Averaged Reynolds-Averaged Navier-Stokes equations “interFoam” with a few modifications and new boundary conditions to generate and absorb waves. Both solvers simulate regular, irregular, and random waves, wave-current interactions, and wave-breaking phenomena. When dealing with big areas of space where waves don't become steeper or break, using these tools will lead to expensive computational requirements. The current article describes a solver developed from scratch using the OpenFOAM functions that will create and propagate waves at a lower computational expense until the point where the wave starts to overturn, which is a key element currently lacking in OpenFOAM for coastal engineering studies.

To model real-world problems involving fluid flow, it is necessary to simulate the full Navier-Stokes equations for both air and water above and below the free surface, including the effects of aeration during wave impact on structures, wind effects on waves, and the hydro-elastic response of compliant structures. A domain decomposition approach is necessary to minimize computational costs while still capturing the physics of wave propagation and interactions with structures. This means that flow solvers with varying degrees of physics and computational overheads are required. For wave generation at the boundary, a scalar nonlinear full potential method is suitable, while a multi-fluid Navier-Stokes solver can resolve detailed flow physics in both air and water regions as waves approach, steepen, and break over a fixed or floating wave converter. Close to the wave converter where impacting waves may entrain air into the water and/or enclose an air pocket, a compressible Navier-Stokes solver may be required. The developed solver in this paper is implemented in OpenFOAM due to its flexible framework, which allows users to customize existing solvers or develop new ones to meet their specific requirements. However, the developed solver cannot capture wave breaking or compressibility effects, which are important aspects of wave dynamics. To address this, the simulation results of the existing solver will be coupled with existing incompressible and compressible Navier-Stokes solvers to capture these phenomena. A new boundary condition will be created to facilitate this coupling.

OpenFOAM employs a method of discretizing finite volumes on unstructured meshes that are made up of various types of convex polyhedral shapes. The finite volume method is a mathematical technique employed to solve partial differential equations that govern the motion of fluids. These equations include the Navier-Stokes equations and the continuity equation. It is a grid-based method that partitions the domain of interest into a finite number of small control volumes, with the equations solved at the center of each control volume. The finite volume method is extensively used in computational fluid dynamics (CFD) simulations, and it is particularly suitable for modeling complex flow phenomena such as turbulent flows, multiphase flows, and free surface flows [9, 26, 2, 27]. Moreover, it is a popular method utilized in water wave modeling. The approach involves dividing the domain into a grid of discrete control volumes and then utilizing the

equations of fluid dynamics to calculate the fluid properties, like velocity and pressure, in each control volume. The method then updates the fluid's properties at each control volume, creating a time-dependent simulation of the fluid's behavior. The finite volume method is versatile and can be applied to a broad range of geometries, including irregular and complex shapes [11, 25, 27, 28]. Furthermore, it permits the easy implementation of boundary conditions and can be coupled easily with other numerical techniques, such as boundary element methods. However, the method is computationally intensive, and the results' accuracy may be grid resolution-dependent.

The modelling of water wave propagation is a complex task, owing to various factors such as the nonlinear behavior of water waves and their interactions with boundaries such as shores, boats, and other obstacles, which can cause unpredictable wave interactions. Additionally, several variables, such as wave height, wave period, wave direction, and water depth, need to be taken into account. The physics of wave propagation is governed by complex hydrodynamic equations, which include wave dispersion and nonlinear wave-wave interactions. To address these complexities, different models have been developed and classified into two main categories: "surface capturing" and "surface tracking". The surface capturing approach uses the so-called "particle-based" method, which captures the free surface of the water by tracking the motion of individual water particles. The motion of these particles is then used to calculate the water surface and wave properties such as wave height, velocity, and direction. One of the most well-known surface capturing methods is the Smoothed Particle Hydrodynamics (SPH) method [14], which uses a set of Lagrangian particles to represent the water surface and a set of equations to calculate the forces acting on these particles. These equations include the Navier-Stokes equations, which describe the motion of fluid, and the continuity equation, which describes the conservation of mass. The surface-capturing approach allows for the simulation of complex wave behavior, such as the interaction of waves with obstacles, and the breaking of waves. Using surface capturing technique, researchers including [27, 29, 30] used Navier-Stokes equations for wave modeling that takes into account the effects of viscosity, compressibility, and turbulence in the fluid. The Navier-Stokes equations describe the conservation of mass, momentum, and energy of a fluid and are much more complex than the potential flow equations.

On the other hand, researchers including [31, 16, 32, 33] used the potential flow wave modeling approach which assumes the fluid to be inviscid, incompressible, and irrotational. Using the potential flow assumptions, numerical methods, such as finite element method [31, 16, 32, 34] or boundary element method [35], have been used to model the complex wave phenomena. These methods involve discretizing the flow field into a grid of points and using numerical algorithms to solve fluid motion equations. These models are computationally efficient, especially when compared to more complex models that include viscous effects (e.g., Navier-Stokes models). This efficiency allows for quick simulations of large-scale wave phenomena, which is especially useful in engineering and design applications. The surface tracking method is a technique that uses a "grid-based" approach to capture the free surface of water by solving the equations of motion on a fixed grid. The movement of the water surface is then determined by solving the equations at each point on the grid. A well-known example of this approach is the Volume of Fluid (VOF) method [23, 1, 11, 36], which utilizes a scalar field to represent the water surface and track its evolution over time. In this method, the Navier-Stokes equations are used to calculate the velocity and pressure fields, while the continuity equation is used to determine the evolution of the water surface. By employing this technique, complex wave behavior, such as the interaction of waves with obstacles and the breaking of waves, can be simulated. However, when trying to solve for two fluids within the computational areas along with an extra scalar transport equation, it leads to expensive computational requirements, and the scalar field is also prone to numerical diffusion.

The Mixed Eulerian and Lagrangian method (MEL) is a numerical technique used in computational fluid dynamics (CFD) to solve fluid flow problems [37, 38]. It combines aspects of both Eulerian and Lagrangian methods to capture the advantages of each approach. The Mixed Eulerian and Lagrangian method (MEL) technique combines the advantageous aspects of both Eulerian and Lagrangian approaches to solve fluid flow issues. In CFD, the Eulerian method is often employed to solve the governing equations of fluid flow on a fixed grid [39]. Although this approach is effective for examining large-scale flow phenomena, it may face challenges in accurately representing transient or highly localized events. Conversely, the Lagrangian method tracks individual fluid particles as they traverse through the flow field. This methodology is particularly beneficial for simulating particle-laden flows or highly unsteady flows since it follows the motion of discrete entities. However, it can be computationally expensive and less efficient at capturing global flow characteristics [38]. To overcome their individual limitation, we merge the strengths of both Eulerian and Lagrangian methods. We divide the computational domain into two regions: one where the Eulerian method solves the equations of fluid motion on a fixed grid (i.e., Laplacian equations), and another where the

Lagrangian method tracks the motion of individual fluid particles on the free surface.

