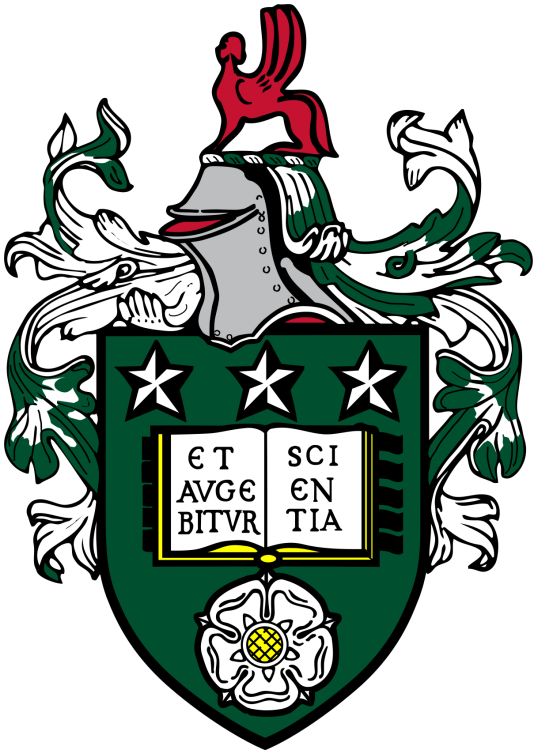
A novel approach for modelling water waves and fluid-structure interactions

Wajiha Rehman

Submitted in accordance with the requirements for the degree of Doctor of Philosophy



The University of Leeds

Department of Applied Mathematics

April 2024

Intellectual Property Statement

The candidate confirms that the work submitted is her own, except where work which has formed part of jointly authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Chapters 2 and 3 include work which is published in:
[69] *Rehman, W., Bokhove, O. and Kelmanson, M. "A Systematic Approach of Developing a Numerical Wavetank to Simulate Driven Shallow- and Deep-Water Waves." Proc. ASME 2023 42nd Int. Conf. on Ocean, Offshore and Arctic Eng.: p. 10. 2023. ASME.* Other minor details are in the EU project report that will be publicly available through the EU website.

The original idea was proposed by O. Bokhove, which is based on the extension of his previous collaborations with F. Gidel [34], T. Salwa[72], and A. Kalogirou [10]. The candidate derived and implemented the mathematical and numerical models of the piston-driven wavetank models based on shallow water equations and potential flow theory. The publication was written in collaboration with O. Bokhove, with proofreading from M. Kelmanson and T. Bunnik.

Chapter 4 includes work which is published in:
[69] *Rehman, W., Bunnik, T., Bokhove, O. and Kelmanson, M. "Experimental Modeling of Water-Wave Interactions with a Flexible Beam." Proc. ASME 2023 42nd Int. Conf. on Ocean, Offshore and Arctic Eng.: p. 10. 2023. ASME.* Moreover, a detailed version of this work is publicly accessible on Earth ArXiv: DOI https:doi.org10.31223X5998B.

The experimental study explained in Chapter 4 is conducted under the supervision of T. Bunnik, with proofreading from O. Bokhove, and M. Kelmanson.

Chapters 5 and 6 include work from an ASME manuscript which has been submitted:
Rehman, W., Bunnik, T. "Fluid-structure interaction modelling of the regular water waves impact on a flexible beam." Proc. ASME 2024 Power and Energy Conference.: p. 10. 2024. ASME.

This work has been completed under the supervision of T. Bunnik, and L. Kaydihan. The candidate used MARIN's in-house linear and high-fidelity fluid-structure interactions analysis software to conduct this study.

This doctoral dissertation was conducted as a part of an EU-funded project entitled "*Eagre/Aegir: high-seas wave-impact modelling*"(DOI 10.3030/859983), with grant agreement ID: 859983. The detailed research plan is explained in ANNEX 1 of the **Grant Agreement-859983-EAGRE.pdf**. Preliminary findings presented in this thesis have been incorporated into deliverable reports for the European Union.

Acknowledgements

Since childhood, I connected my life's purpose with serving humanity and this project is a significant step towards it because knowledge is an everlasting contribution. This thesis is the result of unconditional support, help, encouragement and guidance from many individuals, to whom I want to extend my gratitude.

Being a believer, I would start by paying my gratitude to Allah and Prophet Muhammad (PBUH) for being my light and hope in the darkest of times. I am extremely thankful to my supervisors Prof. Onno Bokhove, and Prof. Mark Kelmanson from the University of Leeds, and Dr Tim Bunnik from Maritime Research Institute Netherlands (MARIN) for sharing their vast knowledge, unwavering support and valuable advice. Special thanks to Onno for choosing me for this research opportunity and for all the hard work he has put into efficiently managing this project in the challenging times of the COVID-19 pandemic.

Thanks to the Firedrake developers for their technical assistance. I am especially grateful to Dr Koki Sagiyama for his advice and support with code implementation.

Many thanks to my friends and colleagues from the School of Maths, Joseph, Janet, Jonny, Muyang and Sakina for the valuable discussions and their company. Special thanks to my friends, Fatima Sehar, Shazia Waqas, Asad, Tianqi, Rukia, Sara, Wafa, Isha, Simi, Aqsa, Marlous, Umar, and Azhar, for being my family far from my family and for their support during the COVID-19 pandemic. A particular gratitude is reserved for Yang Lu for his companionship and scientific help throughout the extensive PhD journey.

On a personal note, I would like to express my love and heartfelt gratitude to my parents, Shahjahan and Abdul Rehman, and my siblings, Fatima, Abdullah, Zain, Farhan, and Maria for their unconditional love, support and encouragement in every phase of my life. Thank you for being with me in all the ups and downs of life. Special thanks to Tania Imran for

her companionship in Wageningen and for proofreading my OMAE presentations and thesis. I genuinely enjoyed your company and jokes.

I also want to thank my teachers, Shellah Aziz, Shagufta Aslam, Saba Qamer, Dr Muhammad Farhan, Dr Faraz Fazal, and Dr Rabiya who made me a science enthusiast. Lastly, I am grateful to the European Commission, Marie Curie Actions - Initial Training Networks (ITN), project number 859983, for funding this project.

Abstract

The significance of the maritime industry is undeniable in the sectors of global trading, renewable energy, and oil and gas industry. As a growing industry, there is continuous research on the development of better ships, efficient wind farms and reliable floating structures for which wavetanks are extensively used to perform scaled-model testing. However, such experimental campaigns are not only expensive but also time-consuming, therefore, limiting the number of tests required for data acquisition. To address this problem, researchers have proposed numerous numerical models to replace extensive experimental testing but these models are limited in their applications. Hence, we present a solution which is a novel approach for developing numerical wavetank models that can simulate a broad spectrum of water dynamics, i.e. shallow- and deep-water dynamics, and can be extended to analyse water-wave interactions with flexible offshore structures, thereby offering applicability to a wide array of maritime industrial challenges.

The establishment of such mathematical and numerical wavetank models is a challenge which can only be accomplished systematically. The dynamics of linear and nonlinear water waves are governed by a variational principle (VP) that emphasizes the conservative structure of nonlinear water-wave dynamics. This is because energy and mass conservation, as well as the conservation of phase-space volume, are intimately connected with the conservative structure. A novel feature of our model is that it implements the time-discretized variational principle directly in the finite-element-based environment, *Firedrake*, which automates the derivation of time-discretized weak formulations and subsequently reduces the time and effort for the model implementation. At first, we developed a depth-averaged numerical wavetank model based on linear and nonlinear shallow water dynamics and established our novel approach through extensive testing. After that, we increased the model's complexity by developing a piston-driven numerical wavetank model based on linear and nonlinear potential flow theory. Our time-discretized VP-based model is capable of generating a numerical representation of actual

wavetanks by including piston wavemaker for wave generation and beaches for wave absorption. The results from the novel approach are promising and we are confident that this approach will simplify the development process of waveflap-driven numerical wavetank model, a widely utilized tool in the maritime industry. Furthermore, we have also developed and shared a hyperelastic structure model that can be coupled with the numerical wavetank to solve fluid-structure interaction (FSI) problems. However, the coupling is still in the development stages and is not presented in this thesis.

The numerical wavetank models must undergo experimental validation before they are deemed suitable for industrial use. Recognizing this necessity, we designed an experimental setup which is capable of measuring incoming waves, the structure's accelerations in response to the wave interactions, and waves reflected from the structure, simultaneously. After designing the setup, we conducted a series of experiments under a wide range of sea conditions ranging from regular-to-irregular and moderate-to-extreme wave height and steepness. The study of such a wide range of conditions makes the experiments suitable for providing reliable data in the validation of a suite of mathematical and numerical FSI solvers, i.e., linear, nonlinear and high-fidelity. The data from the experiments has been made publicly available through open-source data-sharing platforms. Lastly, we have used the experimental data to validate MARIN's in-house linear and high-fidelity FSI solvers, which confirms that the data is suitable for validation purposes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project overview

This PhD project falls under Work Package 2 (WP2): "Wave Turbine Impact" of a European Industry Doctorate (EID) project: "Eagre/Aegir: High-Seas Wave-Impact Modelling" that is funded by the H2020 Marie Skłodowska-Curie programme. The project aims to develop an affordable and practical numerical tool for simulating water waves and their interactions with the offshore wind turbine's mast and can be employed by the maritime industry for designing offshore wind turbine farms. The target of creating such a tool will be achieved in two major steps, as follows:

1. Developing the numerical wavetank which is a representation of the actual experimental wavetanks and can be extended to perform fluid-structure interaction (FSI) analysis, i.e. water-wave interactions with the hyperelastic structures.

2. Experimental modelling of water-wave interactions with a flexible beam to perform benchmark test cases for validation of the numerical solvers that solve fluid-structure interactions (FSI) problems.

The initiation of this EID program is driven by industrial demand, which is the development of an effective numerical tool for modelling water waves and their interactions with offshore structures. The purpose of an EID project is to increase the collaboration between industry and academia for better training of researchers, therefore, this project is done in collaboration with the Maritime Research Institute Netherlands (MARIN). The first eighteen months of the

research project were spent at the University of Leeds under the supervision of Prof. Onno Bokhove and the second eighteen months were spent at Maritime Research Institute (MARIN) Academy BV, The Netherlands, under the supervision of Dr Tim Bunnik, Sanne van Essen, and Dr. Bulent Duz. The mathematical model and codes are developed at the University of Leeds while the experimental modelling of water-beam interactions is performed at MARIN Academy BV. Furthermore, validation of MARIN's in-house linear and nonlinear FSI software has been carried out by utilizing the data from experimental modelling.

1.2 Motivation and objectives

Keeping in view the project targets and industrial demands, the thesis can be divided into two parts. The first part is about the mathematical and numerical modelling of numerical wavetank models that are driven by piston wavemakers to simulate shallow- and deep-water dynamics and can be extended to perform FSI analysis. The second part concerns the experimental study and validation of the numerical tools which are used for modelling fluid-structure interaction problems, i.e. water-wave interactions with a flexible beam. This section briefly introduces each of the aforementioned parts, with more detailed explanations provided in subsequent chapters of this thesis.

1.2.1 Mathematical and numerical modelling of a wavetank with versatile applications

The maritime industry frequently uses wavetanks to study water waves and their interactions with maritime structures in an experimental environment. The water waves are generated in the wavetanks using different types of wavemakers whose specific form depends on the desired sea state, i.e., the relation between the water depth with the amplitude and wavelength of the water waves to be modelled. For example, to model shallow- or deep-water dynamics, wavetanks are equipped with piston- or waveflap-wavemakers respectively. Model basin tests can be used to validate numerical models. However, performing such laboratory experiments is expensive and time-consuming, potentially impeding the acquisition of sufficient experimental data. Hence, mathematical and numerical models of wavetanks are extensively used to facilitate initial design processes and to validate and augment laboratory wavetank experiments. Additionally, numerical simulations are very useful in the early design stage when model tests are not yet an

option.

Gidel [34] has successfully implemented the variational principle (VP) of potential flow equations to develop a numerical wavetank model driven by a piston wavemaker by utilizing the classical approach of VP implementation, i.e. time-discrete weak formulations. This wavetank model can accurately predict the water-wave dynamics in shallow water conditions, whereas an effective numerical wavetank to simulate deep-water dynamics still needs to be developed. Thus we aimed to develop a mathematical and numerical model of a waveflap-wavemaker wavetank by implementing the variational principle (VP) of nonlinear potential-flow equations in the finite-element-based environment *Firedrake*. However, establishing the mathematical and numerical model of such a wavetank by using Gidel [34] (classical) approach for VP implementation in *Firedrake* is complex and time-consuming. The two-dimensional waveflap-wavemaker boundary motion of the computational domain further increases the complexity far beyond that of the piston-wavemaker problem in which the boundary motion is one-dimensional. Hence, we have developed a novel approach for implementing the equations of motion by using a time-discrete VP instead of weak formulations. The finite-element compiler architecture *Firedrake* allows this method to automatically generate the time-discrete weak formulation, thereby reducing both the likelihood of human error and the time taken to develop and implement the code. Therefore, the focus of this thesis is on the development of the new approach instead of the development of the waveflap-driven numerical wavetank model.

The development of the novel approach is done systematically so that it can be verified and validated at each development stage. First, the VPs of linear and nonlinear shallow-water equations are implemented in *Firedrake* to develop a numerical piston-wavemaker wavetank. Second, the complexity of the first model is increased by deriving the VP based on potential-flow theory to develop a numerical piston-wavemaker wavetank. Additionally, we have briefly discussed our strategy [69] for the waveflap-driven wavetank model. The proposed systematic approach facilitates not only cumulative validation of results but also understanding and implementation for both developer and user. This novel approach holds promise for future extensions of the piston-driven wavetank model to waveflap-driven wavetank model, within manageable development times, which can be further extended to numerical monolithic models of fluid-structure-interactions based on coupled VPs for fluid and structure.

1.2.2 Experimental modelling of water-wave interactions with a flexible beam

In the maritime industry, mathematical and numerical modelling is gaining significance because experimental scaled-model testing is not always feasible in early design stages due to time and budgetary constraints. Furthermore, the experimental modelling of flexible structures at the model scale is not straightforward, motivating researchers to develop mathematical and numerical models for solving FSI problems. These models generally fall into two categories. First, they range from straightforward linear shallow-water equations and linear modal analysis to intermediate-complexity linear potential-flow solvers coupled to linear elastic structural equations [72, 71]. Second, there are more sophisticated approaches based on nonlinear potential flow, Navier-Stokes (NS) equations [86], and Smoothed Particle Hydrodynamics (SPH) [22] coupled with nonlinear hyperelastic structural equations. However, results generated by numerical models require validation using benchmark experimental data.

Therefore, a series of experiments, divided into three experimental cases, have been performed with two goals in mind [69]: to understand fluid-structure interactions (FSIs) of waves impacting on a flexible beam with simultaneous measurements of beam accelerations and incident and reflected waves; and, to use the acquired data set for validation of a hierarchy of FSI numerical models. The experiments are divided into three cases, each of which aims to study the dynamic response of the flexible beam to varying wave conditions; from regular-to-irregular and moderate-to-extreme wave height and steepness. Experimental Case 1 concerns interactions with the flexible beam when the carriage is at rest; this case will help in validating the linear FSI solvers in the non-resonant regime, as the natural frequencies of the beam are higher than the wave frequencies. Experimental Case 2 concerns interactions with the flexible beam when the carriage is moving at a constant speed. Moving the carriage changes the frequency of encounter with the waves, so that the dynamic response of the beam and its interaction with water waves, particularly at the onset of resonance, can be studied. By changing the steepness of the regular wave, both linear and nonlinear FSI solvers can be validated. Experimental Case 3 concerns steep, irregular wave interactions with the flexible beam when the carriage is at rest. This is the most complex case and is designed to yield data on structural dynamics due to nonlinear wave-loading processes related to steep and breaking waves. This case will help to validate the high-fidelity FSI solvers. Hence, the study covers a wide range of FSI problems that can be used to establish benchmarks for FSI-code validations.

To achieve the second goal of the study, i.e. validation of linear, nonlinear and high-fidelity numerical FSI solvers that are commonly employed by the maritime industry in the design of fixed-foundation offshore wind turbines, it is essential to make experimental data publicly accessible. Hence, an open-source online platform, specifically including a GitHub[1] public repository, is created to make the data available to the public. Moreover, we have shared the details of the experiments with the scientific community in the form of a published research article [69]. Finally, we have also performed the validation study of two in-house FSI tools used at MARIN, i.e. SEACAL and ReFRESCO. This study confirms that the experimental data is of high quality and can be used for the validation of the numerical FSI solvers.

1.3 Thesis outline

The thesis is divided into eight chapters. As described in the previous section, the thesis can be divided into two main parts, i.e. i) development of the numerical models of wavetanks, and ii) experimental modelling of water-wave interaction with a flexible beam. Hence, Chapter 2 and 3 are dedicated to the systematic development, verification and validation of the numerical wavetank models, and Chapter 4 to 6 provide a detailed description of the experimental study and the validation of linear (SEACAL) and nonlinear (ReFRESCO) FSI tools used in MARIN. A brief overview of each chapter is as follows:

- Chapter 1 is dedicated to the introduction of the problem, an explanation of the motivation and objectives, and a description of the thesis outline.
- Chapter 2 explains the mathematical and numerical modelling of the wavetank models based on shallow water equations. In this chapter, we have developed and tested a novel approach for the implementation of variational problems in a finite-element-based environment Firedrake. The new approach is to implement a time-discrete variational principle instead of time-discrete weak formulations as it automates the derivation of time-discrete weak formulations and reduces human time and error at the code implementation stage. Furthermore, this approach has the potential to facilitate the development process of a waveflap-driven numerical wavetank model based on potential flow equations.
- Chapter 3 describes the derivation of the mathematical model and the numerical implementation of the piston-driven wavetank models based on potential flow equations by

[1] https://github.com/EAGRE-water-wave-impact-modelling/FSI_Experiments

utilising the novel approach developed in Chapter 2. The results obtained from the numerical analysis are also presented in a published article [67].

- Chapter 4 explains the experimental modelling of the water-wave interactions with a flexible beam which is conducted at MARIN's concept basin. The beam's response is tested under a wide range of wave conditions and parameters. The experimental setup is designed such that it admits the simultaneous measurements of incident waves and the beam's response and, therefore, it is suitable for studying FSI problems and validation of the numerical solvers for FSI problems [70]. A GitHub public repository is also created to make the data available to the public and details of this experimental study have been published in the form of a research article [69].

- Chapter 5 covers the validation of a MARIN's in-house linear FSI solver, i.e. SEACAL [58], which is based on linear potential flow theory. The numerical results obtained from SEACAL are compared with the experimental data obtained from Case 1, followed by conclusions. This validation study confirms that the experimental data is suitable for the validation of numerical FSI solvers.

- Chapter 6 is about the validation of MARIN's in-house nonlinear FSI solver, i.e. ReFRESCO [57], which is based on Reynolds Averaged Navier Stokes Equations (RANSE). The test cases from experimental Case 1 and Case 3, in Chapter 4, are simulated in ReRESCO and the numerical results obtained from ReFRESCO are compared with the experimental data.

- Chapter 7 is dedicated to the explanation and implementation of the codes developed for the simulation of the numerical wavetank models in finite-element based environment Firedrake [66, 4, 5]. Additionally, a public GitHub repository has been created to share the codes.

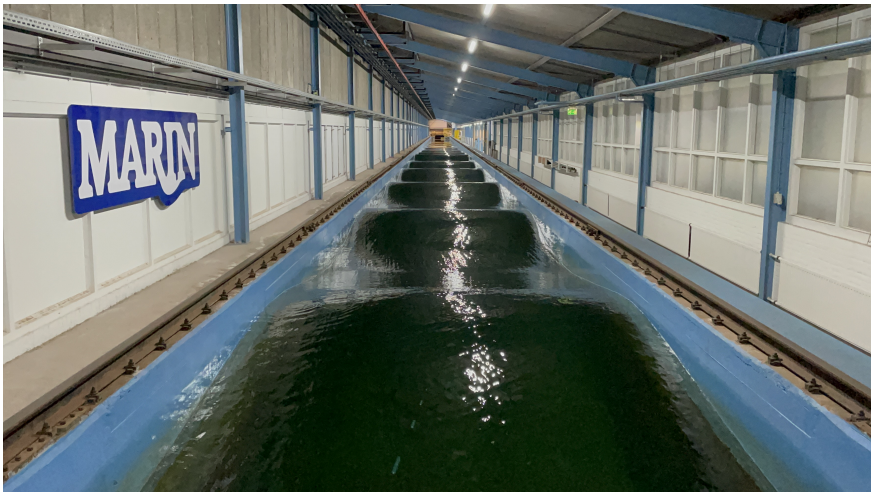
- Chapter 8 summarizes this dissertation, presents key conclusions, suggests improvements and possible extensions of the present work, and provides a detailed account of the outreach activities.