The aim of the current solver is to improve the efficiency of numerical wave simulations in marine environments using OpenFOAM. The solver utilizes the potential flow theory approach and successfully implements wave generation and absorption boundary conditions. The solver tracks the deformable free surface by utilizing an OpenFOAM solver that is typically used for simulating incompressible fluid flows with dynamic meshes. At each time step, the internal mesh adapts to accommodate the deformation of the free surface. Unlike the OpenFOAM standard distribution, this solver solves Laplace's equation using the finite volume method (FVM) and incorporates necessary kinematic and dynamic boundary conditions for free surface waves. Additionally, the solver includes wave generation and absorption boundary conditions that were developed specifically for this purpose.

II. MATHEMATICAL FORMULATIONS

Considering an irrotational flow, which is also referred to as a potential flow, the flow's behaviour can be determined by solving a 2D Laplace's equation. Mathematically, it can be expressed as:

$$\nabla^2\varphi(x,y,t) = 0 \tag{1}$$

Here, φ represents the velocity potential and ∇^2 is the Laplacian operator, which measures the rate of change of the velocity potential in space. Solving Laplace's equation as in (1) allows us to determine the velocity of the fluid at any point in space, which can be calculated using the gradient of the velocity potential as $v = \nabla\varphi$. Once the velocity field is known, the pressure distribution can be found using Bernoulli's equation.

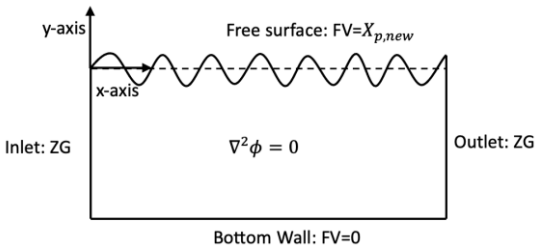


Figure 1: Setup of the domain along with the boundary conditions, FV, which represents fixed value, and ZG, which represents zero gradient.

In our Cartesian coordinate system, the upper-left corner of the undisturbed domain serves as the origin, as shown in Figure 1. The undisturbed free surface is represented by a dotted line in Figure 1**,** while the wave-like shape indicates the presence of a generated wave. To solve Eqn. (1), we specify Neumann boundary conditions at the Inlet, Outlet, and Bottom Wall boundaries. At the upper boundary (i.e., the free surface), we apply a dynamic boundary condition expressed as:

$$\frac{\partial\varphi}{\partial t} = -g\zeta - \frac{1}{2}\nabla\varphi.\nabla\varphi \tag{2}$$

where g is the acceleration due to gravity, ζ is the wave surface elevation, t is time and ∇ is the gradient operator. We apply a kinematic boundary condition on the free surface to make sure that it responds to alterations in the velocity field and to maintain fluid volume conservation. The kinematic boundary condition requires that the normal component of the fluid's velocity vector at the free surface matches the normal component of the free surface's velocity. This can be expressed mathematically as:

$$\frac{\partial\zeta}{\partial t} = \frac{\partial\varphi}{\partial y} - \frac{\partial\varphi}{\partial x}\frac{\partial\zeta}{\partial x} \tag{3}$$

Equation (3) for the kinematic boundary condition can also be phrased with reference to the volume of fluid [29] within a system, as follows:

$$\frac{d\zeta}{dt} = \frac{v.n}{n_y} \tag{4}$$

In this equation, v represents the velocity vector, n represents the unit normal vector, and n_y represents the unit normal vector in the y-direction.

However, the velocity potential and computed velocities are known at cell centres, while the flux is defined at the face centres of the control volumes. To appropriately update the mesh, we interpolate the flux $v.n$ from face centres to cell vertices. The mesh is then updated, and the solution is re-computed using the updated mesh. Once the mesh is updated, we solve the dynamic boundary condition as in (2) to calculate the velocity potential on the free surface for the subsequent time step. In the implemented technique, the mesh is allowed to deform without changing its topology. It should be noted that the boundaries attached to the free surface boundary, as shown in Figure 1, are the inlet and outlet, which should have unrestricted displacement, so a zeroGradient boundary conditions are applied. The bottom of the fluid domain should remain fixed, and thus, a fixed value boundary condition "fixedValue=0" is imposed. Once the boundary conditions for cell vertices (i.e. point displacement) are specified, the Laplacian solver is used for mesh motion by solving the Laplace equation for the displacement of each mesh point from its initial position.

The displacement of the vertices of the free surface is determined by Equation (4). In the OpenFOAM implementation of the Finite Volume (FV) methodology, mesh values are defined at vertices, which represent the corners of each cell in the mesh. However, the velocity potential and computed velocities are known at cell centers, while the flux is

defined at the face centers of the control volumes. To properly update the mesh, we perform an interpolation of the flux, represented by $v.n$. This allows for an appropriate update of the mesh. Once the mesh is updated, the solution is re-computed using the modified mesh. To calculate the velocity potential on the free surface for the next time step, we solve the dynamic boundary condition as described in equation (2). In this technique, the mesh is allowed to deform without altering its topology, ensuring consistency in the connectivity of the mesh elements. It's worth noting that the boundaries attached to the free surface boundary, as depicted in Figure 1, are the inlet and outlet boundaries. These boundaries should have unrestricted displacement, and therefore, zeroGradient boundary conditions are applied to them. On the other hand, the bottom of the fluid domain is required to remain fixed, and thus, a fixed value boundary condition, represented by "fixedValue=0," is imposed. Once the boundary conditions for cell vertices (i.e., point displacement) are specified, the Laplacian solver is employed for mesh motion. This involves solving the Laplace equation for the displacement of each mesh point from its initial position. By solving this equation, the mesh points are appropriately updated, accounting for the displacement of the free surface vertices and preserving the overall topology of the mesh.

The variables pertaining to fluid flow are computed at the centers of individual cells and then interpolated to cell vertices, which results in a saw-tooth free surface. To enhance the precision of the solution, a 5-point smoother is used to adjust the values of the function at the nodes of the grid. The formula for the 5-point smoother is $f_i = (-f_{i-2} + 4f_{i-1} + 10f_i + 4f_{i+1} - f_{i+2})/16$, where f is the value of the function at the required node and f_{i-2}, f_{i-1}, f_{i+1}, and f_{i+2} are the values of the function at the four adjacent nodes [24] [25]. It should be noted that when using the 5-point smoother, certain modifications are required to account for the fact that the values at certain adjacent nodes are not available, particularly at the first and last nodes of the grid. For instance, in the case of progressive waves, the values at the first node are computed using a 3rd-order approximation as:

$$f_{i-1} = f_{i+2} - 3f_{i+1} + 3f_i \quad (5)$$

$$f_{i-2} = 3f_{i+2} - 8f_{i+1} + 6f_i \quad (6)$$

Since, f_{i-1} and f_{i-2} are the values of the function out of the domain, and ,f_i f_{i+1}, and f_{i+2} are the values of the function at the first node, the adjacent node, and the adjacent-to-adjacent node, respectively.

2.1. Moving boundary modeling

In the OpenFOAM-Extend environment, we created a new boundary condition class to implement new boundaries. We defined input parameters for the boundary conditions, including coefficients, reference values, and other necessary constants, based on our model. We developed and implemented the following boundary conditions as:

1. *inlet boundary condition:* The following is a description of the inlet boundary condition that we implemented in the OpenFOAM environment. To specify this condition, we use the equation:

$$\frac{\partial \varphi}{\partial x} = u = f(y,t) \quad (7)$$

where $f(y,t)$ is any known function. However, for standing wave test cases, we set $f(y,t) = 0$, which corresponds to a zeroGradient boundary condition. This assumes that the value at the inlet is equal to the neighboring cell value. For progressive wave test cases, we impose a velocity component u in the x-direction of the known wave theory as follows:

$$\frac{\partial \varphi}{\partial x} = u \quad (8)$$

The applied velocity u can be from either the Stokes wave theory or an experimental wave maker. The Stokes wave theory predicts that the wave profile is sinusoidal, and the wave height and wavelength are related to the frequency and water depth. Additionally, we use a ramp function to gradually increase the wave height of the generated waves to a desired level over a period of time. Currently, the code support sinusoidal and linear ramp functions.

2. *freeSurface boundary condition:* We enforce both the kinematic boundary condition described as in (4) and the dynamic boundary condition as in (2) on this surface.
3. *bottom wall boundary condition:* We impose the no-slip condition at this surface which assumes that the fluid is stationary at the bottom wall.
4. *outlet boundary condition:* In order to avoid wave reflections back into the computational domain, which can lead to interference and inaccuracies in the simulation, we employ an absorbing boundary condition. This condition facilitates the smooth exit of waves from the domain without reflection, effectively absorbing them as they leave the modeled region. For this purpose, we implemented the Sommerfeld condition.

$$\frac{\partial \varphi}{\partial t} + v_{phase}\frac{\partial \varphi}{\partial n} = 0 \quad (9)$$

In this equation, n represents the normal vector pointing outward from the outlet boundary's surface, and v_{phase} refers to the wave's phase velocity. The parameter v_{phase} is determined by the linear harmonic wave velocity equation $v_{phase} = \sqrt{g \times \tanh(kH)/k}$, where H signifies the water depth and *k* represents the wave number. The newly developed boundary conditions are designed to be

modular and require only the input of wave amplitude, wave period, and the time required for full development. The wave length and wave number are then computed using wave dispersion relations. As a result, switching to different boundary conditions only requires changing the expression and the necessary variables for initialization. To validate these boundary conditions, we compared simulation results with analytical and experimental data, which are presented in the Results and Discussions section.

2.2. Sequence of the Solution Procedure of the solver

The steps for implementing the potential flow solver in OpenFOAMfrom t^n to t^{n+1} are as follows.

1. Create the grid.
2. Apply the necessary boundary conditions on the computational domain.
3. Calculate the velocity potential by solving Laplace's equation.
4. Compute velocities at the centers of cells and fluxes at the centers of faces.
5. Find the updated shape of the surface of the fluid by solving (4) that describes the behavior of the fluid at the boundary.
6. Based on the free surface elevation computed in the previous step (step 5), update the grid.
7. Solve (2) that describes the dynamic boundary condition at the free surface on the updated grid. This calculation provides the velocity potential on the boundary for the next time step. Adjust the boundary conditions for the remaining three boundaries based on the updated information.
8. To progress the solution in time, repeat the aforementioned steps (steps 4-7) for each time step.

Moreover, to run a case(example) using the current solver in OpenFOAM, one will need to follow these steps:

1. *Set up the geometry:* The first step is to create the 3D geometry of the domain that includes the fluid and the free surface. This can be done using a CAD software or OpenFOAM's built-in meshing tools
2. *Mesh the domain:* The next step is to generate a suitable mesh that will be able to resolve the wave features. This can be done using OpenFOAM's meshing tools, such as snappyHexMesh or Gmsh.
3. *Set up the case:* The next step is to set up the case by defining the solver which is named as "potDyMFoam", time and spatial discretization schemes, initial and boundary conditions.
4. *Run the simulation:* The simulation is run by using the command in a terminal window ``potDyMFoam". The solver solves the Laplace equation for the velocity potential, and calculates the velocity fields by the gradient of velocity potential and then pressure using the Bernoulli's equations.
5. *Post-processing:* Once the simulation is complete, the results can be post-processed to visualize the wave height, velocity field, and pressure distribution. OpenFOAM provides several post-processing tools, such as ParaView, foamToVTK, and foamToSurface.

III. RESULTS AND DISCUSSION

The primary objective of developing the current solver within the OpenFOAM environment was to leverage the built-in capabilities of OpenFOAM and enable seamless coupling with existing incompressible and compressible Navier-Stokes solvers, thus encompassing a wide range of wave conditions. In order to demonstrate the effectiveness of the newly created solver and its associated boundary conditions, several simulations were conducted. These simulations were performed on a workstation equipped with an Intel Core i7-4790 CPU, 16 GB of memory, and a 3.6 GHz Power 8 processor. By utilizing the computational resources and capabilities of this setup, we were able to validate the functionality and performance of the solver and its boundary conditions. We validated the accuracy of our developed numerical model by comparing its results with theoretical solutions, published numerical simulation results, and experimental data. Our goal was to establish the numerical model's reliability and accuracy. The validation process consisted of several steps, including convergence analysis in space and time, wave period comparison, comparison of the simulation results for standing water waves, progressive waves, and finally, comparison with experimental data. For all our simulations, we employ the Crank-Nicolson time integration method to handle the wave elevation equations, while we utilize the Euler time integration method for the remaining equations.