Chapter 2

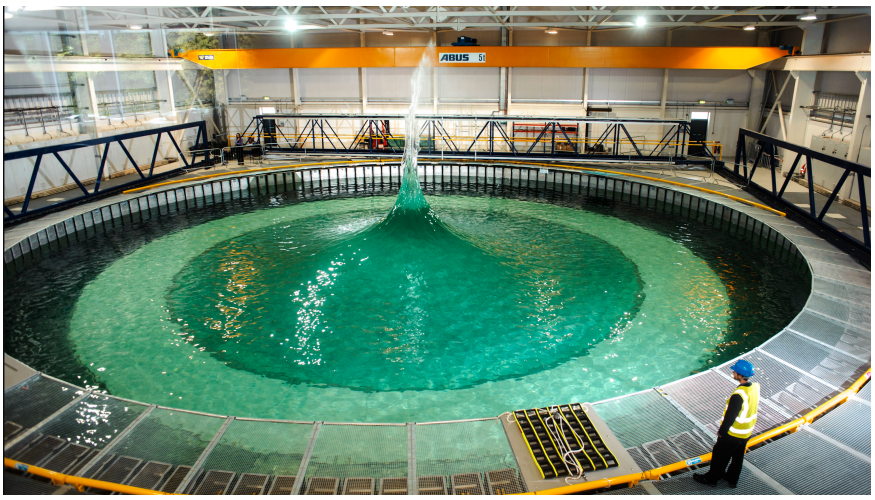
Systematic development of a novel approach for better modelling of wavemaker-driven numerical wavetanks

2.1 Introduction

Wavetanks have prime significance in the maritime industry for the experimental modelling of water waves and their interactions with scaled offshore structures. A typical wavetank is a water-filled rectangular-shaped tank which has a wavemaker to generate waves at one end while the other end has a wave-absorbing surface such as porous beaches. Wavetanks of different shapes and dimensions are equipped with different types of wavemakers to simulate specific sea states, as specified by the relation between water depth, amplitude and wavelength of the modelled water waves. To model shallow-water waves, wavetanks are equipped with piston-type wavemakers; and, to model intermediate-depth or deep-water waves, wavetanks tend to be equipped with waveflap-type wavemakers. Fig. 2.1 shows an instant when scaled experimental modelling of water waves is performed in different wavetanks, i.e., a rectangular wavetank with regular waves and a circular wavetank with a freak wave [79].



(a) Regular water waves generated in a rectangular wavetank at MARIN. Photo courtesy of MARIN.



(b) A freak wave generated in a circular wavetank [79].

Figure 2.1: Two different types of wavetanks are shown.

The challenges associated with scaled model tests are that they are time-consuming, budget-intensive, and sometimes not suitable because the scaled model is not sufficiently large to capture the actual physics, thus limiting the maritime industry to test the models in a wide range of sea conditions. Therefore, to address the challenges faced by the maritime industry and to facilitate the initial design process, researchers have proposed mathematical and numerical models to simulate water dynamics in the wavetanks.

Before explaining our model of the numerical wavetank we give a brief overview of different numerical wavetanks models in §2.2. After that, we explain our mathematical model in §2.3 and its numerical implementation in §2.4. Section §2.5 shows the comparison of numerical results obtained from our approach with the classical approach and conclusion is drawn in §2.7.

2.2 A brief overview of numerical wavetanks

Similar to an actual wavetank, a typical numerical wavetank has three zones; (i) wave generation, (ii) wave propagation, and (iii) wave absorption. At first, the water waves of desired amplitude and period are generated in the wave generation zone which are then propagated through the computational domain until they reach the wave absorption zone where these waves are absorbed. Fig 2.2 demonstrates these zones [85].

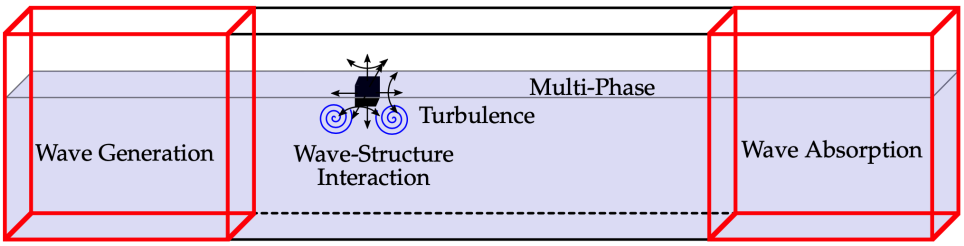


Figure 2.2: Different zones of a typical numerical wavetank are depicted [85].

Each zone has its specific role which can be achieved by a variety of mathematical and numerical models which are briefly explained in this section.

2.2.1 Wave generation and absorption zones

Windt et al. [85] have presented a thorough assessment of a variety of numerical wavemaker and absorption methodologies which are summarised in Fig 2.3. Generally, wave generation methods can be classified into five categories, whereas wave absorption methods can be classified into six categories, described as follows.

- In the relaxation zone method (RZM)[41], the wave signal is gradually introduced in the wavetank over a certain distance known as the relaxation zone. Within the relaxation zone, the generated wave signal is smoothly adjusted with the target wave signal through an iterative process which depends on the implemented algorithm. Thus, minimizing any artificial effects or disturbances that may occur if the waves were abruptly introduced into the tank. This method ensures that the generated waves accurately represent the target wave conditions for various numerical simulations or experiments conducted in the wave tank.

- In the static boundary method (SBM), the wave velocity and wave elevation are defined as the Dirichlet boundary condition at the inlet and outlet of the numerical wavetank.

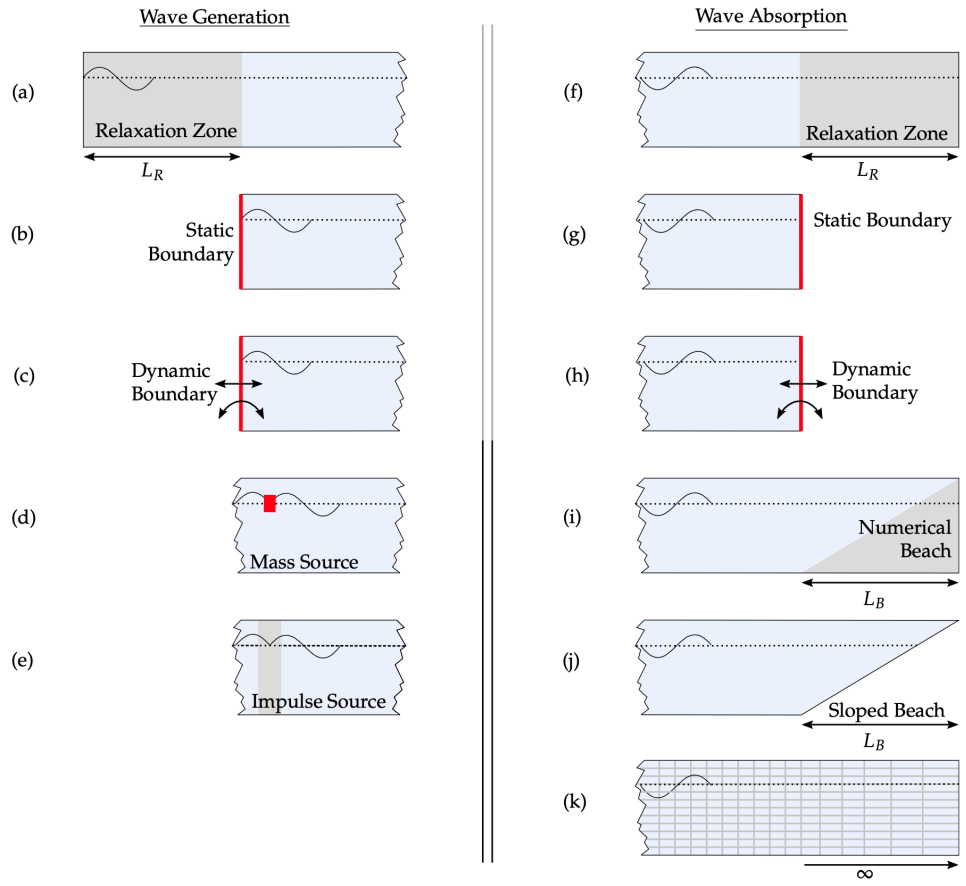


Figure 2.3: Schematic of different methods for the wave generation and absorption are presented [85].

This method requires less computational time than RZM.

- In the dynamic boundary method (DBM), the velocity inlet boundary of the numerical wavetank replicates the wavemaker motion of the actual experimental wavetank, hence, allowing a real representation of the physical problem. This allows maritime researchers to use the experimental wave signal, i.e. the time series of wavemaker displacement and velocity, as input to the numerical model. Similar to wave generation, in DBM the waves can be absorbed by a moving boundary whose motion is controlled by experimental measurements, analytical expressions or forced feedback. Implementation of DBM can be found in [35, 63].

- Mass source method (MSM)[54] and impulse source method (ISM)[17] are utilised in Reynolds Averaged Navier Stokes (RANS)-based numerical wavetanks by adding a source term to the RANS equations which are responsible for wave generation, whereas a separate beach is required for wave absorption.

For numerical wave absorption, there are three other methods which are not discussed previously, i.e. numerical beaches, sloped beaches, and mesh stretching. Numerical beaches absorb the waves by means of numerical dissipation which is done by introducing a damping term in the mathematical model. The damping term varies gradually from the start to the end of the numerical beach to avoid reflections due to the sharp interface. Another method is to change the domain shape by introducing a slop and letting the mathematical model replicate real-world physics. This method is called sloped beach. The mesh stretching method uses gradually increasing cell size towards the far field of the fluid domain which filters out the waves smaller than the mesh size. Examples of the mesh stretching method can be found in [8, 23].

Based on these studies, we have selected a wave generation method based on the dynamic boundary method since we aim to facilitate the maritime industry by replicating their actual wavetank and this method provides the most realistic model. The wave signal generated by a wave generation method is used as the initial condition for the mathematical model which numerically simulates wave propagation and interactions.

2.2.2 Wave propagation zone

The wave propagation zone is where the problems concerning the wave interactions, both wave-to-wave and wave-to-structure, are investigated through numerically discredited mathematical models. The well-known governing equations or mathematical models for the development of numerical wavetanks include viscid Reynolds Averaged Navier Stokes equations (RANSE), inviscid Laplace equation, Boussinesq-type equations, and shallow water equations.

RANSE-based numerical wavetank models

In the field of fluid mechanics, the flow of an incompressible (constant density) and viscous fluid can be expressed by a set of partial differential equations, also known as Navier-Stokes equations (NSE), i.e. mass conservation or continuity equation, and momentum conservation equation. In the Cartesian plane, the fluid flow can be described as a vector field that varies in space and time [82]. The velocity vector in three-dimensional space is written as:

$$\vec{V}(x_1, x_2, x_3, , t) = u_1(x_1, x_2, x_3, t)\vec{i} + u_2(x_1, x_2, x_3, t)\vec{j} + u_3(x_1, x_2, x_3, t)\vec{k}, \tag{2.1}$$

where, u_1, u_2, and u_3 respectively are the velocity components corresponding to x_1-, x_2-, and x_3- coordinate axes. To satisfy the incompressibility condition, the divergence of the velocity field must be zero, i.e.

$$\nabla \cdot \vec{V} = \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3} = 0. \tag{2.2}$$

Therefore, the continuity equation can be written as:

$$\nabla \cdot (\rho \vec{V}) = 0, \tag{2.3}$$

where ρ is the fluid density. The conservation of momentum equation is given as

$$\rho\Big(\frac{\partial \vec{V}}{\partial t} + \vec{V} \cdot \nabla \vec{V}\Big) = -\nabla P + \nabla \cdot \boldsymbol{\tau} + \vec{F_b}, \tag{2.4}$$

where $\frac{\partial \vec{V}}{\partial t}$ is acceleration; $\vec{V} \cdot \nabla \vec{V}$ is convective acceleration; $-\nabla P$ is pressure term; τ_{ij} is the viscous part of the stress tensor for a Newtonian fluid, given as $\mu\Big(\frac{\partial V_i}{\partial x_j} + \frac{\partial V_j}{\partial x_i}\Big)$; and $\vec{F_b}$ represents body forces acting on the fluid, e.g. gravity. In addition to the Navier stokes equations, a transport equation i.e. conservation of arbitrary scalar quantity ϕ is also solved. The generic transport equation for ϕ is given as:

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla \cdot (\rho\phi V) = \nabla \cdot (\nu \nabla \phi) + \rho Q_\phi, \tag{2.5}$$

where ν is diffusivity of ϕ, and Q_ϕ represents a source and sink of ϕ. Capturing turbulent flow by numerically solving NSE requires a very fine spatial and temporal discretisation which is computationally intensive and time-consuming. Therefore, instead of using NSE, the problem is solved by using Reynolds Averaged Navier Stokes equations (RANSE). In RANSE [83], the instantaneous quantities of velocity and pressure are decomposed into fluctuating and time-averaged components. Let u be the instantaneous component of velocity V, then it can be written as a combination of a time-averaged term $\bar{u_i}(x_i)$ and a fluctuating term $u_i'(x_i, t)$, as follows:

$$u_i(x_i, t) = \bar{u_i}(x_i) + u_i'(x_i, t). \tag{2.6}$$

The momentum conservation after Reynolds averaging is written as:

$$\rho(\frac{\partial \bar{u}_i}{\partial t} + \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j}) = -\frac{\partial \bar{P}}{\partial x_i} + \mu \frac{\partial^2 \vec{V_i}}{\partial x_j \partial x_j} - \rho \frac{\partial u_i' u_j'}{\partial x_j} + \rho F_b \tag{2.7}$$

Here, $\frac{\partial u_i' u_j'}{\partial x_j}$ is an additional unknown term called Reynolds stresses. To close the equations, Reynolds stresses must be modelled by equations of known quantities. In 1877, Boussinesq [12] proposed a formula for Reynolds stresses based on molecular viscosity theory which is given as follows:

$$-\rho \bar{u_i' u_i'} = \tau_{ij} = \mu_t (\frac{\partial \bar{u}_j}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}) - \frac{2}{3}(pk + \mu_t \frac{\partial \bar{u}_k}{\partial x_k})\delta_{ij} \tag{2.8}$$

where, μ_t and k are turbulent viscosity and turbulent kinetic energy respectively. Turbulent viscosity should be modelled again to close the system of equations. The most frequently used models for this purpose are known as the two-equation eddy viscosity models which are k–ϵ, k–ω, and SST models. The two-equation models have two partial differential equations; one for the turbulence length scale and the other for the turbulent velocity scale.

RANSE-based numerical wavetanks are becoming common in maritime industry due to the rising availability and computational power of high-performance computing systems. Many researchers [84, 41, 62] have developed RANSE- based numerical wavetanks to predict the performance of wave energy converter devices [74], seakeeping performance of a ship [78], designing of offshore floater [47], and response of offshore structures [19, 55]. The RANSE-based numerical wavetanks can predict the nonlinear turbulent flows, effects of viscosity and intricate free surface elevations like wave breaking accurately; however, at high computational cost and time which makes them infeasible to solve the problem at a large computational domain and long duration. To overcome this challenge, researchers have proposed low-fidelity models based on potential flow theory which are described next in this section.

Potential flow equations based numerical wavetanks

Potential flow equations are based on potential flow theory in which we assume the fluid flow ($\vec{V}(x, y, z)$) is inviscid, irrotational ($\nabla \times \vec{V} = 0$), and incompressible,

$$\nabla^2 \phi = 0, \tag{2.9}$$

which is the Laplace equation. In potential flow equations-based methods, we find the velocity potential ($\phi(x, y, z, t)$) by solving the Laplace equation with free-surface boundary conditions, i.e. kinematic boundary conditions, and dynamic boundary conditions, which can be complemented by additional boundary conditions in the computational domain, for example, no-slip

boundary condition at the bottom of the wavetank, and velocity boundary condition at the wavemaker surface. Once velocity potential is known we can find the fluid velocity by using the following relation:

$$\nabla\phi = \vec{V}(x, y, z). \tag{2.10}$$

The kinematic free-surface boundary condition specifies the behaviour of fluid interface with its surroundings which is air which is known as the free surface of the fluid at $z = h(x, y, t)$, where z is the vertical coordinate and $h(x, y, t) = H + \eta(x, y, t)$ is the total fluid depth. It states that the fluid particles at the fluid-free surface stay at the interface, i.e. there is fluid flow along the free surface or interface but not through it. It is given as

$$\frac{\partial h}{\partial t} + \nabla\phi \cdot \nabla h - \frac{\partial \phi}{\partial z} = 0, \tag{2.11}$$

where the first term shows free surface is time-dependent, the second term shows fluid is space-dependent, and the third term is a component of fluid velocity in z direction, i.e. perpendicular to the free surface. The dynamic free-surface boundary condition also known as the unsteady Bernoulli equation of motion states that the pressure exerted by fluid particles is in equilibrium with the pressure exerted by the surroundings. At atmospheric pressure, the dynamic boundary condition is given as:

$$\frac{\partial \phi}{\partial t} + \frac{1}{2}(\nabla^2\phi) + g(h - H_0) = 0, \tag{2.12}$$

where the second and third terms denote the kinetic and potential energy of the fluid, respectively. It is challenging to solve these equations because the free surface is not only a boundary of the domain but also the unknown of the problem. Researchers have developed numerical wavetank models based on linear and nonlinear potential flow equations to compute regular- [87, 29, 71] and irregular-wave [11] loading on structures and found satisfactory results from experimental validation.

Wavetanks based on potential flow equations are capable of simulating deep-water dynamics. The water dynamics are considered deep when the water depth h is greater than the half of wavelength (Λ), i.e. $h/\Lambda \geq 1/2$, as in this condition water dynamics are effectively unaffected by the bottom ($z = 0$) and fluid particles follow the orbital path. On the other hand, when the water depth is 20 times smaller than the wavelength, i.e. $h/\Lambda \leq 1/20$, the water dynamics are strongly affected by the bottom ($z = 0$) and water particles follow the elliptical path. To model

shallow-water dynamics we need to solve shallow-water equations which are described next in this section. Fig. 2.4 [25] demonstrates the transition of water particles' motion as the ratio of water depth and wavelength varies.

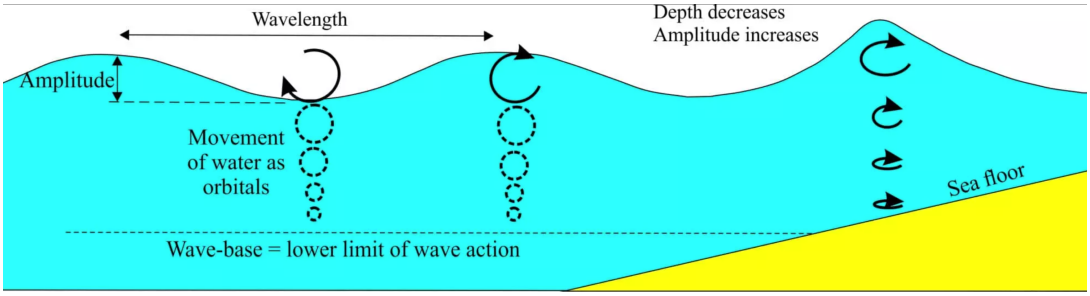


Figure 2.4: A schematic demonstration of water-particle motion when the ratio of water-depth to -wavelength varies [25].