3.1. Convergence in Space and time

To ensure independence from grid size, we conducted a simulation of a sinusoidal wave. The simulations were conducted with a mean water depth of H = 0.8 m, and the wave characteristics were set as follows: amplitude a = 0.01 m and wavelength λ = 1.0 m. The simulation was carried out using several regular grids, as described in Table-1, using the function $\zeta = a\sin(kx)$ and a dimensional time step of Δt = 0.005 s. We plotted the error on the y-axis and the grid size on the x-axis, both in a logarithmic scale, as shown in Figure 2. We calculated the error as the difference between the numerical and analytical solutions. As indicated by the plot, the error decreases with a reduction in the grid size, demonstrating the expected first-order accuracy.

Table 1: The number of different meshes used.

Cases	Grid size
Grid-1	13 x 11 x 1
Grid-2	20 x 17 x 1
Grid-3	33 x 26 x 1
Grid-4	49 x 39 x 1
Grid-5	75 x 59 x 1
Grid-6	113 x 89 x 1
Grid-7	169 x 134 x 1
Grid-8	211 x 167 x 1
Grid-9	253 x 201 x 1

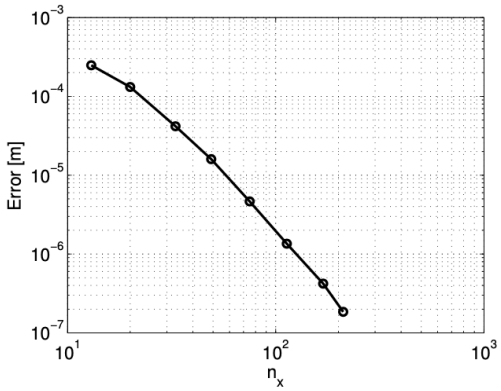


Figure 2: The plot shows the error as a function of the number of grid points in the x-direction.

We conducted a time discretization study using Grid-4, which comprises 49 points in the x-direction and 39 points in the y-direction. The simulation utilized a sinusoidal wave with a mean water depth of H = 0.8 m, and the wave characteristics were set as follows: amplitude a = 0.01 m and wavelength λ = 1.0 m. The simulation was carried out for different time step sizes, namely Δt = 0.025, Δt = 0.0125, Δt = 0.00625, and Δt = 0.003125. The L1-error was calculated for each time step size, and the results were plotted against the time steps in Figure 3. The graph shows that decreasing the time step size results in lower error, indicating that the simulation achieves good temporal convergence.

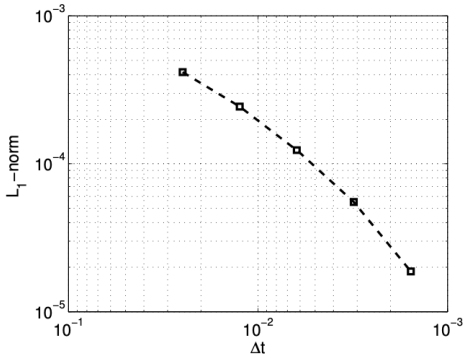


Figure 3: Change in error estimate with time step.

3.2. Wave Period Comparison

We also made wave period comparison for the developed numerical solution. We obtain the numerical solution for different kinds of waves covering a range of water depths from shallow to deep. We calculated the wave period of the numerical solution by identifying the time interval between two consecutive wave peaks. We then compared the results we obtained with the wave period calculated using 2nd-order Stokes theory. We examined two distinct wave amplitudes, namely a = 0.005 m and 0.01 m, while keeping the wave length constant at λ = 1.0 m. We gradually altered the mean water depth and recorded the wave periods from the resulting simulations. Figure 4 shows a comparison between the wave periods obtained and the analytical values, plotted against the mean water depth. The depicted plot indicates a close agreement between the wave periods obtained from the current numerical scheme and those predicted by the 2nd-order Stokes theory.

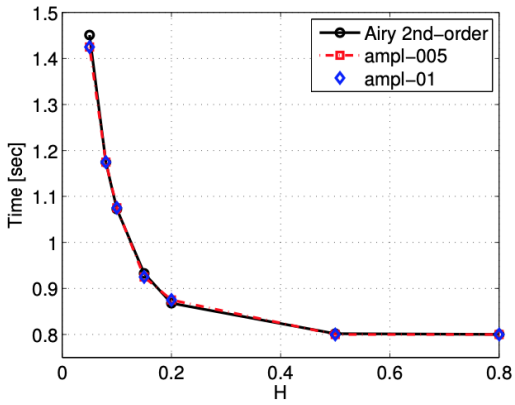


Figure 4: The plot shows how the wave period varies with the mean water depth, with the wavelength used as a normalizing factor.

3.3. Standing Waves

Standing water waves are a common occurrence in bodies of water such as lakes, rivers, and oceans. These waves can be generated by several factors, such as wind, tides, and the interaction of waves with obstacles like breakwaters, piers, and other structures. Accurate simulation of standing water waves is crucial for predicting the behavior of more complex phenomena and for designing structures, bridges, and ships. It can also lead to improved models for a range of physical phenomena, making it an essential area of study in fluid dynamics. Understanding the behavior of standing water waves is, therefore, critical for advancing the field of fluid dynamics.

We utilized a mathematical function ζ, also known as the wave profile or waveform, to simulate standing waves. The initial shape of the free surface was specified using this function, as depicted in Figure 5. The test cases were chosen to have

relatively large wave amplitudes and small water heights, enabling us to observe nonlinear effects. We used a wave profile based on the 2nd-order Stokes theory [40] for this purpose.

$$\zeta(x,t) = a\cos(kx)\cos(\omega t) + \frac{\pi a^2}{\lambda}\left[\cos^2(\omega t) - \frac{1}{4\cosh^2(kH)} + \frac{3\cos(2\omega t)}{4\sinh^2(kH)}\right]\cos(2kx) \quad (10)$$

$$\frac{\partial \varphi}{\partial x} = \left(\frac{H}{2}\frac{\cosh\big(k(y+h)\big)}{\sinh(kh)}\cos(kx-\omega t) + \frac{3}{16}h^2\omega k\frac{\cosh\big(k(y+h)\big)}{\sinh(kh)}\frac{\cosh\big(2k(y+h}{\sinh^3(kh)}\right. \cos 2(kx-\omega t) \quad (11)$$

Here, a is the amplitude of the wave, k is the wave number $(2\pi/\lambda)$, ω is the angular frequency $(2\pi/T)$, and T is the period of the wave and h is the height of the wave at position x and time t. The wave profile used in our simulation takes into account nonlinearity and dispersion effects, which give rise to a wave shape different from that of a purely sinusoidal wave predicted by linear wave theory.

We consider a wave with amplitudes of 0.01 m and 0.02 m, a wavelength of 1.0 m, and a mean water depth of H=0.1 m, as studied by Santos and Greaves [32]. To generate the initial profile for the standing wave, we utilized the "arc" utility, which is a built-in tool in OpenFOAM. This utility writes the initial profile based on known wave theories over all grid points of the free surface boundary. The initial point displacements are then extracted from the grid points and saved in the pointDisplacement file, which is located in the 0 folder of the case directory. Once the initial values for the velocity potential and the initial shape of the standing wave are set, we run the "potDyMFoam" solver to simulate the wave motion over the run time of the case.

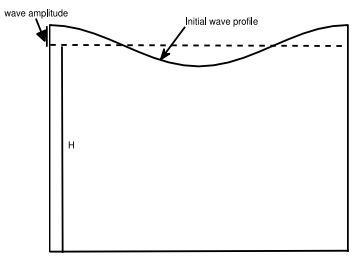
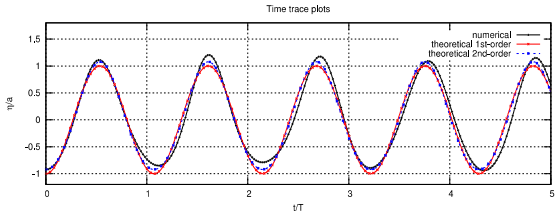
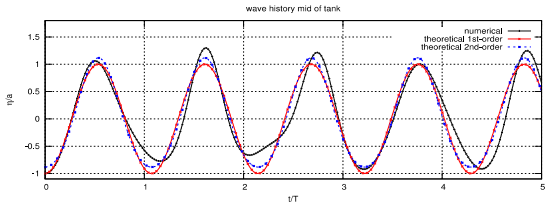


Figure 5: The starting shape of the stationary wave

Figure 6 displays the changes over time in the height of the wave at the halfway point of the free surface boundary, and also shows the results obtained from the linear and 2nd-order Airy wave solutions to compare with the simulation results. The wave elevation is nondimensionalized by the wave amplitude, providing a relative measure of the wave elevation compared to the amplitude. As shown in Figure 6(a) and Figure 6(b), this nondimensionalization results in values between -1 and 1, irrespective of the actual wave amplitude. Similarly, we nondimensionalized time by dividing it by a characteristic time scale, such as the wave period, which can be determined from the relation $T = 2\pi/\omega$, where $\omega = \sqrt{g \times k \times \tanh(kH)}$. The simulation results accurately predict the general behavior of the wave, as predicted by the theoretical solutions. However, the simulations differ from the theoretical solutions in the crest level, where the theories fail to capture the high crest. The simulations show much larger crests and flatter troughs than a linear wave with the same amplitude due to the nonlinear interactions between the different wave components. Santos and Greaves [19] (Fig. 13) also observed similar nonlinear behavior. In a nonlinear wave, the crest can become steeper, and eventually break, resulting in a much larger wave. The nonlinear interactions between the different wave components lead to complex and unpredictable wave behavior [41].



(a)



(b)

Figure 6: A comparison of the time trace history between the current simulations and the predicted theories of the free surface elevation.

3.4. Progressive Waves

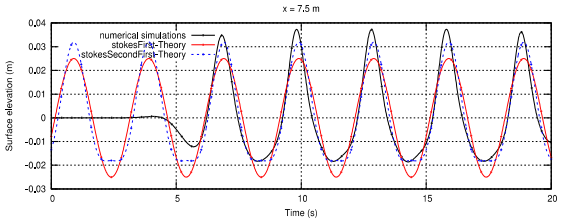
Regular water waves, also known as linear waves, are a type of water wave that follows the linear wave equation, which describes the small amplitude, long wavelength waves that are commonly found in oceans and other large bodies of water. These waves

have a sinusoidal shape, with a constant amplitude and wavelength, and they travel in a particular direction. Regular water waves can be characterized by several parameters, including wavelength λ, wave period T, wave frequency f, wave speed c, and wave height H. The wavelength, period, and frequency are related by the equation $c = \lambda/T = 2\pi f$. In order to simulate progressive waves, it is necessary to specify the inlet boundary as the location where the waves are generated and the outlet boundary as the location where the waves exit the domain. In our simulations, we utilized an inlet boundary condition (7) and a Sommerfeld condition (9) to achieve this. Various test cases with different wave heights were simulated, and for validation purposes, two test cases are presented with their corresponding data in Table 2. To avoid interference between incoming and reflected waves in the computational domain, the tank length was set to L=10 m for both test cases. At the inlet boundary of the domain, the horizontal component of the velocity was determined based on the Stokes-I theory. The wave number k and angular frequency ω of the wave were calculated using the linear dispersion relation $\omega^2 = g \times k \times \tanh(kh)$. To study the wave's behavior over time, free surface elevation was measured at three different locations: 7.5 m, 7.87 m, and 8.424 m from the inlet boundary.

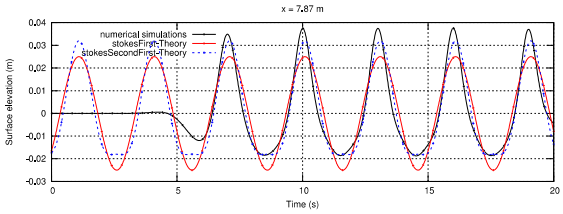
Table 2: Results obtained for a progressive wave.

	Wave amplitude (cm)	Water depth (cm)	Time period (sec)
Case 1	2.5	50	3.0
Case 2	4.5	100	2.0

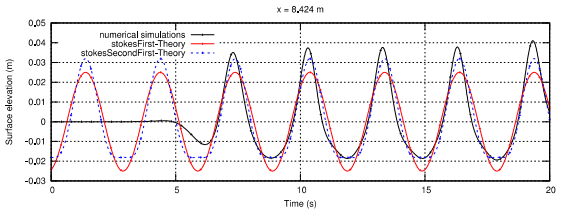
Figure 7 and Figure 8 depict the variation in wave amplitude over time. The x-axis represents time in seconds, while the y-axis represents wave amplitude in meters. In Figure 7, the wave has an amplitude of 2.5 cm, a water depth of H=50 cm, and a time period of T=3.0 seconds. In Figure 8, the wave has an amplitude of 4.5 cm, a water depth of H=10 m, and a time period of T=2.0 seconds. The wave elevation is also plotted based on Stoke's 1st and 2nd-order theories. The numerical simulations for the considered wave parameters generally follow the theoretical trends in terms of time period and phase, but the theories do not capture the non-linear interaction between different wave components. In Figure 7(a), (b), and (c), the wave crests and troughs are symmetric and sinusoidal according to Stokes First theory (linear theory), represented by a red line. The wave profile remains unchanged as the wave travels, and the crest and trough are always aligned with the direction of wave propagation. In contrast, Stokes 2nd theory, represented by the blue dashed line, and the current simulations, represented by the black solid line, do not necessarily produce symmetric or sinusoidal crests and troughs. The non-linear interaction between different wave components can cause the wave profile to become distorted, with the crest becoming steeper and narrower while the trough becomes wider and flatter. This process is known as wave steepening, and it can lead to the formation of breaking waves [41].