Shallow-water equations based numerical wavetank models

In shallow water, the horizontal velocity component of the flow is dominant as compared to the vertical velocity component and hence we can consider depth-averaged dynamics, i.e. free-surface elevation $\eta(x, y, t)$ and velocity potential $\phi(x, y, t)$ are independent of fluid depth (z-coordinate). Unlike potential-flow equations, the shallow-water equations (SWEs) do not solve the Laplace equation and only solve free-surface boundary conditions, i.e. the kinematics boundary condition

$$\frac{\partial}{\partial t}(\alpha H_0 + \eta) = -\frac{\partial}{\partial x}\left((H_0 + \alpha\eta)\phi_x\right), \tag{2.13}$$

and dynamic boundary condition

$$\frac{\partial \phi}{\partial t}\phi = -\frac{1}{2}\alpha|\frac{\partial \phi}{\partial x}|^2 - g\eta, \tag{2.14}$$

where $\alpha = 0$ yields linear shallow water dynamics and $\alpha = 1$ yields nonlinear shallow water dynamics. SWEs are widely used to study surface waves and for the modelling of floods [21], tsunamis [80], and tides [52] as SWEs are capable of modelling the breaking of waves on the shore, which occurs when the water depth becomes comparable to the wavelength of the waves. To utilise this capability of SWEs, researchers [61, 35] have developed hybrid models of numerical wavetank by coupling either Boussinesq-type equations or potential-flow equations with nonlinear shallow water equations to model the phenomena of wave breaking.

In Fig. 2.5 we have implemented non-dimensionalised linear shallow water equations when

bottom morphology changes, i.e. $H_0 = h(x)$ to demonstrate the variations of water-wave steepness and amplitude with water depth/ sea-bed morphology.

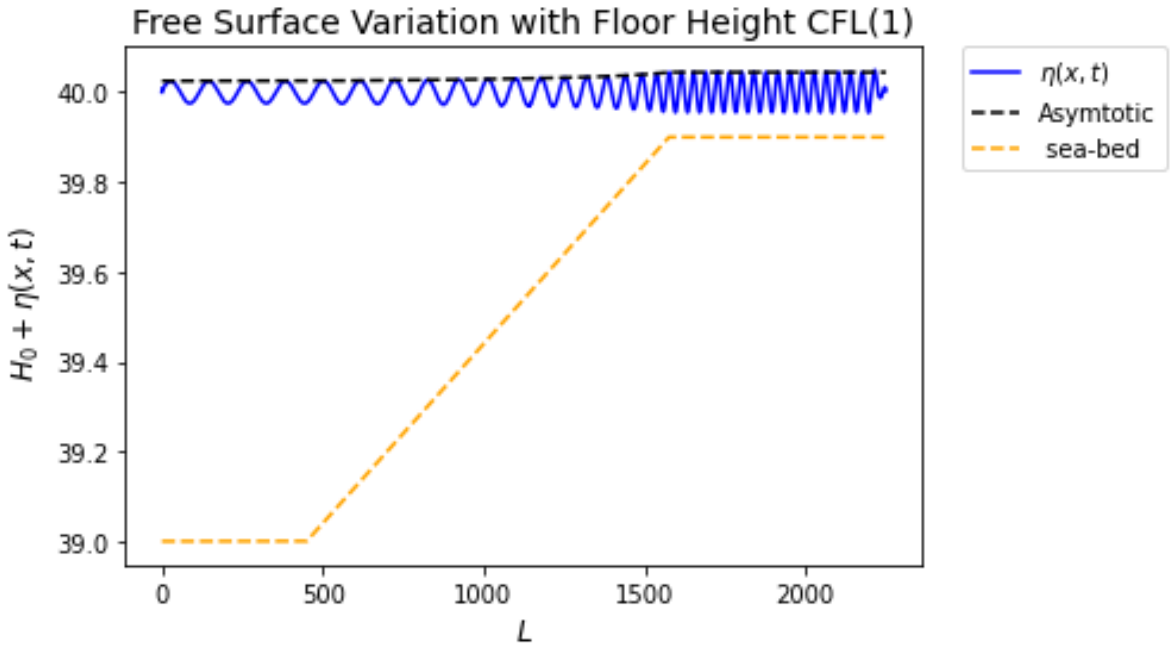


Figure 2.5: Variations of water-wave steepness and amplitude with sea-bed morphology. L in abscissa mean length of the domain, $\eta(x,t)$ is free surface elevation, and H_0 is a variable depending on horizontal x-coordinate.

In this research work, we present a Variational Principle(VP)-based approach of potential flow theory for developing wavetank models to simulate water dynamics. The original aim was to extend Gidel's [34] piston-driven numerical wavetank model to include waveflap wavemaker as it is widely used by the maritime industry. However, deriving and implementing a model of a waveflap-driven numerical wavetank based on Gidel's approach is time-consuming as the waveflap motion is space- and time-dependent which makes equations cumbersome to derive and implement. To address this challenge, we have developed a novel method for implementing VP in the finite-element environment, i.e. Firedrake, and successfully used this approach for the development of a piston-driven numerical wavetank model. This approach is capable of automating the derivation of weak formulations which drastically reduces the time and effort for the code implementation and therefore can simplify the development process of piston- and waveflap-driven wavetank models and coupling of hyperelastic structure's equations with the numerical wavetank to perform FSI analysis.

In this chapter, we explain the development, verification and validation of our novel approach for

implementing the variational principles. We commence by explaining the variational principle for water dynamics in the next section.

2.3 Mathematical modelling of water waves based on variational principle

Luke's variational principle [56] for an inviscid, incompressible and irrotational fluid in a three-dimensional domain is the starting point of developing a numerical wavetank, which is given as

$$0 = \delta \int_0^T \mathcal{L}[\phi, h]\,\mathrm{d}t = \delta \int_0^T \int_{\Omega_H} \int_0^{h(x,y,t)} \partial_t \phi + \frac{1}{2}|\boldsymbol{\nabla}\phi|^2 + g(z - H_0)\,\mathrm{d}z\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t, \tag{2.15}$$

where t is time; x, y and z are the spatial coordinates; g is gravitational acceleration; $\phi(x, y, z, t)$ is the velocity potential, and $h(x, y, t)$ is the total fluid depth which is obtained by adding free-surface elevation $\eta(x, y, t)$ and fluid depth at rest $H_0(x, y)$, i.e., $h(x, y, t) = H_0(x, y) + \eta(x, y, t)$.

The use of a VP emphasizes the conservative structure of nonlinear water-wave dynamics, since energy and mass conservation, as well as the conservation of phase-space volume, are intimately connected with this structure. The preservation of these conservation properties in the numerical discretisation ensures that simulations are compatible with the continuum dynamics, facilitating accuracy and stability. Several groups have therefore based their numerical discretisations directly on space and/or space-time discrete analogues of the underlying VP or associated Hamiltonian dynamics, see, e.g. [13, 33] and references therein for potential-flow dynamics in a vertical cross-section and for Hamiltonian Boussinesq dynamics in [50]. Such an approach based on VPs that generate the dynamics can be extended to include water-wave interactions with floating and flexible structures such as ships and offshore wind-turbine masts. However, the time to development of stand-alone special-purpose numerical implementations of numerical techniques, including those based on variational techniques for such maritime applications, can be substantial. A solution to the variational issue is to use a domain-specific compiler architecture, e.g. for the implementation of finite-element methods, because both finite-element methods and VPs are naturally aligned integral methods. Firedrake is one such "*automated system for the solution of partial differential equations using the finite element method (FEM).*

Firedrake *uses sophisticated code generation*"[1].

2.3.1 Water dynamics based on shallow water equations

Now we will explain the mathematical modelling of a numerical wavetank based on shallow water equations. Consider a rectangular wavetank with a piston wavemaker on the left side, a stationary solid wall on the right side, and water free surface at the top boundary. The wavemaker moves along the x direction up to $x = L_w$. A schematic of the wavetank is demonstrated in Fig. 2.6.

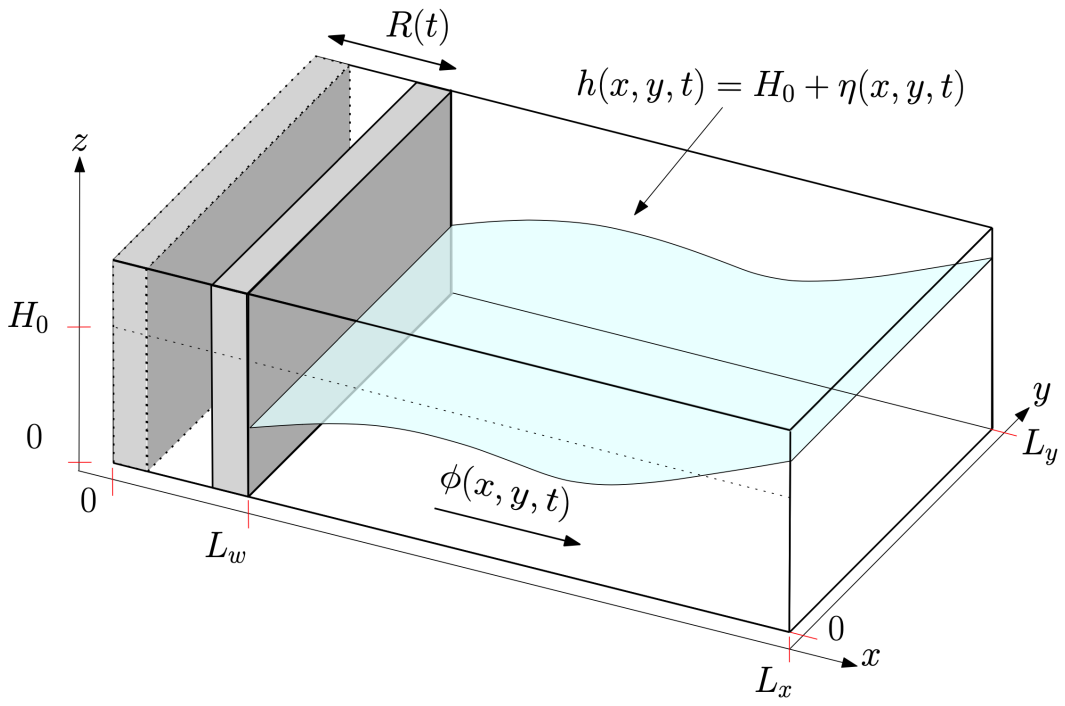


Figure 2.6: Schematic of a three-dimensional rectangular wavetank with piston wavemaker at $x = R(t)$. The piston wavemaker oscillates horizontally in $0 \leq x \leq L_w < L_x$ to generate water waves. The vertical coordinate is z. The free surface resides at $z = h(x, y, t)$ above a flat bottom at $z = 0$; t is time, y the lateral horizontal coordinate and the velocity potential is $\phi(x, y, t)$. The depth at rest is H_0, which defines a perturbation η of the free surface from rest. The lateral extent of the tank is L_y.

In one spatial dimension, the mathematical model is based on the variational principle for linear and nonlinear shallow-water dynamics, which is a simplification of (2.15). Luke's VP for a two-dimensional numerical wavetank driven by a piston wavemaker is given as:

$$0 = \delta \int_0^T \int_{R(t)}^{L_x} \int_0^{h(x,t)} \partial_t \phi + \frac{1}{2}|\partial_x \phi|^2 + g(z - H_0)\, \mathrm{d}z\, \mathrm{d}x\, \mathrm{d}t, \tag{2.16}$$

[1] http://firedrakeproject.org/index.html

where $\phi(x, z, t)$ has dependency on the fluid depth which is denoted by z coordinate. In shallow water, the dynamics become depth-independent, with variables $h(x,t) = H_0 + \eta(x,t)$, and $\phi(x,t)$ as in Fig. 2.6. Therefore, we can integrate along the depth $z = h(x,t)$ as follows

$$0 = \delta \int_0^T \int_{R(t)}^{L_x} h\partial_t\phi + \frac{1}{2}h|\partial_x\phi|^2 + \frac{1}{2}gh^2 - ghH_0 \,\mathrm{d}x \,\mathrm{d}t. \tag{2.17}$$

The boundary condition at the wavemaker is obtained by applying the product rule to the first term of (2.17) as $h\partial_t\phi = \partial_t(h\phi) - \partial_t h\phi$, and evaluating the derivative term $(\partial_t(h\phi))$ by employing the Leibniz integral rule, given as:

$$\frac{\mathrm{d}}{\mathrm{d}x}\int_{a(x)}^{b(x)} f(x,t)\,\mathrm{d}t = f(x,b(x))\frac{\mathrm{d}}{\mathrm{d}x}b(x) - f(x,a(x))\frac{\mathrm{d}}{\mathrm{d}x}a(x) + \int_{a(x)}^{b(x)} \frac{\partial}{\partial x}f(x,t)\mathrm{d}t. \tag{2.18}$$

The byproduct of the steps from (2.16) to (2.18) yields the VP for piston-driven numerical wavetank based on nonlinear shallow water equations, as follows:

$$0 = \delta \int_0^T \int_{R(t)}^{L} \phi\partial_t h - \frac{1}{2}h|\partial_x\phi|^2 - \frac{1}{2}gh^2 + ghH_0 \,\mathrm{d}x \; - \dot{R}h\phi|_{x=R(t)} \,\mathrm{d}t. \tag{2.19}$$

Furthermore, a VP for the linearized shallow water dynamics can be derived from (2.19) by omitting the high-order terms from Taylor expansion of the variables ϕ and h in (2.19). The VP for the linear shallow-water dynamics is given as:

$$0 = \delta \int_0^T \int_0^L \phi\partial_t\eta - \frac{1}{2}H|\boldsymbol{\nabla}\phi|^2 - \frac{1}{2}g\eta^2 \,\mathrm{d}x - H\partial_t R\phi|_{x=0} \,\mathrm{d}t, \tag{2.20}$$

A VP for the combined linear and nonlinear shallow-water dynamics can be written as:

$$\begin{aligned} 0 = \delta \int_0^T \int_{\alpha R(t)}^{L} \phi\partial_t\eta - \frac{1}{2}(H_0 + \alpha\eta)|\partial_x\phi|^2 - \frac{1}{2}g(\alpha H + \eta)^2 + \alpha g H_0(H_0 + \eta)\,\mathrm{d}x \\ - \Big((H_0 + \alpha\eta)\dot{R}\phi\Big)\,|_{x=\alpha R(t)}\,\mathrm{d}t; \end{aligned} \tag{2.21}$$

when $\alpha = 0$ the dynamics resulting from varying (2.21) are linear and, when $\alpha = 1$, the nonlinear dynamics are recovered. Variations of (2.21) are defined, see e.g. [51], by

$$\delta\mathcal{L}_{swe}[\phi,\eta] \equiv \lim_{\epsilon\to 0} \frac{\mathcal{L}_{swe}[\phi + \epsilon\delta\phi, \eta + \epsilon\delta\eta] - \mathcal{L}_{swe}[\phi,\eta]}{\epsilon}$$

with the following functions as variations $\delta\eta = \delta\eta(x,t), \delta\phi = \delta\phi(x,t)$. After taking such varia-

tions of (2.21) with respect to ϕ and η, as follows:

$$= \int_0^T \int_{R(t)}^L \delta\phi\big(\partial_t h + \partial_x(h\partial_x\phi)\big) - \delta h\big(\partial_t\phi + \frac{1}{2}|\partial_x\phi|^2 + g(h - H_0)\big)\,\mathrm{d}x + \big(h\partial_x\phi - h\dot{R}\big)|_{x=R(t)}\delta\phi|_{x=R(t)}\,\mathrm{d}t, \tag{2.22}$$

the following equations of motion and boundary conditions emerge when arbitrariness of variables is used

$$h_t = -\partial_x\left(h\phi_x\right), \tag{2.23a}$$

$$\partial_t\phi = -\frac{1}{2}|\partial_x\phi|^2 - g\eta \tag{2.23b}$$

$$\partial_x\phi = R_t \quad \text{at} \quad x = 0, \quad \text{and} \tag{2.23c}$$

$$\partial_x\phi = 0 \quad \text{at} \quad x = L. \tag{2.23d}$$

These equations respectively comprise the continuity equation, the Bernoulli equation and the boundary conditions. The VP has a time-dependent horizontal domain $R(t) \leq x \leq L$ because the wavemaker position $R(t)$ is a function of time. Both the time-dependent position of the free surface $h(x,t) = H_0 + \eta(x,t)$ and the velocity potential $\phi(x,t)$ are unknown. In order to render static the mesh in the numerical discretisation, the VP (2.19) is transformed into a new, fixed-coordinate system in which, e.g., the new longitudinal coordinate satisfies $\xi \in [0, L]$.

The forward and backward coordinate transformations are

$$x(\xi,\tau) = \begin{cases} R(\tau) + \dfrac{\xi(L_w - R(\tau))}{L_w} & \xi \in [0, L_w] \\ \xi & \xi \in [L_w, L] \end{cases}$$

$$= \frac{\xi L_w + (L_w - \xi)R(\tau)\Theta(L_w - \xi)}{L_w}, \quad \tau = t, \tag{2.24a}$$

$$\xi(x,t) = L_w\frac{(x - R(t))}{L_w - R(t)} - \frac{R(t)(x - L_w)\Theta(x - L_w)}{L_w - R(t)}, \quad t = \tau, \tag{2.24b}$$

where $\Theta(L_w - \xi)$ is the Heaviside function which makes transformation effective in the time-dependent-mesh part of the computational domain i.e. $x \in [R(t), L_w]$. Note that every Heaviside function in these expressions is multiplied by its argument and that there is no ratio of Heaviside functions. That is, it does not matter whether the Heaviside function is 0, 1/2 or 1 when its argument is zero, because this very argument always multiplies the corresponding

Heaviside function. Using the short-hand notations:

$$X(\xi) = \xi - L_w; \tag{2.25a}$$

$$\tilde{R}(\tau) = R\Theta(L_w - \xi); \tag{2.25b}$$

$$\tilde{R}_\tau(\tau) = R_\tau\Theta(L_w - \xi); \tag{2.25c}$$

$$V(\xi, \tau) = L_w - \tilde{R}. \tag{2.25d}$$

The Jacobian of transforms (2.24b) mapping (x, t) to (ξ, τ) is,

$$J \equiv \frac{\partial(x,t)}{\partial(\xi,\tau)} = \begin{bmatrix} \dfrac{L_w - \tilde{R}}{L_w} & \dfrac{\tilde{R}_\tau(L_w - \xi)}{L_w} \\ 0 & 1 \end{bmatrix}, \tag{2.26}$$

where $L_w = O(\lambda)$, with λ is the wavelength generated by the wavemaker. The inverse of J^{-1} is calculated as

$$J^{-1} \equiv \frac{\partial(\xi,\tau)}{\partial(x,t)} = \begin{bmatrix} \dfrac{L_w}{L_w - \tilde{R}} & -\dfrac{\tilde{R}_\tau(L_w - \xi)}{L_w - \tilde{R}} \\ 0 & 1 \end{bmatrix}. \tag{2.27}$$

After using the chain rule, given as

$$\frac{\partial}{\partial t} = \frac{\partial\xi}{\partial t}\frac{\partial}{\partial\xi} + \frac{\partial\tau}{\partial t}\frac{\partial}{\partial\tau}, \qquad \frac{\partial}{\partial x} = \frac{\partial\xi}{\partial x}\frac{\partial}{\partial\xi}, \qquad \mathrm{d}x\,\mathrm{d}t = |J|\,\mathrm{d}\xi\,\mathrm{d}\tau, \tag{2.28}$$

and (2.27), the space and time derivatives are transformed as follows:

$$\partial_t = \frac{\tilde{R}_\tau(\xi - L_w)}{V}\partial_\xi + \partial_\tau, \tag{2.29a}$$

$$\partial_x = \frac{L_w}{V}\partial_\xi, \tag{2.29b}$$

$$\mathrm{d}x\,\mathrm{d}t = |J|\,\mathrm{d}\xi\,\mathrm{d}\tau = \frac{L_w - \tilde{R}}{L_w}\,\mathrm{d}\xi\,\mathrm{d}\tau. \tag{2.29c}$$

After applying the transformations (2.24a) to (2.29c) to (2.19), the VP in the new coordinate system $\{\xi, \tau\}$ reads

$$\begin{aligned} 0 = \delta \int_0^T \int_0^L \Big[-\frac{1}{2}\frac{L_w^2}{V} h(\phi_\xi)^2 + \phi(Vh_\tau + X\tilde{R}_\tau h_\xi) - Vgh(\frac{1}{2}h - H_0) \Big]\,\mathrm{d}\xi \\ - L_w R_\tau \phi h|_{\xi=0}\,\mathrm{d}\tau. \end{aligned} \tag{2.30}$$

The process of domain transformation is graphically illustrated in Fig 2.7.