(a)

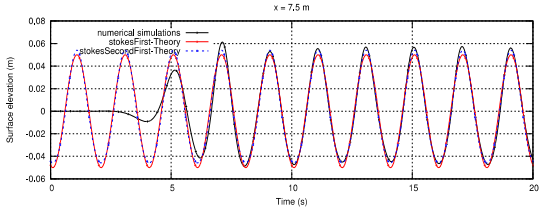


(b)

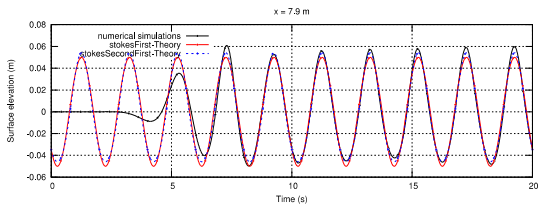


(c)

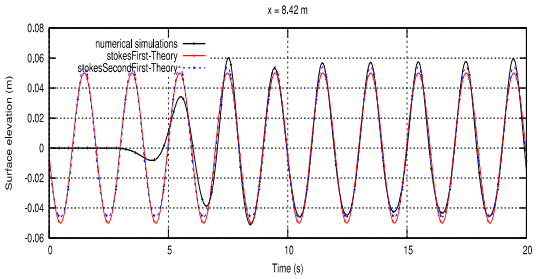
Figure 7: Comparison of the free surface elevation obtained from the first and second Stokes theories with the results obtained from current simulations at three different locations (a) 7.5 m, (b) 7.87 m and (c) 8.24 m from inlet domain, where a=6 cm and T=1.5 sec.



(a)



(b)



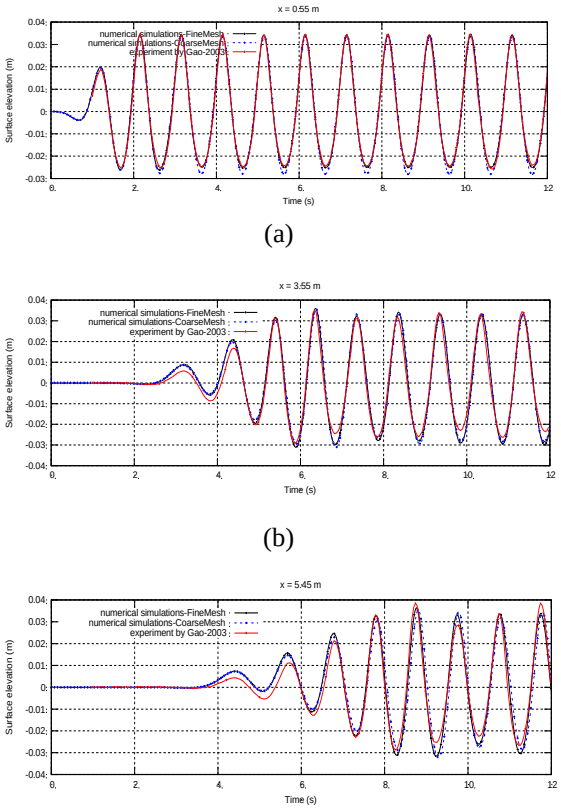
(c)

Figure 8: Comparison of the free surface elevation obtained from the first and second Stokes theories with the results obtained from current simulations at three different locations (a) 7.5 m, (b) 7.9 m and (c) 8.42 m from inlet domain, where H=6 cm and T=2.0 sec.

For the second test case, we modified the wave by increasing its wave amplitude to 4.5 cm and decreasing the time period to T=2.0 seconds. These modifications can make higher-order waves more susceptible to breaking and losing their harmonic structure. The wave crest in Figure 8(a), (b), and (c) is steeper than those in Figure 7(a), (b), and (c) due to the shorter wavelength, which causes an increase in the wave profile's steepness. The current simulations accurately predict this nonlinear behavior that the theories cannot predict. It is crucial to capture this nonlinear behavior as the steeper wave profile can lead to easier wave breaking, causing the higher-order waves to dissipate quickly. To further verify and validate our current numerical simulation results, we conducted a comparison with Feng Gao's experimental data [4]. Feng Gao placed three gauges in the flume to measure wave elevation at positions 0.55 m, 3.55 m, and 5.45 m. The length of the wave tank was maintained at 8.85 m to match the experimental conditions, with water depth H set at 0.28 m. Previous full Navier-Stokes (NS) computations were also conducted for this test case by Qian, Causon, Mingham and Ingram [42] , as well as Bai, Mingham, Causon and Qian [43] . The wave amplitude was set to 0.025 m and the wave period T was set to 1.0 seconds, with the following mathematical expression used for wave generation.

$$u = a\omega \sin(kx - \omega t) \qquad (12)$$

In these simulations, we used a linear ramped function that was added to the system from $t=0$ to $t=T$. To ensure both accuracy and efficiency of the current solver, we utilized both coarser mesh (354 x 17 x 1) and fine mesh (708 x 33 x 1) and compared the obtained solutions at the same locations as the experiment (Figure 9). The purpose of using different mesh sizes was to verify the solver's performance. The experimental data was plotted in red, while the finer mesh and coarse mesh were plotted in black solid line and blue dotted line, respectively. The computational time for the coarse grid was 12 minutes, while the fine mesh required 20 minutes. The results of the current solver showed excellent agreement with the experimental data, very similar to that of Navier-Stokes simulations conducted by Qian Causon, Mingham, and Ingram [42] (Fig. 7(a)) and Bai, Mingham, Causon, and Qian [43] (Fig. 13(a)). The solver successfully captured the steeper wave crests, even with the coarsest discretization used. Although the free surface was not accurately captured at gauge no. 3, the general trend and nonlinear behavior were adequately represented. Gauge 1 and 2 showed excellent agreement with the observations, whereas gauge 3 showed some differences, particularly in terms of dispersion. A possible explanation for this discrepancy could be the change in celerity due to the higher wave amplitude or the discretization near the reflecting wall. Considering the provided dimensional characteristics of the wave flume by Feng Gao's [4] and the wave properties, it appears that the flow in the flume is in the laminar regime. This suggests the presence of wall viscous effects, which could potentially explain any discrepancies observed between our simulations and the experimental data.



(a)

(b)

(c)

Figure 9: The current numerical solver for free surface elevation is being compared with experimental observations at positions (a) Gauge 1 at 0.55 m, (b) Gauge 2 at 3.55 m, and (c) Gauge 3 at 5.45 m.

IV. CONCLUSIONS

In this paper, we have developed a methodology that is able to simulate fluid flow using the OpenFOAM-Extend environment. The method is based on the potential flow solver that uses Laplace's equation to determine the velocity potential at any point in space. The simulation of standing water waves with relatively large wave amplitudes and small water heights enables the observation of nonlinear effects, making it an essential area of study in fluid dynamics. Subsequently, we compared model results with experimental data and obtained excellent agreement, demonstrating the successful implementation of the model. Our findings suggest that the current solver, along with the boundary conditions, is efficient for modeling real-life applications where the flow remains irrotational, inviscid, and incompressible. To facilitate collaboration and knowledge transfer and to improve the solver further, the developed model and corresponding boundary conditions will be released as open source in the OpenFOAM environment. In the future, our focus will be on developing a numerical wave tank that achieves a balance between computational efficiency and accuracy. To achieve this, we plan to employ different solvers for different regions within the tank, allowing us to maintain the required level of detail while optimizing computational resources.

Author Contributions: All authors contributed equally to this work: conceptualization, A,M, DI.G and D.Greaves; methodology, A.M., D.I. Graham and K.Langfeld; software, A.M., D.I. Graham and K.Langfeld; validation, A.M.; writing–original draft preparation, all authors; writing–review and editing, all authors; visualization, all authors; supervision, D.I. Graha, and D.M.Greaves All authors have read and agreed to the published version of the manuscript.

Funding: This research received EPSRC grant reference number EP/K038303/1

Conflicts of Interest: The authors declare no conflict of interest.

REFERENCES

[1] P. Higuera, J. L. Lara and I. J. Losada, "Realistic wave generation and active wave absorption for Navier–Stokes models: Application to OpenFOAM®," Coastal Engineering, vol. 71, pp. 102-118, 2013.

[2] H. Jasak, "Error analysis and estimation for the finite volume method with applications to fluid flows," Imperial College London, London, 1996.

[3] J.-l. Gao, J. Lyu, J.-h. Wang, J. Zhang, Q. Liu, J. Zang and T. Zou, "Study on Transient Gap Resonance with Consideration of the Motion of Floating Body," *China Ocean Engineering*, vol. 36, p. 994–1006, 2022.

[4] F. Gao, "An efficient finite element technique for free surface flow," Brighton University, UK, Brighton, 2003.

[5] Z. Xu and B. Liang, "Experimental study on the breaker bar response to tides," *Coastal Engineering*, vol. 183, p. 104305, 2023.

[6] Q. Fang, J. Liu, R. Hong, A. Guo and H. Li, "Experimental investigation of focused wave action on coastal bridges with box girder," *Coastal Engineering*, vol. 165, p. 103857, 2021.

[7] V. Sriram, T. Schlurmann and S. Schimmels, "Focused wave evolution using linear and second order wavemaker theory," *Applied Ocean Research*, vol. 53, pp. 279-296, 2015.

[8] M. P. d. Ridder, J. Kramer, J. P. d. Bieman and I. Wenneker, "Validation and practical application of nonlinear wave decomposition methods for irregular waves," *Coastal Engineering*, vol. 183, p. 104311, 2023.

[9] I. Akhtar, A. H. Nayfeh and C. J. Ribbens, "On the stability and extension of reduced-order Galerkin models in incompressible flows," *Theoretical and Computational Fluid Dynamics*, vol. 23, p. 213–237, 2009.

[10] J. Liu, Y. Wang and Z. Yuan, "Numerical study on the nonwetting ability of trapezoid topography," *Journal of Fluids and Structures*, vol. 119, p. 103868, 2023.

[11] N. G. Jacobsen, D. R. Fuhrman and J. Fredsøe, "A wave generation toolbox for the open-source CFD library: OpenFoam®," *Numerical Methods in Fluids*, vol. 70, no. 9, pp. 1073-1088, 2011.

[12] J. O'Connor and B. D. Rogers, "A fluid–structure interaction model for free-surface flows and flexible structures using smoothed particle hydrodynamics on a GPU," *Journal of Fluids and Structures*, vol. 104, p. 103312, 2021.

[13] S. T. Grilli, J. Skourup and I. A. Svendsen, "An efficient boundary element method for nonlinear water waves," *Engineering Analysis with Boundary Elements*, vol. 6, no. 2, pp. 97-107, 1989.

[14] G. Wei, J. Kirby, S. Grilli and R. Subramanya, "A Fully Nonlinear Boussinesq Model for Surface Waves. Part 1. Highly Nonlinear Unsteady Waves," *Journal of Fluid Mechanics*, vol. 294, pp. 71-92, 1995.

[15] G. Ducrozet, F. Bonnefoy, D. L. Touzé and P. Ferrant, "Implementation and Validation of Nonlinear Wavemaker Models in a HOS Numerical Wave Tank," *International Journal of Offshore and Polar Engineering*, vol. 16, no. 3, pp. 161-167, 2006.

[16] R. G. Dean and R. A. Dalrymple, Water Wave Mechanics for Engineers and

Scientists, Singapore: World Scientific Publishing C.Pte. Ltd, 1991.

[17] Z. Z. Hu, S. Yan, D. Greaves, T. Mai, A. Raby and Q. Ma, "Investigation of Interaction between Extreme waves and a Moored FPSO using FNPT and CFD Solvers," *Ocean Engineering,* vol. 206, p. 107353–107353., 2020.

[18] S. Yan, Q. W. Ma and J. Wang, "Quadric SFDI for Laplacian Discretisation in Lagrangian Meshless Methods," *Journal of Marine Science and Application,* vol. 19, no. 3, p. 362–380, 2020.

[19] V. Sriram, S. Sannasiraj and V. Sundar, "Simulation of 2-D nonlinear waves using finite element method with cubic spline approximation," *Journal of Fluids and Structures,* vol. 22, no. 5, pp. 663-681, 2006.

[20] B. Chu and X. Zhang, "On the natural frequencies and modal shapes in two-dimensional moonpools with recesses in finite water depth," *Applied Ocean Research,* vol. 115, p. 102787, 2021.

[21] M. I. Mata and M. R. van Gent, "Numerical modelling of wave overtopping discharges at rubble mound breakwaters using OpenFOAM®," *Coastal Engineering,* vol. 181, p. 104274, 2023.