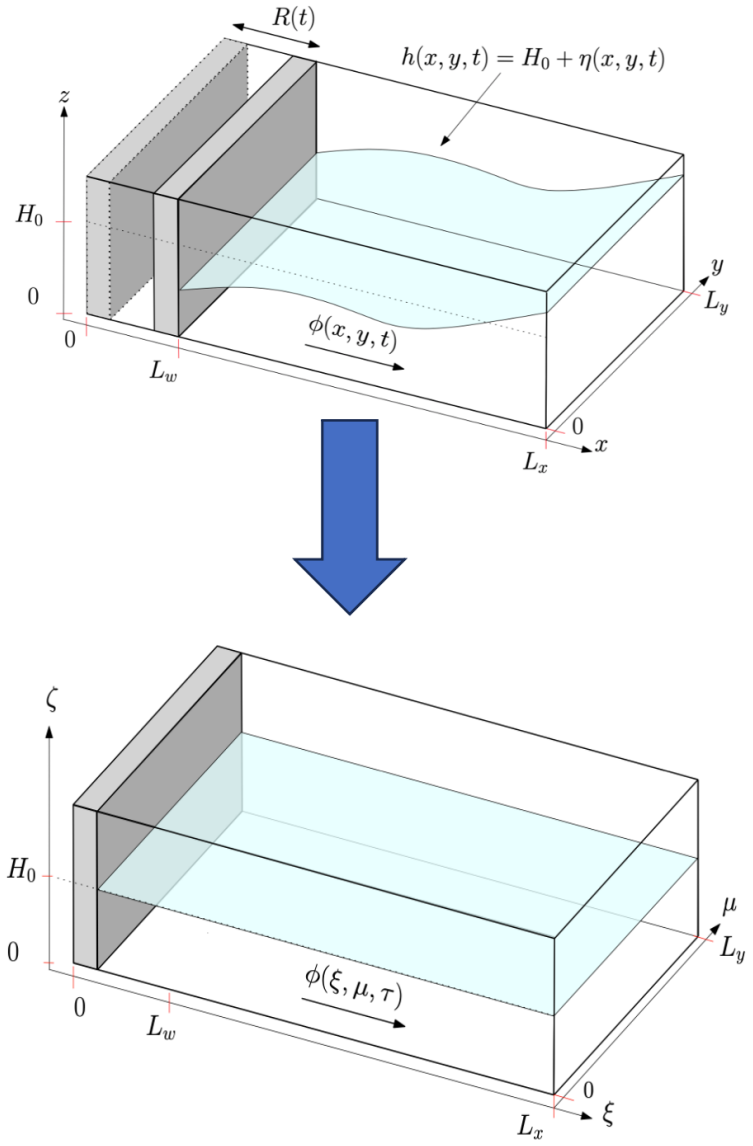


Figure 2.7: The three-dimensional time-dependent computational domain, i.e. (x, y, z), is transformed into a new static computational domain denoted by (ξ, μ, ζ).

Note that the expression (2.30) is similar to the transformed VP stated in equation (8) in [36] after excluding dependencies on the y and z coordinates and taking the velocity potential as a function of x and t only. After taking the variations with respect to $\delta\phi$ and δh, as follows:

$$
\begin{aligned}
0 = \int_0^T \Big(\int_0^L \Big[& \delta\phi \Big(X\tilde{R}_\tau h_\xi + V h_\tau \Big) - \frac{L_w^2}{V} h \partial_\xi \phi \, \partial_\xi(\delta\phi) \\
& - \delta h \Big(\frac{1}{2} \frac{L_w^2}{V} (\phi_\xi)^2 + V\phi_\tau + Vg(h - H_0) \Big) + X\tilde{R}_\tau \phi \partial_\xi(\delta h) \Big] \, \mathrm{d}\xi \\
& - L_w R_\tau h \delta\phi|_{\xi=0} - L_w R_\tau \phi \delta h|_{\xi=0} \Big) \, \mathrm{d}\tau,
\end{aligned} \tag{2.31}
$$

and evaluating the boundary terms by performing integration by parts, the resulting weak formulation is

$$\begin{aligned} 0 = \int_0^T \int_0^L \Big(\Big[X\tilde{R}_\tau h_\xi + V h_\tau - \frac{L_w^2}{V} \partial_\xi (h\phi_\xi) \Big] \delta\phi \\ - \Big[X\tilde{R}_\tau \phi_\xi + V\phi_\tau + \frac{1}{2}\frac{L_w^2}{V}(\phi_\xi)^2 + Vg(h - H_0) \Big] \delta h \Big) \, \mathrm{d}\xi \\ + \Big(\frac{L_w^2}{V}(h\phi_\xi) - L_w R_\tau h \Big) \Big|_{\xi=0} \delta\phi|_{\xi=0} \mathrm{d}\tau. \end{aligned} \tag{2.32}$$

The resulting transformed equations of motion are given as:

$$\delta\phi: \quad X\tilde{R}_\tau h_\xi + V h_\tau - \frac{L_w^2}{V} \partial_\xi (h\phi_\xi) = 0, \tag{2.33a}$$

$$\delta h: \quad X\tilde{R}_\tau \phi_\xi + V\phi_\tau + \frac{1}{2}\frac{L_w^2}{V}(\phi_\xi)^2 + Vg(h - H_0) = 0, \tag{2.33b}$$

$$\delta\phi|_{\xi=0}: \quad \Big(\frac{L_w^2}{V}(h\phi_\xi) - L_w R_\tau h \Big) \Big|_{\xi=0} \delta\phi|_{\xi=0} = 0. \tag{2.33c}$$

Since each variation $\delta\phi$ in the interior, $\delta\eta$ in the interior and $\delta\phi$ at the (transformed) piston-wavemaker location $\xi = 0$ is arbitrary, each bracketed expression before each variation holds pointwise in space and time. Through this observation, the transformed continuity, Bernoulli and wavemaker equations emerge. Note that, in the above variations, end-point contributions arising through integration by parts in time are annihilated because of the temporal-end-point conditions $\delta\eta(\xi, 0) = \delta\eta(\xi, T) = 0$, the former of which logically follows from the specified initial condition $\eta(\xi, 0)$, whose variation is definitively zero; the latter follows by invoking symmetry in time.

Finally, we observe that the presence of wavemakers implies that the integrands in the VPs are functions of variables whose spatio-temporal dependence is both implicit and explicit, the latter reflecting that the resulting dynamics are non-autonomous. In the absence of wavemakers, the total energy $\mathcal{H}$, comprising the integral sum of kinetic and potential energy, is a conserved quantity such that $\mathrm{d}\mathcal{H}/\mathrm{d}t = 0$, and that $\mathcal{H}(t) = \mathcal{H}(t = 0)$ is independent of time. In contrast, when wavemakers are present, $\mathrm{d}\mathcal{H}/\mathrm{d}t \neq 0$ even though the total mass or water volume is conserved in a closed domain.

2.4 Numerical implementation of the VP

The VP derived in §2.3 represents the continuum world whereas computers can only solve the discrete problems, therefore, researchers have developed different numerical methods for the spatial and temporal discretisation of the mathematical equations. To enhance the reader's understanding, we will explain the theoretical background of the spatial and temporal discretisation of the VP in the finite-element-based environment (Firedrake) in which we are implementing the derived VP. First, we describe the spatial discretisation, briefly as it is highly automated intrinsically within Firedrake, followed by a description of the temporal discretisation of the VPs.

2.4.1 Spatial discretisation of VP based on finite element method

In Firedrake, we spatially discretised the equations by using continuous Galerkin finite elements. This section aims to explain the detailed process of spatial discretisation in Firedrake by employing the finite element method. To keep the derivations simple and understandable we will consider a rather simpler case, i.e the VP for linear shallow water equations without piston wavemaker, given as follows

$$0 = \delta \int_0^T \iint_{\Omega_H} \eta \partial_t \phi + \frac{1}{2}|\nabla\phi|^2 + \frac{1}{2}\eta^2 \,\mathrm{d}x\mathrm{d}y\,\mathrm{d}t, \tag{2.34}$$

and then approximate the unknown variables $\eta(x,t)$ and $\phi(x,t)$ by a finite linear combination of basis functions $\varphi_j(\mathbf{x})$. After utilizing the Einstein's summation convention, we have

$$\phi \approx \phi_h(\mathbf{x},t) = \phi_j(t),\varphi_j(\mathbf{x}), \quad \text{and} \quad \eta \approx \eta_h(\mathbf{x},t) = \eta_j(t)\varphi_j(\mathbf{x}). \tag{2.35}$$

Substituting (2.35) into (2.34) yields

$$\begin{aligned} 0 &= \delta \int_0^T \iint_{\Omega_H} \left[(\partial_t(\phi_i\varphi_i)\eta_j\varphi_j) + \frac{1}{2}|\nabla(\phi_i\varphi_i)|^2 + \frac{1}{2}(\eta_j\varphi_j)^2\right] \mathrm{d}x\mathrm{d}y\,\mathrm{d}t \\ &= \delta \int_0^T \Big[\eta_j \frac{\mathrm{d}\phi_i}{\mathrm{d}t} \iint_{\Omega_H} \varphi_i\varphi_j \,\mathrm{d}x\mathrm{d}y + \frac{1}{2}\phi_i\phi_j \iint_{\Omega_H} \nabla\varphi_i \cdot \nabla\varphi_j \,\mathrm{d}x\mathrm{d}y \\ &\quad + \frac{1}{2}\eta_i\eta_j \iint_{\Omega_H} \varphi_i\varphi_j \,\mathrm{d}x\mathrm{d}y\Big]\mathrm{d}t, \end{aligned} \tag{2.36}$$

in which we can use $M_{ij} = \iint_{\Omega_H} \varphi_i \varphi_j \,\mathrm{d}x\mathrm{d}y$ and $S_{ij} = \iint_{\Omega_H} \nabla\varphi_i \cdot \nabla\varphi_j \,\mathrm{d}x\mathrm{d}y$ to obtain the following VP

$$0 = \delta \int_0^T \frac{\mathrm{d}\phi_i}{\mathrm{d}t} M_{ij}\eta_j + \frac{1}{2}\phi_i S_{ij}\phi_j + \frac{1}{2}\eta_i M_{ij}\eta_j \,\mathrm{d}t. \tag{2.37}$$

Applying the variational principle to the first term:

$$\begin{aligned}
\delta \int_0^T \frac{\mathrm{d}\phi_i}{\mathrm{d}t} M_{ij}\eta_j \,\mathrm{d}t &= \lim_{\epsilon_l \to 0} \frac{1}{\epsilon_l} \int_0^T \frac{\mathrm{d}(\phi_i + \epsilon_l \delta\phi_i)}{\mathrm{d}t} M_{ij}(\eta_j + \epsilon_l \delta\eta_j) - \frac{\mathrm{d}\phi_i}{\mathrm{d}t} M_{ij}\eta_j \,\mathrm{d}t \\
&= \lim_{\epsilon_l \to 0} \frac{1}{\epsilon_l} \int_0^T \left(\frac{\mathrm{d}\phi_i}{\mathrm{d}t}\eta_j + \eta_j \epsilon_l \frac{\mathrm{d}\delta\phi_i}{\mathrm{d}t} + \epsilon_l \frac{\mathrm{d}\phi_i}{\mathrm{d}t}\delta\eta_j + \epsilon_l^2 \frac{\mathrm{d}\delta\phi_i}{\mathrm{d}t}\delta\eta_j \right) M_{ij} \\
&\quad - \frac{\mathrm{d}\phi_i}{\mathrm{d}t} M_{ij}\eta_j \,\mathrm{d}t \\
&= \lim_{\epsilon_l \to 0} \int_0^T \left(\frac{\mathrm{d}\delta\phi_i}{\mathrm{d}t} M_{ij}\eta_j + \frac{\mathrm{d}\phi_i}{\mathrm{d}t} M_{ij}\delta\eta_j + \epsilon_l \delta\eta_j M_{ij} \frac{\mathrm{d}\delta\phi_i}{\mathrm{d}t} \right) \mathrm{d}t,
\end{aligned} \tag{2.38}$$

and evaluating the limits and temporal boundary condition

$$\begin{aligned}
&= \int_0^T \left(\frac{\mathrm{d}\delta\phi_i}{\mathrm{d}t} M_{ij}\eta_j + \frac{\mathrm{d}\phi_i}{\mathrm{d}t} M_{ij}\delta\eta_j \right) \mathrm{d}t \\
&= \delta\phi_i M_{ij}\eta_j|_0^T - \int_0^T \left(\delta\phi_i M_{ij} \frac{\mathrm{d}\eta_j}{\mathrm{d}t} - \frac{\mathrm{d}\phi_i}{\mathrm{d}t} M_{ij}\delta\eta_j \right) \mathrm{d}t,
\end{aligned} \tag{2.39}$$

we obtain the final expression, as follows

$$= \int_0^T \left(\frac{\mathrm{d}\phi_i}{\mathrm{d}t}\delta\eta_j - \frac{\mathrm{d}\eta_j}{\mathrm{d}t}\delta\phi_i \right) M_{ij} \,\mathrm{d}t. \tag{2.40}$$

Similarly, the variations of the second term of (2.37) are as follows:

$$\delta \int_0^T \frac{1}{2}\phi_i S_{ij}\phi_j \,\mathrm{d}t = \frac{1}{2} \int_0^T \phi_i S_{ij}\delta\phi_j + \delta\phi_i S_{ij}\phi_j \,\mathrm{d}t. \tag{2.41}$$

and variations of the third term are given as:

$$\delta \int_0^T \frac{1}{2}\eta_i M_{ij}\eta_j \,\mathrm{d}t = \frac{1}{2} \int_0^T \eta_i M_{ij}\delta\eta_j + \delta\eta_i M_{ij}\eta_j \,\mathrm{d}t. \tag{2.42}$$

Combining all three terms by substituting equations (2.40)–(2.42) into (2.37), we have

$$0 = \int_0^T \frac{\mathrm{d}\phi_i}{\mathrm{d}t} M_{ij}\delta\eta_j - \frac{\mathrm{d}\eta_j}{\mathrm{d}t}\delta\phi_i M_{ij} + \frac{1}{2}\phi_i S_{ij}\delta\phi_j + \frac{1}{2}\phi_j S_{ij}\delta\phi_i + \frac{1}{2}\eta_i M_{ij}\delta\eta_j + \frac{1}{2}\eta_j M_{ij}\delta\eta_i \,\mathrm{d}t \tag{2.43}$$

which yields the equations of motion after using the arbitrariness of variations $\delta\eta$ and $\delta\phi$, and

considering M_{ij} and S_{ij} being symmetrical, as follows

$$\begin{aligned}
\delta\eta: \quad & \frac{\mathrm{d}\phi_i}{\mathrm{d}t} M_{ij}\delta\eta_j + \frac{1}{2}\eta_i M_{ij}\delta\eta_j + \frac{1}{2}\delta\eta_i M_{ij}\eta_j = \frac{\mathrm{d}\phi_i}{\mathrm{d}t} M_{ij}\delta\eta_j + \delta\eta_j\eta_i M_{ij} = 0, \\
\Rightarrow \quad & \frac{\mathrm{d}\phi_i}{\mathrm{d}t} M_{ij} + \eta_i M_{ij} = 0 \quad \Leftrightarrow \quad M_{ij}\frac{\mathrm{d}\phi_j}{\mathrm{d}t} + M_{ij}\eta_j = 0; \qquad (2.44) \\
\delta\phi: \quad & -\frac{\mathrm{d}\eta_j}{\mathrm{d}t}\delta\phi_i M_{ij} + \frac{1}{2}\phi_i\delta\phi_j S_{ij} + \frac{1}{2}\delta\phi_i S_{ij}\phi_j = -\delta\phi_i M_{ij}\frac{\mathrm{d}\eta_j}{\mathrm{d}t} + \delta\phi_i S_{ij}\phi_j = 0, \\
\Rightarrow \quad & M_{ij}\frac{\mathrm{d}\eta_j}{\mathrm{d}t} = S_{ij}\phi_j. \qquad (2.45)
\end{aligned}$$

Fortunately, Firedrake sped up the process of implementing the derived equations of motion by eliminating the need to define all the vectors and matrices as it automatically defines all the vectors and matrices once the user provides the weak formulations and defines the mesh and function spaces.

2.4.2 Time discretisation

The conservative nature of the dynamics requires that special time integrators are applied. Since the spatially-discrete dynamics remains conservative given its generation from a spatially-discrete VP without any numerical damping added, that requirement to use special integrators for stiff problems remains valid. We will therefore use so-called geometric time integrators [37, 33, 10] such as the first-order symplectic-Euler (SE) and second-order Störmer-Verlet (SV) time-integration schemes. These are explicit for linear wave dynamics but semi-implicit with a timestep restriction for nonlinear wave dynamics. Alternatively, a fully implicit second-order geometric mid-point scheme has been under consideration. The latter equates to the Crank-Nicolson time-stepping scheme for linear wave dynamics.

2.4.3 Shallow-water equations with piston wavemaker

For linear shallow-water dynamics, two formulations are derived to facilitate a comparison between a numerical implementation with classical weak formulations and one using the time-discrete VPs provided below. The time-discrete weak formulations are derived by multiplying the time-discrete equations of motion found in (2.23) by respective test functions, i.e., by $\delta\phi^n$ and $\delta\eta^{n+1}$, and integration (by parts) in space. A combined forward-Euler timestep for updating η and backward-Euler timestep for updating ϕ, forming a so-called linear SE scheme, is employed.

The resulting time-discrete weak formulations are as follows

$$\int_0^L \delta\phi^n \frac{(\eta^{n+1}-\eta^n)}{\Delta t} - H_0 \boldsymbol{\nabla}\phi^n \cdot \boldsymbol{\nabla}\delta\phi^n \,\mathrm{d}x - H_0 R_t^n \delta\phi^n|_{x=0} = 0, \tag{2.46a}$$

$$\int_0^L \Big(\frac{(\phi^{n+1}-\phi^n)}{\Delta t} + g\eta^{n+1}\Big)\,\delta\eta^{n+1}\,\mathrm{d}x = 0. \tag{2.46b}$$

The corresponding combined linear and nonlinear time-discrete weak formulations are

$$\begin{aligned}\int_0^L V^n \frac{\eta^{n+1}-\eta^n}{\Delta t}\delta\phi^n - \frac{L_w^2}{V^n}(H_0+\alpha\eta^{n+1})\phi_\xi^n\,\partial_\xi(\delta\phi^n)\,\mathrm{d}\xi = \\ \int_0^L -\alpha X \tilde{R}_\tau^n \eta_\xi^{n+1}\delta\phi^n + L_w R_\tau^n (H_0+\alpha\eta^{n+1})\delta\phi^n\Big|_{\xi=0},\end{aligned} \tag{2.47a}$$

$$\begin{aligned}\int_0^L \Big(\frac{V^{n+1}\phi^{n+1}-V^n\phi^n}{\Delta t} + gV^n\eta^{n+1}\Big)\,\delta\eta^{n+1}\,\mathrm{d}\xi = \\ \int_0^L \alpha X \tilde{R}_\tau^n \phi^n \partial_\xi(\delta\eta^{n+1}) - \frac{1}{2}\alpha\frac{L_w^2}{V^n}(\phi_\xi^n)^2\delta\eta^{n+1}\,\mathrm{d}\xi \\ - \alpha L_w R_\tau^n \phi^n \delta\eta^{n+1}\Big|_{\xi=0},\end{aligned} \tag{2.47b}$$

which coincide with (2.46), for $\alpha = 0$ and $V = L_w$. Note that, in the nonlinear case, the conjugate variable to η is the combination $V\phi$. The geometric factor V can be evaluated at (indexed) times $n, n+1/2$ or $n+1$ for this first-order scheme; here we evaluated at n, see [10, 33]. The corresponding time-discrete analogue of the continuum VP (2.30) for linear/nonlinear shallow-water equations reads:

$$\begin{aligned}0 = \delta\int_0^L \Big[\phi^n\Big(\alpha X\tilde{R}_\tau\partial_\xi\eta^{n+1} + V^n\frac{\eta^{n+1}-\eta^n}{\Delta t}\Big) - V^{n+1}\phi^{n+1}\frac{\eta^{n+1}}{\Delta t} \\ - \frac{1}{2}\frac{L_w^2}{V^n}(H_0+\alpha\eta^{n+1})|\phi_\xi^n|^2 - \frac{1}{2}gV^n\eta^{n+1})^2\Big]\,\mathrm{d}\xi - L_w R^n{}_\tau (H_0+\alpha\eta^{n+1})\phi^n|_{\xi=0},\end{aligned} \tag{2.48}$$

which is directly implemented into Firedrake. The above time-discrete VP is a simplified version of a systematic derivation, following work by [10, 32]. Employing the Firedrake (partial) "functional" derivative command derivative with respect to $\delta\phi^n, \delta\eta^{n+1}$ automatically generates the weak formulations to be solved numerically (see §17.5.1 in [1] and §6.4 in [2]). While the above variational approach is straightforward for shallow-water dynamics, for the (transformed) potential-flow dynamics of coupled nonlinear water-wave dynamics and dynamics of hyperelastic structures, described by coupled VPs, this automated procedure using time-discrete VPs is more effective than manual derivation. Based on a similar systematic derivation, a time-discrete