[22] J. R. K. Qwist and E. D. Christensen, "Development and implementation of a Direct Surface Description method for free surface flows in OpenFOAM," *Coastal Engineering,* vol. 179, p. 104227, 2023.

[23] Y. Li, D. Benites-Munoz, C. W. Windt, A. Feichtner, S. Tavakoli, J. Davidson, R. Paredes, T. Quintuna, E. Ransley, M. Colombo, M. Li, P. Cardiff and G. Tabor, "A Review on the Modelling of Wave-Structure Interactions Based on OpenFOAM," *OpenFOAM® Journal,* vol. 2, p. 116–142, 2022.

[24] M. I. Mata and M. R. v. Gent, "Numerical modelling of wave overtopping discharges at rubble mound breakwaters using OpenFOAM," *Coastal Engineering,* vol. 181, p. 104274, 2023.

[25] J. Bohacek, J. Kominek, A. Vakhrushev, E. Karimi-Sibaki and T. Lee, "A Sequential Inverse Heat Conduction Problem in OpenFOAM," *OpenFOAM® Journal,* vol. 1, p. 27–46, 2021.

[26] A. Mehmood, A. Abdelkefi, I. Akhtar, A. H. Nayfeh, A. Nuhait and M. R. Hajj, "Linear and nonlinear active feedback controls for vortex-induced vibrations of circular cylinders," *Journal of Vibration and Control,* vol. 20, no. 8, pp. 213--237, 2012.

[27] W. Benz, " Smooth Particle Hydrodynamics: A Review," in *The Numerical Modelling of Nonlinear Stellar Pulsations. NATO ASI Series*, Dordrecht, Springer, 1990, p. 269–288.

[28] S. Mayer, A. Garapon and L. S. Sørensen, "A fractional step method for unsteady free-surface flow with applications to non-linear wave dynamics," *Intl J Numerical Methods in Fluids,* vol. 28, pp. 293--315, 1998.

[29] P. J. Zwart, G. D. Raithby and M. J. Raw, "The Integrated Space-Time Finite Volume Method and Its Application to Moving Boundary Problems," *Journal of Computational Physics,* vol. 154, no. 2, pp. 497-519, 1999.

[30] D. M. Greaves, A. G. L. Borthwick, G. X. Wu and R. E. Taylor, "A Moving Boundary Finite Element Method for Fully Nonlinear Wave Simulations," *Journal of Ship Research,* vol. 41, pp. 181-194, 1997.

[31] Q. W. Ma, G. X. Wu and R. E. Taylor, "Finite element simulation of fully non-linear interaction between vertical cylinders and steep waves. Part 1: methodology and numerical procedure," *Intl J Numerical Methods in Fluids,* vol. 36, pp. 265-285, 2001.

[32] C. M. S. Santos and D. M. Greaves, "A mixed Lagrangian–Eulerian method for non-linear free surface flows using multigrid on hierarchical Cartesian grids," *Computers & Fluids,* vol. 36, no. 5, pp. 914-923, 2007.

[33] G. X. Wu and R. E. Taylor, "Time stepping solutions of the two-dimensional nonlinear wave radiation problem," *Ocean Engineering,* vol. 22, no. 8, pp. 785-798, 1995.

[34] J. L. Lara, N. Garcia and I. J. Losada, "RANS modelling applied to random wave interaction with submerged permeable structures," *Coastal Engineering,* vol. 53, no. 5-6, pp. 395-417, 2006.

[35] B.-L. Dang, H. Nguyen-Xuan and M. A. Wahab, "An effective approach for VARANS-VOF modelling interactions of wave and perforated breakwater using gradient boosting decision tree algorithm," *Ocean Engineering,* vol. 268, p. 113398, 2023.

[36] P. A. Dirac, "The lorentz transformation and absolute time," *Physica,* vol. 19, no. 1-12, pp. 888-896, 1953.

[37] S. Fouques and C. Pákozdi, "A mixed Eulerian-Lagrangian High-Order Spectral method for the propagation of ocean surface waves over a flat bottom," *Journal of Computational Physics: X,* vol. 8, p. 100071, 2020.

[38] H. S. Udaykumar, W. Shyy and M. M. Rao, "Elafint: A mixed Eulerian-Lagrangian method for fluid flows with complex and moving boundaries," *International Journal*

for Numerical Methods in Fluids, vol. 22, no. 8, pp. 691-712, 1996.

[39] H. Han, Z. Yin, Y. Ning and H. Liu, "Development of a 3D Eulerian/Lagrangian Aircraft Icing Simulation Solver Based on OpenFOAM," *Entropy*, vol. 24, no. 10, p. 1365, 2022.

[40] O. Cozzi, "Free Surface Flow Simulation: Correcting and Benchmarking the ALE Method in Code_Saturne," University of Manchester. School of Mechanical, Aerospace and Civil Engineering, Manchester, 2010.

[41] G. Wu and E. T. Rodney, "Finite element analysis of two-dimensional non-linear transient water waves," *Applied Ocean Research*, vol. 16, no. 6, pp. 363-372., 1994.

[42] L. Qian, D. M. Causon, C. G. Mingham and D. M. Ingram, "A free-surface capturing method for two fluid flows with moving bodies," *Proceedings of The Royal Society a Mathematical Physical and Engineering Sciences*, vol. 462, no. 2065, 2006.

[43] W. Bai, C. G. Mingham, D. M. Causon and L. Qian, "Finite volume simulation of viscous free surface waves using the Cartesian cut cell approach," *Intl J Numerical Methods in Fluids*, vol. 63, pp. 69-95, 2010.

[44] M. Alletto, "Comparison of Overset Mesh with Morphing Mesh: Flow Over a Forced Oscillating and Freely Oscillating 2D Cylinder," *OpenFOAM® Journal*, vol. 2, pp. 13-30, 2022.

[45] H. M. Blackburn and R. D. Henderson, "A study of two-dimensional flow past an oscillating cylinder," *Journal of Fluid Mechanics*, vol. 385, pp. 255 - 286, 1999.

[46] S. Muzaferija and M. Peri´c, "COMPUTATION OF FREE-SURFACE FLOWS USING THE FINITE-VOLUME METHOD AND MOVING GRIDS," *Numerical Heat Transfer, Part B: Fundamentals*, vol. 32, no. 4, pp. 369-384, 1997.

[47] E. Mansard and E. Funke, "The Measurement of Incident and Reflected Spectra Using a Least squares Method," in *In Proceedings of 17th Conference on Coastal Engineering*, Sydney, 1980.