VP leading to a second-order SV time discretisation reads:

$$
\begin{aligned}
0 = \delta \int_0^L \Big[& -2\eta^n \Big(\frac{V^{n+1/2}\phi^{n+1/2} - V^n\phi^n}{\Delta t} \Big) - 2\eta^{n+1} \Big(\frac{V^{n+1}\phi^{n+1} - V^{n+1/2}\phi^{n+1/2}}{\Delta t} \Big) \\
& + \alpha X \tilde{R}_\tau^{n+1/2} \Big(\partial_\xi \eta^{n+1} + \partial_\xi \eta^n \Big) \phi^{n+1/2} \\
& - \frac{1}{2} \frac{L_w^2}{V^{n+1/2}} (\phi_\xi^{n+1/2})^2 \Big((H_0 + \alpha\eta^{n+1}) + (H_0 + \alpha\eta^n) \Big) \\
& - \frac{1}{2} g V^{n+1/2} \Big((\eta^{n+1})^2 + (\eta^n)^2 \Big) \Big] \, \mathrm{d}\xi \\
& - L_w R_\tau^{n+1/2} \phi^{n+1/2} \Big((H_0 + \alpha\eta^{n+1}) + (H_0 + \alpha\eta^n) \Big) \Big|_{\xi=0}.
\end{aligned} \tag{2.49}
$$

As illustration, for the case $\alpha = 0$ and $V = L_w$, variations of (2.49) with respect to $\delta\eta^n$, $\delta\phi^{n+1/2}$ and $\delta\eta^{n+1}$ yield the time-discrete weak formulations of linear shallow-water dynamics based on a second-order Störmer-Verlet time-stepping scheme, as follows

$$
\int_0^L -2 \Big(\frac{\phi^{n+1/2} - \phi^n}{\Delta t} \Big) \delta\eta^n - g\eta^n \delta\eta^n \, \mathrm{d}x = 0, \tag{2.50a}
$$

$$
\begin{aligned}
\int_0^L & 2 \Big(\frac{\eta^{n+1} - \eta^n}{\Delta t} \Big) \delta\phi^{n+1/2} - 2H_0 \partial_\xi \phi^{n+1/2} \partial_\xi (\delta\phi^{n+1/2}) \, \mathrm{d}x \\
& - 2H_0 R_t^{n+1/2} \delta\phi^{n+1/2}|_{x=0} = 0,
\end{aligned} \tag{2.50b}
$$

$$
\int_0^L -2 \Big(\frac{\phi^{n+1} - \phi^{n+1/2}}{\Delta t} \Big) \delta\eta^{n+1} - g\eta^{n+1} \delta\eta^{n+1} \, \mathrm{d}x = 0. \tag{2.50c}
$$

In *Firedrake*, however, we do not implement these weak formulations explicitly but instead generate the weak forms automatically from the time-discrete VP.

2.4.4 Timestep criterion

The SE and SV schemes are conditionally stable. A linear stability criterion can be based on analysis of the linear harmonic oscillator, with frequency ω_{max} [37], as follows

$$
\Delta t = CFL(2/\omega_{max}), \tag{2.51}
$$

where for the Courant-Friedrichs-Lewy (CFL)-number we take $CFL \leq 1$. For linear wave dynamics, ω_{max} is the maximum wave frequency, which can be estimated using the linear, potential-flow dispersion relation

$$
\omega_{max} \approx \sqrt{gk \tanh(kH_0)} \leq \sqrt{gH_0} k \tag{2.52}
$$

the upper bound holding in the shallow-water limit. The maximum wavenumber $k = 2\pi/\Delta x$ depends on an estimate Δx of the minimum mesh size. For nonlinear wave dynamics, the CFL (again satisfying $CFL \leq 1$) needs to be estimated and tested. Codes are anticipated to become unreliable when waves become too steep; this can be circumvented by using parametrised wave-breaking schemes such as the one in [81]. Wave-breaking parametrisation schemes have not been considered here.

2.5 Verification and validation

2.5.1 Comparison of linear shallow water equation with exact solution

The VP for linear shallow-water dynamics is given as:

$$0 = \delta \int_0^T \int_0^L \phi \partial_t \eta - \frac{1}{2} H |\boldsymbol{\nabla} \phi|^2 - \frac{1}{2} g \eta^2 \, \mathrm{d}x \, \mathrm{d}t. \tag{2.53}$$

The time-discrete variational principle (VP) based on the Symplectic Euler scheme [37] for the linear potential-flow shallow-water dynamics reads

$$0 = \delta \int_0^L \phi^n \frac{(\eta^{n+1} - \eta^n)}{\Delta t} - \phi^{n+1} \frac{\eta^{n+1}}{\Delta t} - \frac{1}{2} H |\boldsymbol{\nabla} \phi^n|^2 - \frac{1}{2} g (\eta^{n+1})^2 \, \mathrm{d}x. \tag{2.54}$$

Variations of (2.54) with respect to $\delta\phi^n$ and $\delta\eta^{n+1}$ yield the Symplectic-Euler time-discrete weak formulations, as follows

$$\int_0^L \delta\phi^n \frac{(\eta^{n+1} - \eta^n)}{\Delta t} - H \boldsymbol{\nabla} \phi^n \cdot \boldsymbol{\nabla} \delta\phi^n \, \mathrm{d}x = 0 \quad \text{and} \tag{2.55}$$

$$\int_0^L \Big(\frac{(\phi^{n+1} - \phi^n)}{\Delta t} + g \eta^{n+1} \Big) \delta\eta^{n+1} \, \mathrm{d}x = 0. \tag{2.56}$$

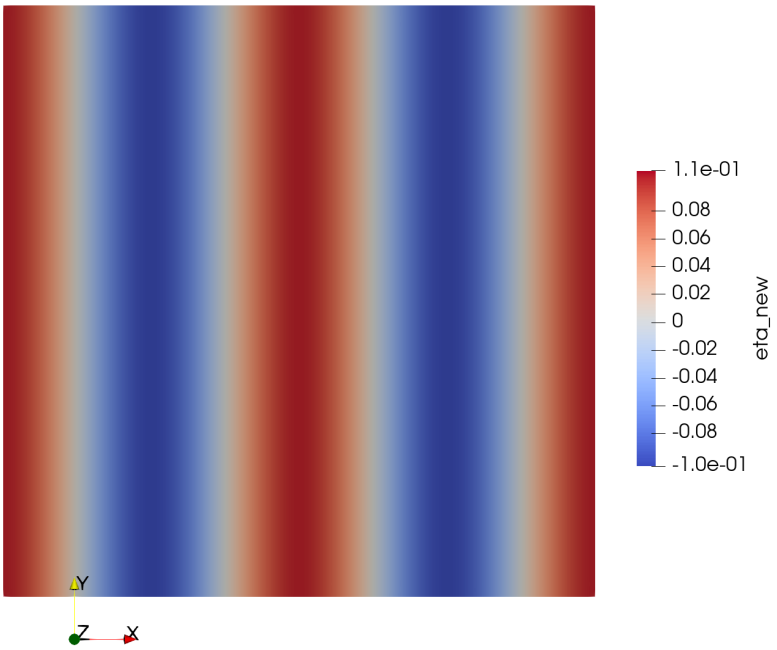
The exact standing wave solution for $\phi(x,t)$ and $\eta(x,t)$, for the linearised shallow water equations with piston wavemaker, is derived in Appendix A, which is given as

$$\phi(x,t) = \frac{g}{\omega} \cos k_1 x \, (-A \sin \omega t + B \cos \omega t), \tag{2.57}$$

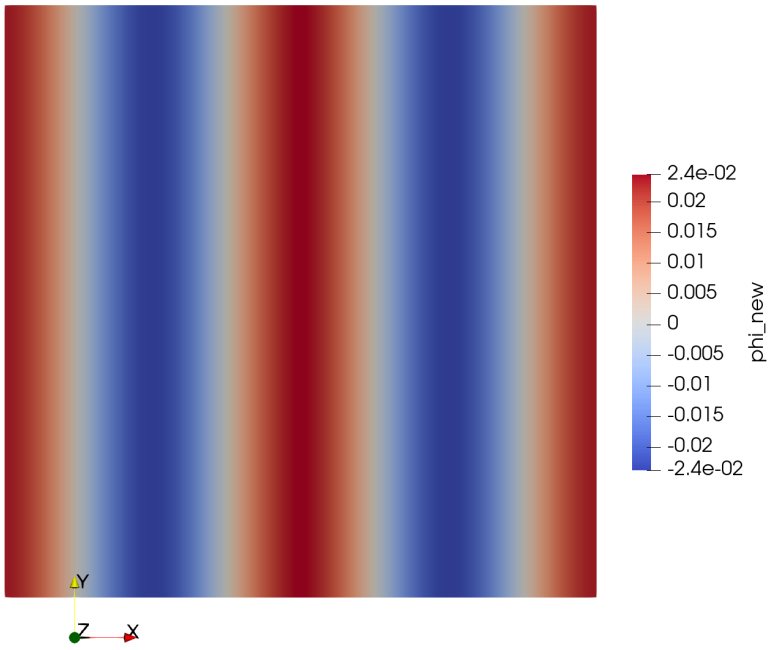
$$\eta(x,t) = \cos k_1 x \, (A \cos \omega t + B \sin \omega t). \tag{2.58}$$

The initial conditions are the same as the exact solutions, given in (2.57) and (2.58), evaluated at time $t = 0$. In this case, a unit square computational domain, i.e. $L_x \times L_y = 1 \times 1$, is spatially

discretised into 99×99 square finite elements. The initial conditions for ϕ and η at time $t = 0$ are shown in Fig. 2.8.



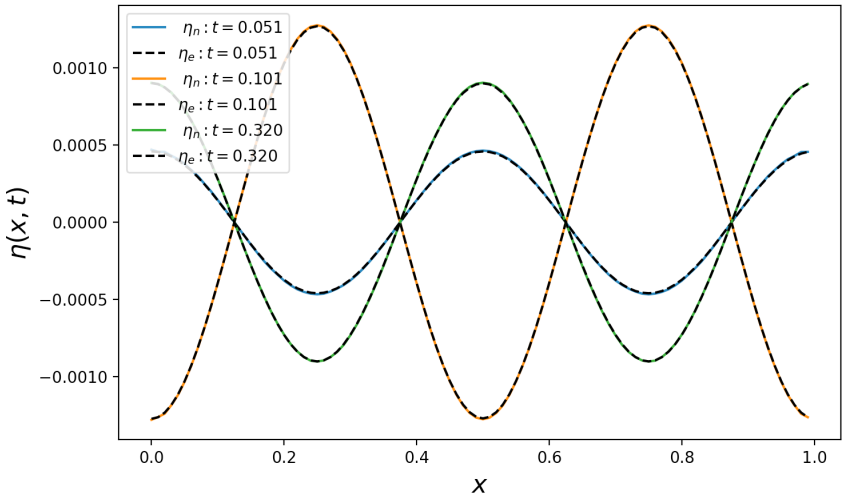
(a) Initial condition for η (2.58) at $t = 0$.



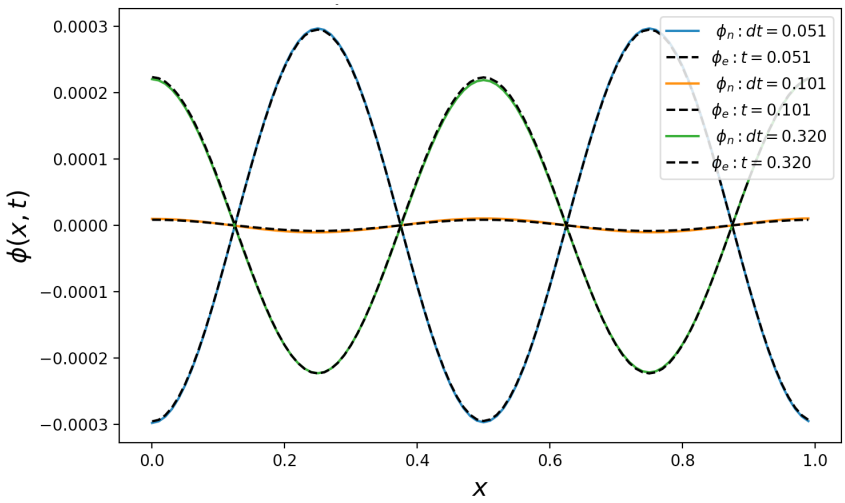
(b) Initial condition for ϕ (2.57) at $t = 0$.

Figure 2.8: Plots of the initial conditions for η and ϕ, given in (2.57) and (2.58), evaluated at time $t = 0$ in Firedrake.

The comparison of the numerical results obtained for $\phi(x, t)$ and $\eta(x, t)$, from the novel approach for implementing the VP, with the exact solution is shown in Fig. 2.9. The final time of the simulation is twice the time period of wave i.e. $T_p = 2\pi/\omega$.



(a) Free surface elevation η when $R_t = 0$.



(b) Velocity potential ϕ when $R_t = 0$.

Figure 2.9: Comparison of numerical and exact solution of η and ϕ at different time steps. Numerical results are shown by solid lines while the dashed black lines show the exact solution.

The comparison shown in Fig. 2.9 of the numerical solution, i.e. shown by solid lines, with the exact solution, i.e. shown by dashed black lines, depicts that the numerical results agree with the exact solution. After obtaining satisfactory results from the novel approach, we divide our study into two cases to compare the results from the novel approach of implementing the VP (Case 1) with the classical approach of implementing the weak formulations (Case 2). This comparison aims to prove that the numerical results from case 1 are consistent with case 2, and both approaches are equivalent mathematically and numerically. Therefore we performed the comparison by plotting the L_∞ norm at each time step t of the simulation, given as

$$|L_\infty|_t = max|\varepsilon_{c1} - \varepsilon_{c2}|_t, \quad 0 \leq t \leq Tp, \tag{2.59}$$

where ε_{c1} represents the numerical results from Case 1, ε_{c2} represents the numerical results from Case 2, and T_p is the end time of the simulation. Fig. 2.10 shows the L_∞ norm of cases 1 and 2 which proves that the approaches used in each are (numerically) equivalent. Notably, the advantage of using the approach used in case 1 is that it reduces the effort required when implementing software, thereby decreasing the likelihood of introducing human error.

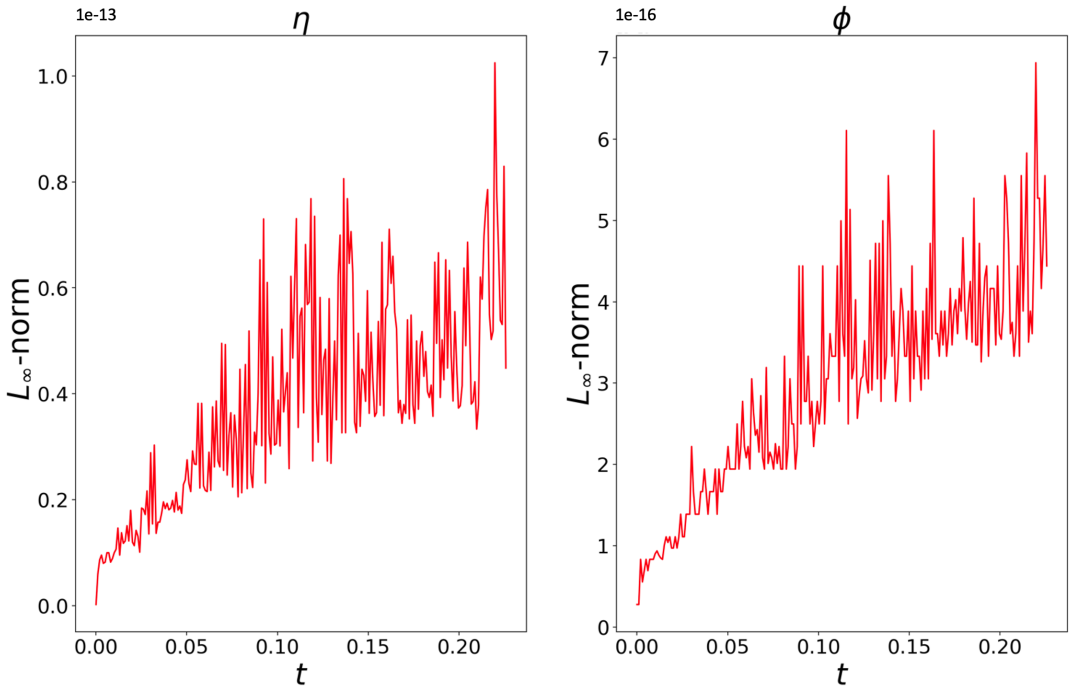


Figure 2.10: Time evolution of L_∞ norms of the difference between numerical simulations of solutions η (left) and ϕ (right), for the linear shallow-water equations, for Case 1 and Case 2. Norms are taken over the full solution domain and each subgraph confirms that the results of the two cases are, as expected, equivalent to within machine precision. The positive mean slope in both plots reflects error accumulation with the evolving number of calculations. Vertical axes display multiples of 10^{-13} and 10^{-16} in left- and right-hand plots respectively. A CG1 spatial discretisation with 200 elements has been used.

2.5.2 Comparison of two implementation approaches for nonlinear shallow water dynamics

In this section, we solve the VP for nonlinear shallow water dynamics with the aim of comparing the two approaches, i.e. the novel- (time-discrete VP) and the classical-(time-discrete weak formulations) approach. Furthermore, we will use two different time-integration schemes, i.e. the first-order Symplectic-Euler (SE) and second-order Störmer-Verlet (SV), for both approaches. We start the comparison by defining a two-dimensional horizontal computation domain of 140m $\times$ 40m which is discretised by 200 elements in x-direction and 1 element in y-direction. The

number of elements/element size is selected after performing a mesh convergence study which is not shown here. Note that the computational domain, shown in Fig. 2.11, and equations are non-dimensionalized, however, we have used units to give an idea of the order of magnitude.

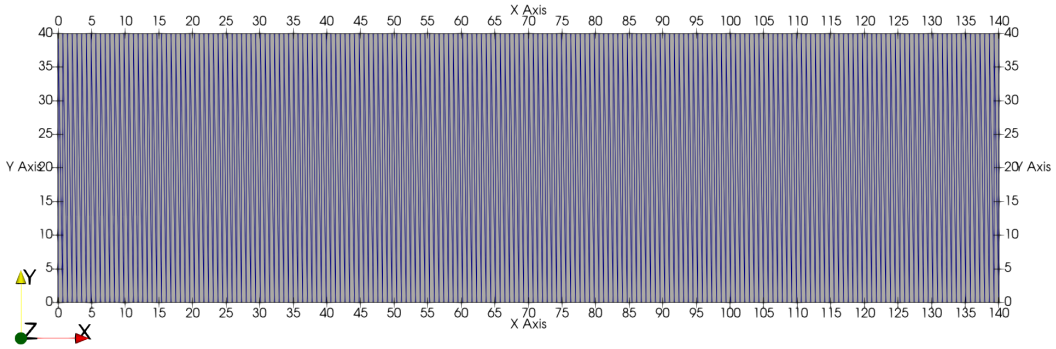
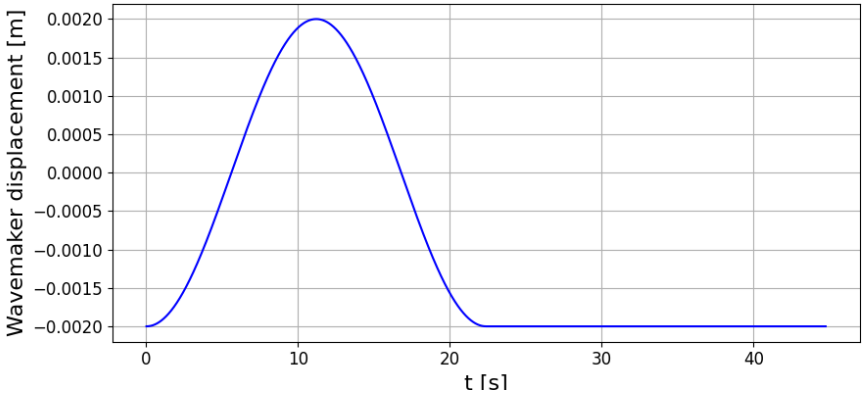
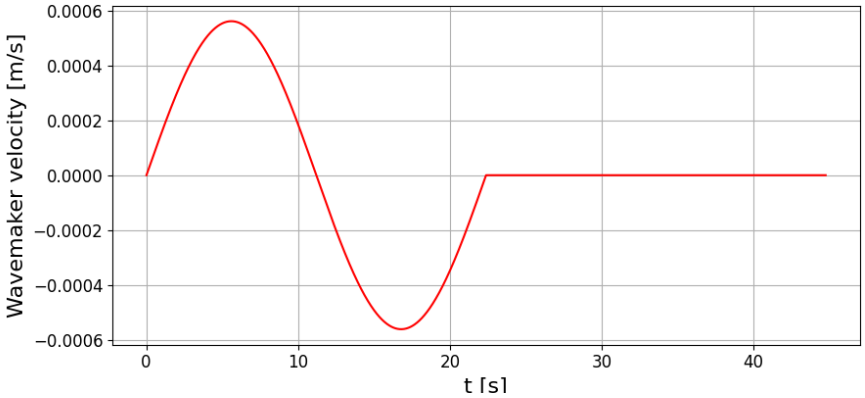


Figure 2.11: The two-dimensional spatially-discretised computational domain for solving the VP of shallow water equations is shown.



(a) The plot of the wavemaker displacement.



(b) The plot of wavemaker velocity.

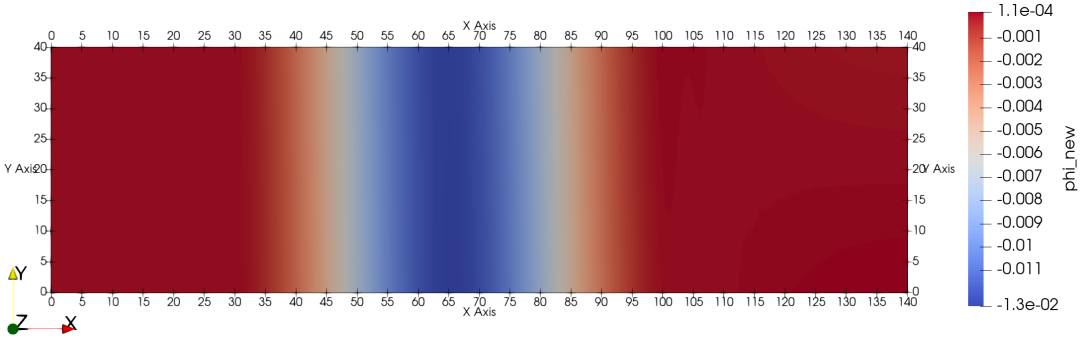
Figure 2.12: The evolution of piston wavemaker displacement and velocity is plotted for the complete computational time.

Initially, free surface elevation η and velocity potential ϕ are set to zero. Note that $\eta = 0$, however the total fluid depth $h = H_0 + \eta = 1\text{m}$ because still water level H_0 is 1m. The wavemaker motion chosen and controllable parameters are:

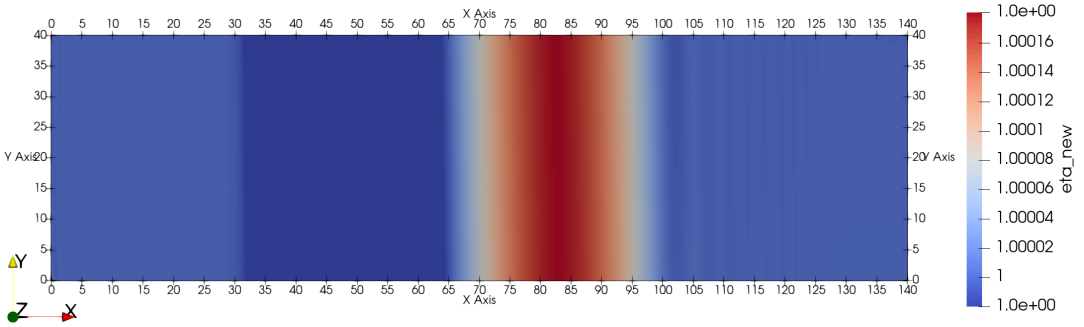
$$R(t) = \begin{cases} \gamma \cos(\sigma t) & 0 \leq t \leq T_p \\ 0 & t > T_p \end{cases}, \tag{2.60}$$

$$\sigma = \sqrt{gH_0}k = \sqrt{gH_0}2\pi/\Lambda, \quad \Lambda = 70\text{m}, \quad T_p = 2\pi/\sigma, \quad \gamma = 0.002\text{m}.$$

Here, σ is the wavemaker angular velocity, k is the wavenumber given as $2\pi/\Lambda$, and γ is the amplitude of the wavemaker displacement. The total computational time is two times the wave time period, i.e. $2T_p = 44.68$ seconds. We turn the wavemaker on for $1T_p$ and keep it off for another $1T_p$, as shown in Fig. 2.12. When the wavemaker moves a wave of $1T_p$ is generated and when the wavemaker stops the wave generated by the wavemaker continues travelling to the end of the numerical wavetank. The length of the wavemaker is chosen as twice the wavelength which is 140m. At this stage, we have not included any wave absorption mechanism in the wavetank and hence to avoid wave reflections we simulate to a time of $2Tp$. The evolution of wave through the computational domain is shown in Fig. 2.13.



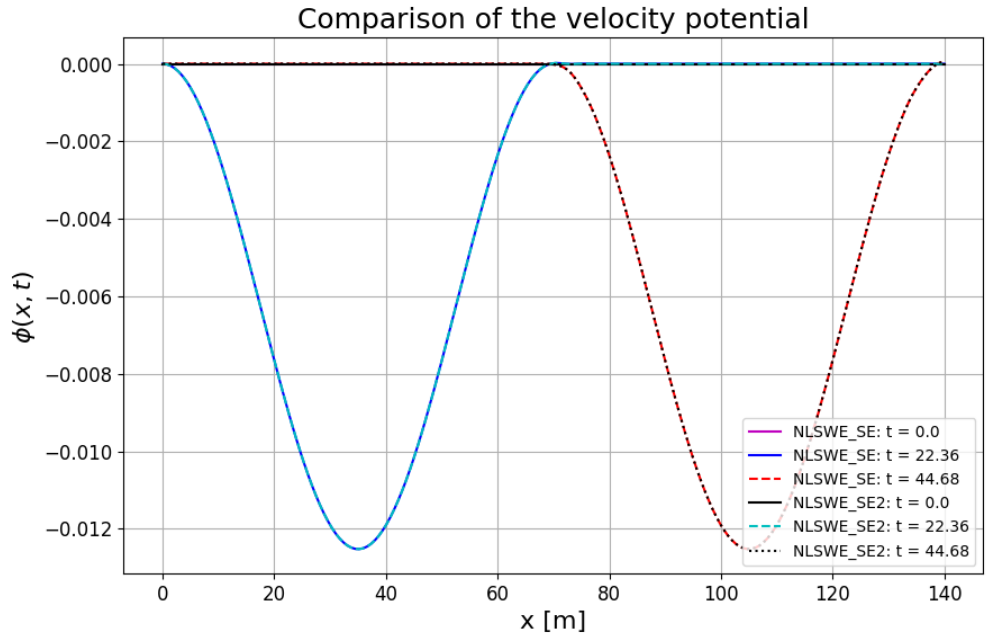
(a) Evolution of velocity potential (ϕ) along the computational domain.



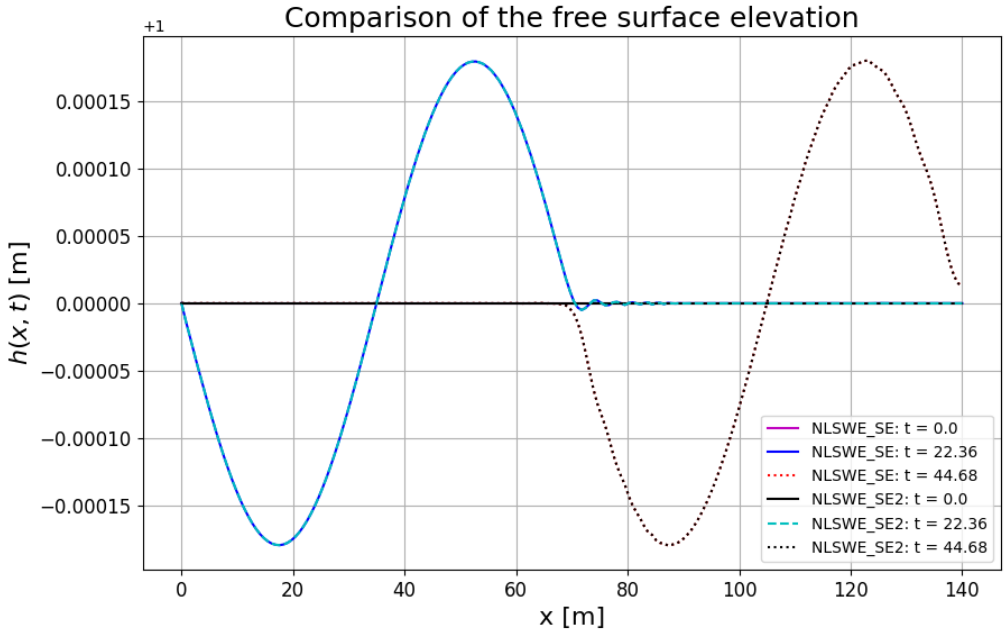
(b) Evolution of free-surface elevation ($h = H_0 + \eta$) along the computational domain.

Figure 2.13: The evolution of the wave through the computational domain is shown at one instant of the simulation.

At first, we test the two approaches when time discretisation is based on the first-order Symplectic-Euler scheme. A slice along the horizontal x-axis of the computational domain is taken to plot the evolution of velocity potential and free-surface elevation through the computational domain at different time steps, shown in Fig. 2.14.



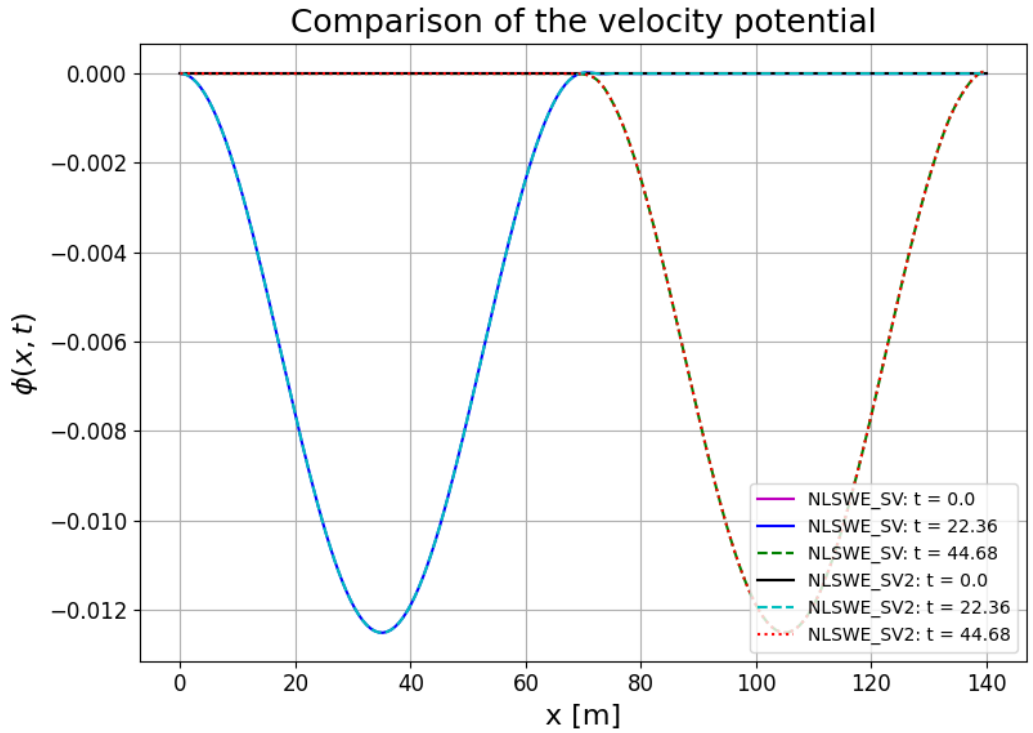
(a) Evolution of velocity potential (ϕ) at different time steps.



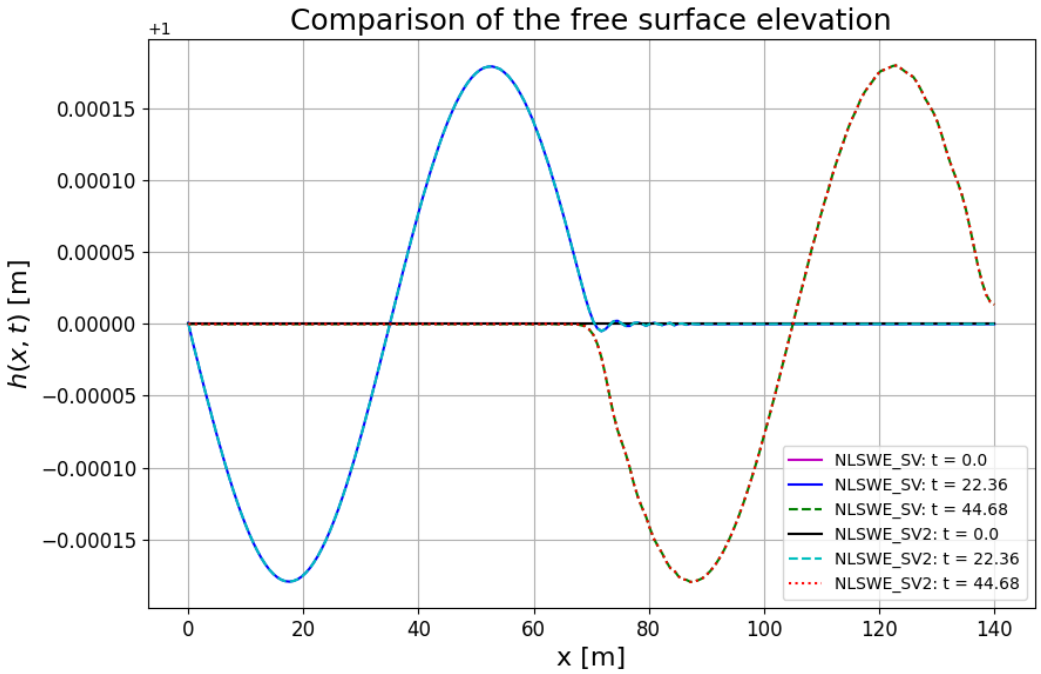
(b) Evolution of free-surface elevation ($h = H_0 + \eta$) at different time steps.

Figure 2.14: Comparison of novel and classical approaches for implementing the VP for nonlinear shallow water equations by using the first-order symplectic-Euler scheme is shown.

Now, we will depict a similar comparison by using the second-order Störmer-Verlet scheme in Fig. 2.15.



(a) Evolution of velocity potential (ϕ) at different time steps.



(b) Evolution of free-surface elevation ($h = H_0 + \eta$) at different time steps.

Figure 2.15: Comparison of novel- and classical-approach for implementing the the VP for nonlinear shallow water equations by using the second-order Störmer-Verlet scheme is shown.

The visual analysis proves that both approaches are mathematically and numerically equivalent and hence the novel approach can replace the classical approach for the development of more sophisticated wavetank models. We have performed a similar comparison of two approaches based on symplectic-Euler and Störmer-Verlet time integration schemes for linear shallow water dynamics and monitored the evolution of energy, which is shown in Fig. 2.16.

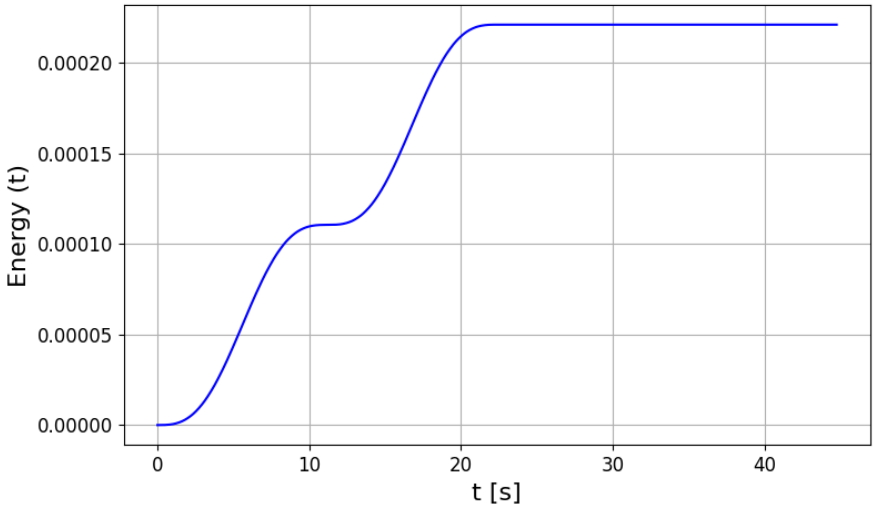
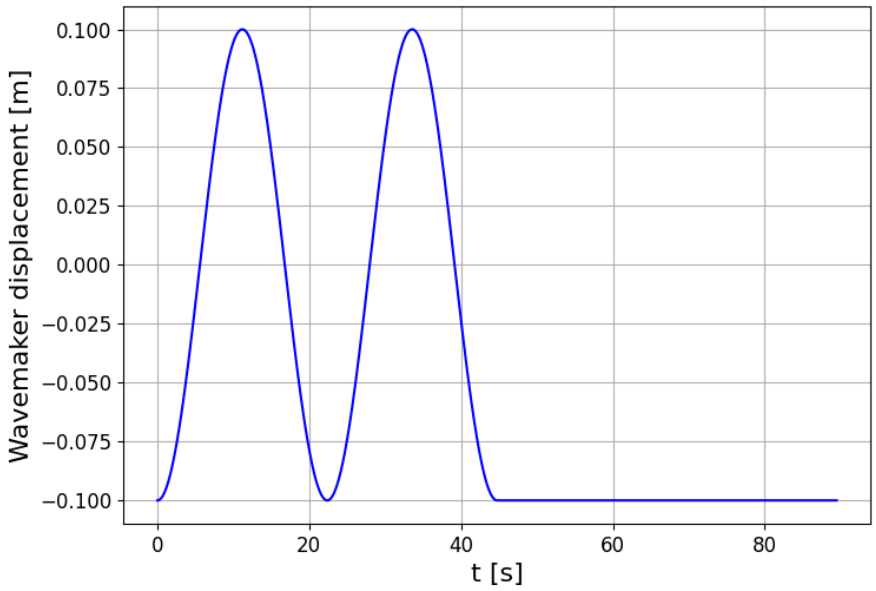


Figure 2.16: Evolution of total energy of the system.

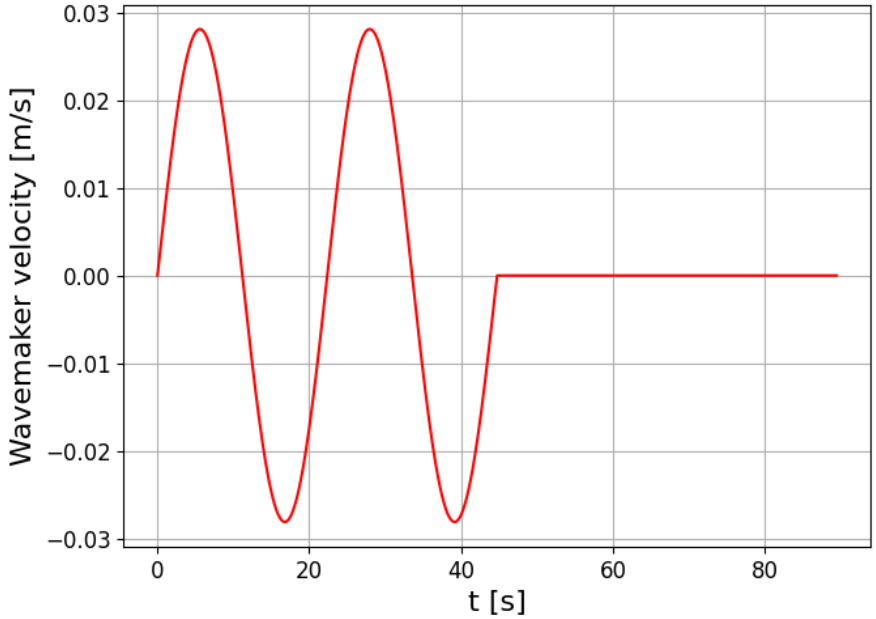
We noticed that the behaviour of energy is in line with real-world physics. When the piston wavemaker is turned on, the system's total energy increases and when the wavemaker is turned off after $1T_p$ the energy stays at the constant level, which shows the system's energy is conserved. After proofing that the results from the traditional and novel approaches agree well, i.e. within machine precision. We can perform more test cases to exhibit the capabilities of the numerical wavetank.

2.5.3 Test case: high amplitude waves

In this test case, we aim to generate waves of amplitude 0.1 m in the wavetank which is 1 m deep. Note that the comparison study explained in the previous study has waves of 1.5 mm amplitude. Therefore, we test the wavetank by increasing the wave amplitude and allowing them to reflect from the fixed wall. The wave period T_p is 22.36 seconds and the wavelength is 70 m. The total simulation time is $4T_p$; the wavemaker is turned off after $2T_p$ and the waves generated by the wavemaker are allowed to reflect. Figs. 2.17a and 2.17b show the displacement and velocity of the wavemaker, respectively.



(a) The plot of the wavemaker displacement.



(b) The plot of wavemaker velocity.

Figure 2.17: The evolution of piston wavemaker displacement and velocity is plotted for the complete computational time.

Figs. 2.18 and 2.19 show the generated and reflected wave heights and velocity potentials, respectively.

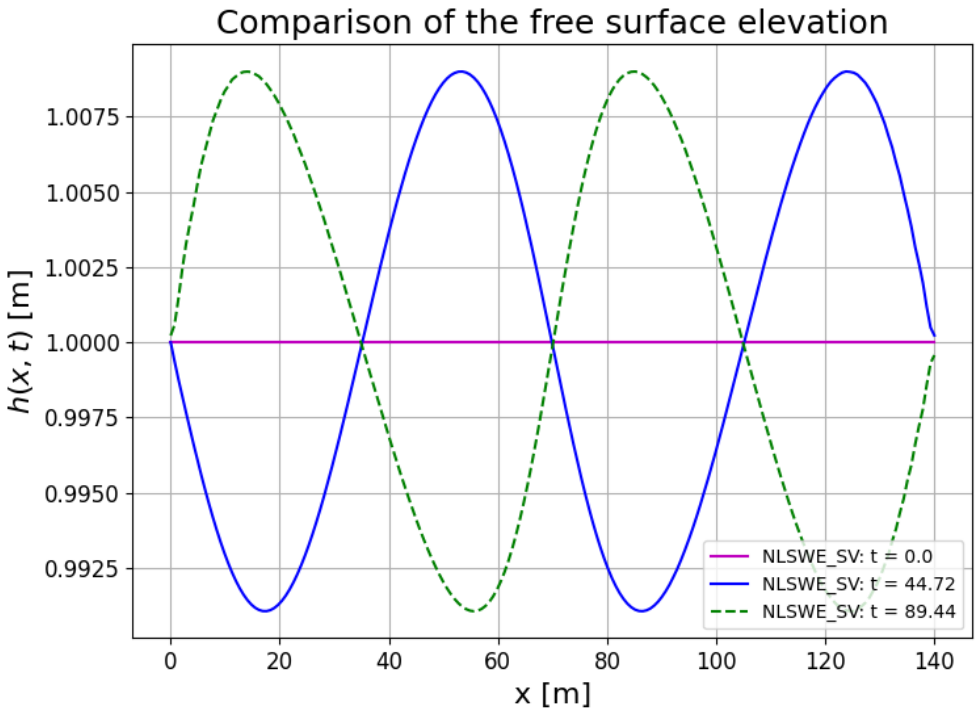


Figure 2.18: Evolution of free-surface elevation ($h = H_0 + \eta$) at different time steps.

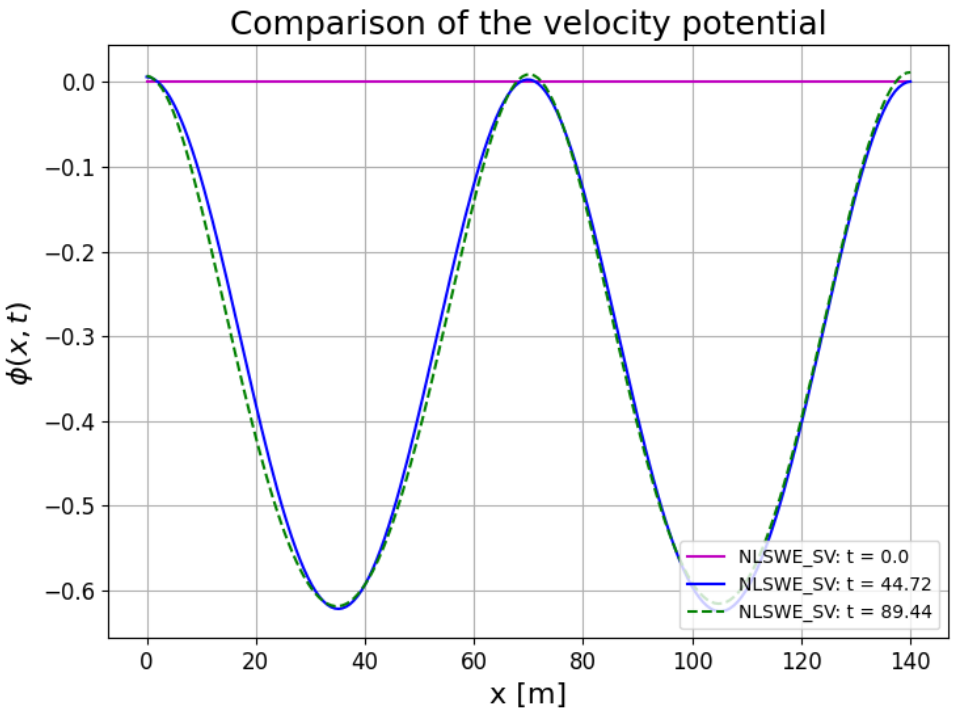


Figure 2.19: Evolution of velocity potential at different time steps.

2.6 Industrial applications of SWE-based numerical wavetank model

Although the numerical wavetank based on shallow water equations is a stepping stone for developing a numerical wavetank based on potential-flow equations by employing the novel approach for the implementation of variational principles, it does have industrial applications. In this section, we will discuss some of the industrial applications of the numerical wavetank based on shallow water equations, which are as follows:

- Modelling of the waves at wavetank beach. The shallow water equations can be coupled with numerical wavetanks based on potential flow equations to model the waves at shallow beaches for wave absorption.
- Ship manoeuvring in shallow waters. The SWE-based numerical wavetank can simulate the flow around the ships and determine the manoeuvring dynamics of ships in shallow water, i.e. near port and inland navigation. Such simulations may eventually lead to development of the automatically manoeuvred ships.
- Floating solar panels. The idea of utilising the surface of shallow water bodies like lakes and ponds for the installation of floating solar panels is getting attention in the maritime industry. Our model of the SWE-based numerical wavetank can assist in the designing process by predicting the wave loads on the floating panels.
- Hydrodynamics loads on offshore structures. Most of the fixed-bottom offshore turbines are installed in shallow waters. Coupled with hyperelastic beam equations, our model can determine the hydrodynamic loads on the turbine's mast by performing fluid-structure analysis. Salwa et al. [71] have developed such a model by coupling the variational principle of the hyperelastic beam model with the variational principle of potential flow equations. We have derived the time-discrete variational principle for hyperelastic equations that can be coupled with SWE-based numerical wavetanks. See §3.5.1 of this report for more details on the derivations and initial results from the time-discrete variational principle for hyperelastic equations.

2.7 Conclusion

In this chapter, we have developed a piston-driven numerical wavetank model to simulate shallow water dynamics and developed a novel approach for implementing the variational principle in Firedrake. In the novel approach, we implement a time-discrete variational principle whereas, in the classical approach, we implement time-discretised weak formulations. The advantage of the novel approach is that it automates the derivation of time-discrete weak formulations and reduces human time and error in the code implementation stage. The comparison of the novel and classical approaches proves that both approaches are equivalent mathematically as well as numerically. Therefore, we can replace the classical approach and use the novel approach for developing more sophisticated numerical wavetank models.

Chapter 3

Mathematical and numerical modelling of piston-driven numerical wavetank based on nonlinear potential flow equations

3.1 Introduction

In this chapter, we explain the extension of the shallow-water equations based on numerical wavetank models, developed in chapter 2, to more sophisticated wavetank models based on potential-flow equations. The numerical models based on potential-flow equations are capable of simulating deep-water dynamics, as explained in §2.2.2. This potential-flow-based model of piston-driven numerical wavetank has been derived and implemented by employing the time-discrete variational principle which is a novel approach to implement the variational problems.

We commence by explaining the mathematical derivation of the variational problem in §3.2. Following that, the numerical discretisation and implementation of the derived mathematical model of the numerical wavetank are described in §3.3. The results obtained from the simulations are discussed in §3.4, and the possible extension of the numerical wavetank to simulate fluid-structure interactions problems is presented in §3.5. Ultimately, the chapter is concluded in section §3.6.

3.2 Variational modelling water dynamics based on potential-flow theory

The piston-driven wavetank model developed from the non-linear shallow water equations can be extended to a wavetank model based on the non-linear potential flow model. The non-linear shallow water equations model solves the equations only at the free surface while the model based on the non-linear potential flow also solves for the inner fluid domain. Therefore, in this model, the velocity potential $\phi(x, y, z, t)$ depends on the horizontal coordinates x and y, and the vertical coordinate z of the tank. The schematic of the wavetank is shown in Fig. 3.1.

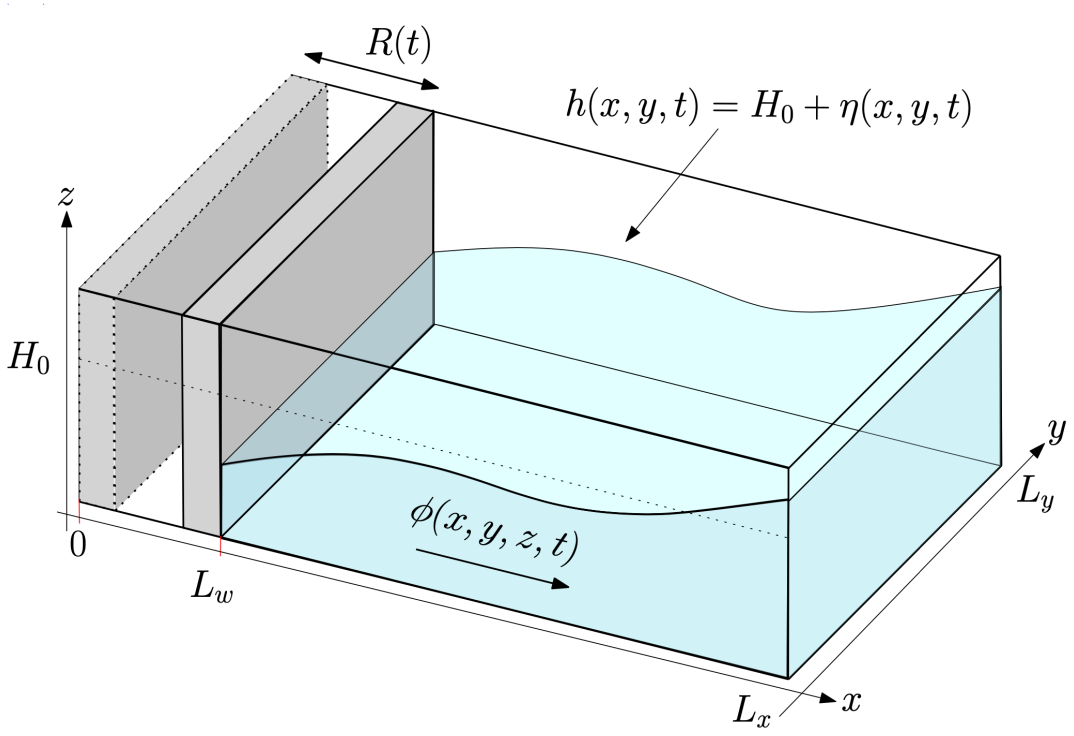


Figure 3.1: Schematic of a rectangular wavetank with piston wavemaker. The piston wavemaker oscillates horizontally in $0 \leq x \leq L_w$ to generate water waves. On the right side of the wave tank, there is a stationary solid wall.

Luke's variational principle for two-dimensional numerical wavetank based on potential-flow water waves with piston wavemaker $R(t)$ at $x = 0$ and solid wall at $x = L$ is given as:

$$\begin{aligned} 0 &= \delta \int_0^T \int_{R(t)}^L \int_0^{h(x,t)} -\rho \partial_t \phi + \mathcal{H} \, \mathrm{d}z \, \mathrm{d}x \, \mathrm{d}t \\ &= \delta \int_0^T \int_{R(t)}^L \int_0^{h(x,t)} -\rho \Big(\partial_t \phi + \frac{1}{2} |\boldsymbol{\nabla} \phi|^2 + g(z - H_0) \Big) \, \mathrm{d}z \, \mathrm{d}x \, \mathrm{d}t, \end{aligned} \tag{3.1}$$

where Hamilton $\mathcal{H}$ consists the kinetic and potential energies.

3.2.1 Linear potential flow equations

The corresponding variational principle (VP) for two-dimensional linear potential-flow equations without piston-wavemaker is given by:

$$0 = \delta \int_0^T \int_0^{L_x} \left(\phi \partial_t \eta - \frac{1}{2} g \eta^2 - \int_0^{H_0} \frac{1}{2} \left| \nabla \phi \right|^2 \mathrm{d}z \right) \mathrm{d}x \, \mathrm{d}t. \tag{3.2}$$

The equations of motions for the linear potential-flow model with piston wavemaker can be derived by taking the variations of (3.2) with respect to $\eta(x,t)$ and $\phi(x,t)$, and then integrating by parts to eliminate the boundary terms by applying the end-point conditions, i.e. $\delta\phi|_{t=0} = \delta\phi|_{t=T} = 0$.

$$\delta\phi : \nabla^2 \phi = 0 \quad \text{on} \quad \Omega, \tag{3.3a}$$

$$\delta\phi|_{z=H_0} : \partial_t \eta = \partial_z \phi \quad \text{at} \quad z = H_0, \tag{3.3b}$$

$$\delta\eta : \partial_t \phi = -g\eta \quad \text{at} \quad z = H_0. \tag{3.3c}$$

In the classical approach of implementing the variational principle we derive the weak formulations by multiplying the equations of motions with a test function and then we integrate by parts in space. In the final step, we discretize the weak formulations in time and implement them in Firedrake. The time-discretised weak formulations for the linear potential flow case are derived as follows:

$$\int_0^{L_x} \left(\frac{(\phi^{n+1} - \phi^n)}{\Delta t} + g\eta^n \right) \delta\eta^n \, \mathrm{d}x = 0, \tag{3.4}$$

$$\int_0^{L_x} \int_0^{H_0} \boldsymbol{\nabla} \delta\phi^{n+1} \cdot \boldsymbol{\nabla} \phi^{n+1} \, \mathrm{d}x \, \mathrm{d}z = 0, \tag{3.5}$$

$$\int_0^{L_x} \delta\phi^{n+1} \frac{(\eta^{n+1} - \eta^n)}{\Delta t} \, \mathrm{d}x - \int_0^{L_x} \int_0^{H_0} \boldsymbol{\nabla} \phi^{n+1} \cdot \boldsymbol{\nabla} \delta\phi^{n+1} \, \mathrm{d}x \, \mathrm{d}z = 0. \tag{3.6}$$

Notice that (3.5) may seem superfluous as it repeats in the (3.6), however in the numerical implementation process we need to explicitly update the velocity potential of the inner domain. In chapter 6 of [72], Salwa has explicitly implemented (3.5) in Firedrake to update the inner domain velocity potential. However, we propose a different approach in which we define the velocity potential ($\phi(x,z,t)$) of the complete domain as a sum of velocity potential for the free surface ($\psi(x,t)$) and velocity potential for the inner domain ($\varphi(x,z,t)$), such that when at free surface the velocity component for the inner domain is zero, i.e. $\phi(x,z,t) = \varphi(x,z,t) +$

$\psi(x,t)z/H_0$ (with $\varphi(x,H_0,t)=0$). Based on this split approach, our time-discrete variational principle based on the symplectic-Euler scheme is given as:

$$0 = \delta \int_0^{L_x} \psi^{n+1} \frac{(\eta^{n+1}-\eta^n)}{\Delta t} + \psi^n \frac{\eta^n}{\Delta t} - \frac{1}{2} g(\eta^n)^2 \,\mathrm{d}x \\ - \int_0^{L_x}\int_0^{H_0} \frac{1}{2}|\boldsymbol{\nabla}\varphi^{n+1}|^2 + \boldsymbol{\nabla}\varphi^{n+1}\cdot\boldsymbol{\nabla}\psi^{n+1} + \frac{1}{2}|\boldsymbol{\nabla}\psi^{n+1}|^2 \,\mathrm{d}x\,\mathrm{d}z. \tag{3.7}$$

The corresponding time-discrete weak formulations that are automatically generated and implemented in Firedrake by using command derivative are as follows:

$$\int_0^{L_x} \Big(\frac{(\psi^{n+1}-\psi^n)}{\Delta t} + g\eta^n\Big)\,\delta\eta^n \,\mathrm{d}x + \int_0^{L_x}\int_0^{H_0} \boldsymbol{\nabla}(\psi^{n+1}+\varphi^{n+1})\cdot\boldsymbol{\nabla}\delta\varphi^{n+1}\,\mathrm{d}x\,\mathrm{d}z = 0, \tag{3.8}$$

$$\int_0^{L_x} \frac{(\eta^{n+1}-\eta^n)}{\Delta t}\delta\psi^{n+1}\,\mathrm{d}x = 0. \tag{3.9}$$

In (3.8), we use the known value of free surface elevation η^n and ψ^n to find the updated value of free surface velocity potential ψ^{n+1} and update the inner domain velocity potential φ^{n+1} by using the updated ψ^{n+1} as Dirichlet boundary condition in unison. Hence, the test function $\delta\phi^{n+1}$ is defined on mixed function space [30]. Following that we use the updated value of velocity potential at free surface ψ^{n+1} in (3.9) to compute the updated value of free surface elevation η^{n+1}. After developing the implementation strategy for the time-discrete variational principle of linear potential flow dynamics we can proceed with implementing the same strategy for nonlinear potential flow equations.

3.2.2 Nonlinear potential flow equations

The equations of motions for the non-linear potential-flow model with piston wavemaker can be derived by taking the variations of (3.1) with respect to $h(x,t)$ and $\phi(x,t)$, and then eliminating the boundary terms by applying the end-point conditions, i.e. $\delta\phi|_{t=0} = \delta\phi|_{t=T} = 0$. Finally, the arbitrariness of variations $\delta\phi$, δh, and $(\delta\phi)|_{z=h}$ yields the following equations of motion

$$\delta\phi: \quad \nabla^2\phi = 0 \quad \text{on } \Omega, \tag{3.10a}$$

$$\delta\phi|_{z=H_0}: \quad \partial_t h + \nabla\phi\cdot\nabla h = \partial_z\phi \quad \text{at } z = h(x,y,t), \tag{3.10b}$$

$$\delta h: \quad \partial_t\phi + \frac{1}{2}|\nabla\phi|^2 + g\big(h-H_o\big) = 0 \quad \text{at } z = h(x,y,t), \tag{3.10c}$$

$$\delta\phi|_{x=R(t)}: \quad \partial_x\phi = \partial_t R. \tag{3.10d}$$

To obtain the VP in a fixed domain, the VP stated in (3.1) is transformed to one in new, fixed coordinate $\eta \in [0, H_0]$ and $\xi \in [0, L]$ with domain movement in $x \in [R(t), L_w]$ and no domain movement in $x \in [L_w, L]$ with $L_w \ll L$. The domain after transformation is shown in Fig. 3.2. The forward and backward coordinate transformations read

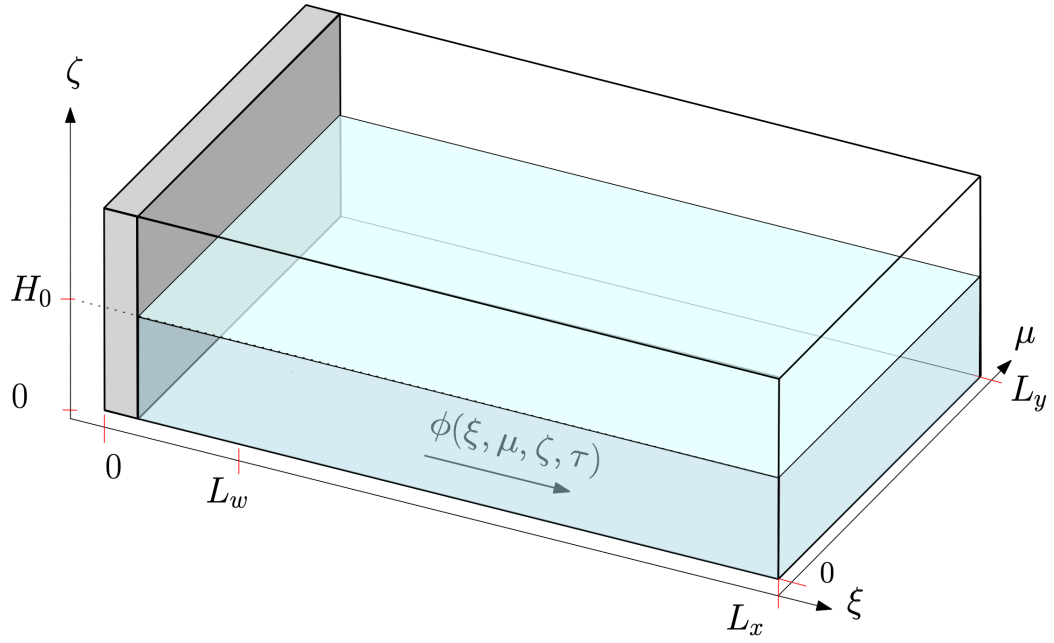


Figure 3.2: Schematic of the static computational domain corresponding to a rectangular wavetank with piston wavemaker. Transformed spatial coordinates are ξ, μ, ζ.

$$
\begin{aligned}
x(\xi,\eta,\tau) =& \begin{cases} R(\tau) + \dfrac{\xi(L_w - R(\tau))}{L_w} & \xi \in [0, L_w] \\ \xi & \xi \in [L_w, L] \end{cases} \\
=& \frac{\xi L_w + (L_w - \xi) R(\tau) \Theta(L_w - \xi)}{L_w}, \quad \tau = t; \qquad (3.11) \\
\xi(x,z,t) =& L_w \frac{x - R(t)}{L_w - R(t)} \\
& - \frac{R(t)(x - L_w)\Theta(L_w - x)}{L_w - R(t)}, \quad t = \tau; \qquad (3.12) \\
\zeta(x,z,t) =& z \frac{H_0}{h(x,t)}. \qquad (3.13)
\end{aligned}
$$

Here $\Theta(L_w - x)$ is the Heaviside function, defined as unity for $x < L_W$ and zero elsewhere. Thus making the coordinate transformation effective in $x \in [R(t), L_w]$ and eliminating the need

to transform the structure away from the wavemaker. Using the short-hand notations:

$$X = \xi - L_w; \tag{3.14a}$$

$$\tilde{R}_\tau = R_\tau \Theta(L_w - \xi); \tag{3.14b}$$

$$\tilde{R} = R(\tau)\Theta(L_w - \xi); \tag{3.14c}$$

$$V = L_w - \tilde{R}; \tag{3.14d}$$

The Jacobian of transforms (3.13) mapping (x, z, t) to (ξ, ζ, τ) where $\zeta \in [0, H_0]$ and $\xi \in [0, L_w]$ is,

$$J \equiv \frac{\partial(x,z,t)}{\partial(\xi,\zeta,\tau)} = \begin{bmatrix} \dfrac{V}{L_w} & 0 & -\tilde{R}_\tau \dfrac{X}{L_w} \\ h_\xi \dfrac{\zeta}{H_0}\dfrac{V}{L_w} & \dfrac{h}{H_0} & \dfrac{\zeta}{H_0}\Big(h_\tau - \tilde{R}_\tau \dfrac{X}{L_w} h_\xi\Big) \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.15}$$

where $\Theta(L_w - \xi)$ is the Heaviside function and $L_w = O(\lambda)$, with λ being the wavelength of the water waves generated by the wavemaker. The inverse of J^{-1} is calculated as

$$J^{-1} \equiv \frac{\partial(\xi,\zeta,\tau)}{\partial(x,z,t)} = \begin{bmatrix} \dfrac{L_w}{V} & 0 & \tilde{R}_\tau \dfrac{X}{V} \\ -h_\xi \dfrac{\zeta}{h}\dfrac{L_w}{V} & \dfrac{H_0}{h} & -\dfrac{\zeta}{h}\Big(h_\tau + \tilde{R}_\tau \dfrac{X}{V} h_\xi\Big) \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.16}$$

Substituting the values of partial derivatives in (3.16) into the chain rule, the partial derivatives in transformed coordinates are given as

$$\partial_t = \tilde{R}_\tau \frac{X}{V}\partial_\xi - \frac{\zeta}{h}\Big(\tilde{R}_\tau \frac{X}{V} h_\xi + h_\tau\Big)\partial_\zeta + \partial_\tau, \tag{3.17a}$$

$$\partial_x = \frac{L_w}{V}\partial_\xi - \zeta \frac{h_\xi}{h}\frac{L_w}{V}\partial_\zeta, \tag{3.17b}$$

$$\partial_z = \frac{H_0}{h}\partial_\zeta, \tag{3.17c}$$

$$\mathrm{d}x\,\mathrm{d}t = |J|\,\mathrm{d}\xi\,\mathrm{d}\tau = \frac{h}{H_0}\frac{V}{L_w}\,\mathrm{d}\xi\,\mathrm{d}\tau. \tag{3.17d}$$

Subsequently, after substituting the transformations (3.17) into (3.1), the VP in the transformed coordinates is given as

$$\begin{aligned}0 =&\delta \int_0^T \int_0^{L_x} \int_0^{H_0} \Big[V\Big(h\phi_\tau - \zeta h_\tau \phi_\zeta\Big) + X\tilde{R}_\tau\Big(h\phi_\xi - \zeta h_\xi \phi_\zeta\Big) \\ &+ \frac{1}{2}\frac{L_w}{V} h(\phi_\xi - \frac{\zeta}{h} h_\xi \phi_\zeta)^2 + \frac{1}{2} V \frac{H_0^2}{h} (\phi_\zeta)^2 \\ &+ gVh\Big(\zeta \frac{h}{H_0} - H_0\Big)\Big] \frac{1}{H_0 L_w}\, \mathrm{d}\zeta\, \mathrm{d}\xi \mathrm{d}\tau. \end{aligned} \tag{3.18}$$

After multiplying by $H_0 L_w$ and integrating by parts in x the third term , and in z the second and fourth terms of VP(3.18), the resulted VP can be written as:

$$\begin{aligned}0 =&\delta \int_0^T \Big[\int_0^{L_x} \Big(\int_0^{H_0} \Big[\frac{1}{2}\frac{L_w^2}{V} h(\phi_\xi - \frac{\zeta}{h} h_\xi \phi_\zeta)^2 + \frac{1}{2} V \frac{H_0^2}{h} (\phi_\zeta)^2\Big]\, \mathrm{d}\zeta \\ &+ H_0\Big[-\tilde{\phi}\Big(X\tilde{R}_\tau h_\xi + V h_\tau\Big) + ghV\Big(\frac{1}{2}h - H_0\Big)\Big]|_{\zeta=H_0}\Big)\, \mathrm{d}\xi \\ &+ \int_0^{H_0} \Big(L_w h \tilde{R}_\tau \phi\Big)|_{\xi=0}\, \mathrm{d}\zeta\Big] \mathrm{d}\tau. \end{aligned} \tag{3.19}$$

Now we will split the velocity potential for the complete domain (ϕ) into a free surface (ψ) and inner domain (φ) components, such that when at free surface the velocity component for the inner domain is zero, i.e. $\phi(x,z,t) = \varphi(x,z,t) + \psi(x,t)z/H_0$ (with $\varphi(x,H_0,t) = 0$). (3.19) with the split is given as follows:

$$\begin{aligned}0 =&\delta \int_0^T \Big[\int_0^{L_x} \Big(\int_0^{H_0} \Big[\frac{1}{2}\frac{L_w^2}{V} h\Big((\psi+\varphi)_\xi - \frac{\zeta}{h} h_\xi \varphi_\zeta\Big)^2 + \frac{1}{2} V \frac{H_0^2}{h} (\varphi_\zeta)^2\Big]\, \mathrm{d}\zeta \\ &+ H_0\Big[-\psi\Big(X\tilde{R}_\tau h_\xi + V h_\tau\Big) + ghV\Big(\frac{1}{2}h - H_0\Big)\Big]|_{\zeta=H_0}\Big)\, \mathrm{d}\xi \\ &+ \int_0^{H_0} \Big(L_w h \tilde{R}_\tau(\psi+\varphi)\Big)|_{\xi=0}\, \mathrm{d}\zeta\Big] \mathrm{d}\tau. \end{aligned} \tag{3.20}$$

After taking the variations of (3.19) with respect to h, $\tilde{\phi}$ and ϕ and eliminating the boundary terms obtained after integrating by parts. Eventually, due to the arbitrariness of variations $\delta\phi$, $\delta\psi$, and δh we obtain the following equations of motions and boundary conditions in the

transformed coordinates system, as follows:

$$\delta\phi: \quad \Big[\frac{L_w^2}{V}\Big(h\phi_{\xi\xi}+\frac{\zeta}{h}(h_\xi)^2(2\phi_\zeta+\zeta\phi_{\zeta\zeta}-\zeta(\phi_\zeta h_{\xi\xi})+2h_\xi\phi_{\xi\zeta})\Big) + V\frac{H_0^2}{h}\phi_{\zeta\zeta}\Big]=0; \tag{3.21}$$

$$\delta\psi|_{\zeta=H_0}: \Big[\frac{L_w^2}{V}\Big(\frac{H_0}{h}h_\xi^2\phi_\zeta-h_\xi\phi_\xi\Big)-X\tilde{R}_\tau h_\xi+V\Big(\frac{H_0}{h}\phi_\zeta-h_\tau\Big)\Big]=0; \tag{3.22}$$

$$\delta h\Big|_{\zeta=H_0}: \Big[\phi_\tau+g(h-H_0)-\frac{1}{2}\frac{H_0^2}{h^2}(\phi_\zeta^2)+\frac{1}{2}\Big(\frac{L_w}{V}\Big)^2\Big(\phi_\xi^2-\frac{H_0^2}{h^2}(h_\xi^2)(\phi_\zeta^2)\Big) + \frac{X}{V}\tilde{R}_\tau\phi_\xi\Big]=0 \tag{3.23}$$

$$\delta\phi|_{\xi=0}: \quad \Big[\frac{L_w}{V}\Big[\phi_\xi-\frac{\zeta}{h}h_\xi\phi_\zeta\Big]-\tilde{R}_\tau\Big]=0. \tag{3.24}$$

After splitting the velocity potential for the complete domain and evaluating the partial derivative in (3.24) yields the following equations of motion:

$$\delta\phi: \quad \Big[\frac{L_w^2}{V}\Big(h(\psi+\varphi)_{\xi\xi}+\frac{\zeta}{h}(h_\xi)^2(2\varphi_\zeta+\zeta\varphi_{\zeta\zeta}-\zeta(\varphi_\zeta h_{\xi\xi})+2h_\xi(\psi+\varphi)_{\xi\zeta})\Big) + V\frac{H_0^2}{h}\varphi_{\zeta\zeta}\Big]=0; \tag{3.25}$$

$$\delta\psi|_{\zeta=H_0}: \Big[\frac{L_w^2}{V}\Big(-h_\xi\phi_\xi\Big)-X\tilde{R}_\tau h_\xi+V\Big(-h_\tau\Big)\Big]=0; \tag{3.26}$$

$$\delta h\Big|_{\zeta=H_0}: \Big[\psi_\tau+g(h-H_0)+\frac{1}{2}\Big(\frac{L_w}{V}\Big)^2\Big(\phi_\xi^2\Big)+\frac{X}{V}\tilde{R}_\tau\psi_\xi\Big]=0 \tag{3.27}$$

$$\delta\phi|_{\xi=0}: \quad \Big[\frac{L_w}{V}\Big[(\psi+\varphi)_\xi-\frac{\zeta}{h}h_\xi\varphi_\zeta\Big]-\tilde{R}_\tau\Big]=0. \tag{3.28}$$

The process of deriving a time-discrete variational principle could be iterative because we need to formulate a variational principle that yields time-discretised weak formulations when variations are taken.

3.2.3 Time discrete VP for potential-flow equations with piston wavemaker

Presently, elliptic equations in *Firedrake* must be solved with a Dirichlet boundary condition that does not include an unknown. For consistent geometric time integrators such as SE or SV schemes, potential-flow equations require the solution of Laplace's equation with the unknown $\phi(\xi,\zeta=H_0,t)$ specified at the transformed free surface. Hence, we introduce a splitting of the velocity potential into a sum of its surface-bound and interior parts, i.e. $\phi(\xi,\zeta,\tau)=\psi(\xi,\tau)\hat{\phi}(\zeta)+\varphi(\xi,\zeta,\tau)$ for $\hat{\phi}(H_0)=1$, with the former $\psi(\xi,\tau)=\phi(\xi,H_0,\tau)$ and the latter zero at the free surface (in transformed coordinates), $\varphi(\xi,H_0,\tau)=0$. To date, we have

mainly considered a vertical structure function $\hat{\phi}(\zeta) = 1$ but other choices may be more optimal. Also note that $\partial_\zeta \psi = 0$ and that the combined Neumann condition $\hat{\mathbf{n}} \cdot (\partial_\xi(\psi + \varphi), \partial_\zeta \varphi) = 0$ holds at solid-wall boundaries with outward normal $\hat{\mathbf{n}}$. Consequently, this condition remains imposed naturally in the weak formulations ensuing from the time-discrete VP. The domain considered here has a flat bottom and a vertical sidewall on the right.

A first-order time-discrete analogue of the VP (3.19) for potential-flow dynamics with a piston wavemaker yielding an SE time discretisation for the two conjugate kinematic and Bernoulli equations at the free surface reads

$$\begin{aligned} 0 =& \delta \int_0^{L_x} \int_0^{H_0} -\Big[\frac{1}{2}\frac{L_w^2}{V^{n+1}} h^n (\psi_\xi^{n+1} + \varphi_\xi^{n+1} + (\zeta/h^n) h_\xi^n \varphi_\zeta^{n+1})^2 + \frac{1}{2} V^{n+1} \frac{H_0^2}{h^n} (\varphi_\zeta^{n+1})^2\Big] \,\mathrm{d}\zeta \,\mathrm{d}\xi \\ &+ \int_0^{L_x} -g H_0 V^{n+1} h^n (\frac{1}{2} h^n - H_0) + H_0 V^{n+1} \psi^{n+1} \frac{(h^{n+1} - h^n)}{\Delta t} + H_0 V^n \psi^n \frac{h^n}{\Delta t} \\ &\qquad - H_0 \psi^{n+1} (\xi - L_w) R_\tau^{n+1} h_\xi^n \,\mathrm{d}\xi \\ &- \int_0^{H_0} L_w R_\tau^{n+1} (\psi^{n+1} + \varphi^{n+1}) h^n|_{\xi=0} \,\mathrm{d}\zeta \,\mathrm{d}\tau. \end{aligned} \tag{3.29}$$

Note that $V\psi$ is considered as conjugate variable to the "position" variable h. Hence, we consistently evaluate terms with explicit time dependence at discrete time level $n+1$, except for one term that links to the past. Two intrinsically coupled weak formulations follow from variations of the VP (3.29) with respect to $\{\delta h^n, \delta\varphi^{n+1}\}$ yielding the combined solution updates $\{\psi^{n+1}, \varphi^{n+1}\}$. The algebraic equation for ψ^{n+1} is nonlinear and coupled to a linear equation for φ^{n+1}. Herein, the Dirichlet boundary condition $\varphi^{n+1}(\xi, H_0, \tau) = 0$ is explicitly imposed at the (transformed) free surface at $z = H_0$. The final weak formulation follows from the variation of VP (3.29) with respect to $\delta\psi^{n+1}$, yielding an explicit update h^{n+1}. We have similarly derived a second-order SV time-discrete VP for potential-flow equations with a piston wavemaker that

reads

$$
\begin{aligned}
0 =& \delta \int_0^{L_x} \int_0^{H_0} -\Big[\frac{1}{4}\frac{L_w^2}{V^{n+1/2}} h^n \zeta/h^n) h_\xi^n) \varphi_\zeta^{n+1/2}\Big)^2 \\
&+ \Big(\frac{1}{4}\frac{L_w^2}{V^{n+1/2}} h^{n+1}(\psi_\xi^{n+1/2} + \varphi_\xi^{n+1/2} + (\zeta/h^{n+1}) h_\xi^{n+1}) \varphi_\zeta^{n+1/2}\Big)^2 \\
&+ \frac{1}{4} V^{n+1/2} \frac{H_0^2}{h^n} (\varphi_\zeta^{n+1/2})^2 + \frac{1}{4} V^{n+1/2} \frac{H_0^2}{h^{n+1}} (\varphi_\zeta^{n+1/2})^2 \Big] \, \mathrm{d}\zeta \, \mathrm{d}\xi \\
&+ \int_0^{L_x} -\frac{1}{2} g H_0 V^{n+1/2} h^{n+1} (\frac{1}{2} h^{n+1} - H_0) - \frac{1}{2} g H_0 V^{n+1/2} h^n (\frac{1}{2} h^n - H_0) \\
&+ H_0 (V^{n+1/2} \psi^{n+1/2} \frac{(h^{n+1} - h^n)}{\Delta t} - V^{n+1} \psi^{n+1} \frac{h^{n+1}}{\Delta t} + V^n \psi^n \frac{h^n}{\Delta t}) \\
&- \frac{1}{2} H_0 \psi^{n+1/2} (\xi - L_w) R_\tau^{n+1/2} (h_\xi^n + h_\xi^{n+1}) \, \mathrm{d}\xi \\
&- \frac{1}{2L_w} \int_0^{H_0} R_\tau^{n+1/2} (\psi^{n+1/2} + \varphi^{n+1/2}) (h^n + h^{n+1})|_{\xi=0} \, \mathrm{d}\zeta \, \mathrm{d}\tau. \qquad (3.30)
\end{aligned}
$$

The combined variations of VP (3.30) with respect to $\{h^n, \varphi^{n+1/2}\}$ yield equations for the pairing $\{\psi^{n+1/2}, \varphi^{n+1/2}\}$, the latter which are solved in unison. These coupled equations are nonlinear in $\psi^{n+1/2}$ and linear in $\varphi^{n+1/2}$. Finally, the variation of VP (3.30) with respect to h^{n+1} yields an equation for the update ψ^{n+1}. The conjugate variable $V\psi$ is again suitably discretised in time. Additionally, a VP matching the modified midpoint time discretisation was derived in [69].

3.3 Numerical implementation

3.3.1 Spatial discretisation

The VPs for the numerical wavetanks are solved after spatial discretisation of the computational domain and weak formulations by using Firedrake. The two-dimensional computational domain is created by extruding the horizontal line mesh (x direction) along the fluid depth, i.e. z direction. The function space to numerically solve the equations is the first-order piece-wise-linear continuous Galerkin (CG); and it is created in a way that the function space of the free-surface velocity potential ($\psi(x, H_0, t)$) and the velocity potential of the inner domain ($\varphi(x, z, t)$) can be differentiated. For this purpose, Firedrake allows the creation of a mixed-function space that can extract the $\psi(x, H_0, t)$ and $\varphi(x, z, t)$ from the total velocity potential $\phi(x, z, t)$ by using a command `split` on the mixed function space. See Appendix B for detailed derivation of the spatial discretisation of linear potential flow equations by using finite element expansions.

A two-dimensional rectangular computational domain of 140m x 1m in x and z directions, respectively, is spatially discretised by using 600 quadrilateral finite elements. There are 200 elements in the x direction and 4 elements in the z direction.

3.3.2 Timestep criterion

Gagarina et al. [31] have developed robust first- and second-order symplectic time integrator schemes that are applicable to Hamiltonian forms. These schemes, i.e. the first-order symplectic Euler (SE), and the second-order Störmer-Verlet (SV), are conditionally stable. A linear stability criterion can be based on analysis of the linear harmonic oscillator, with frequency ω_{max} [37, 34], given as

$$\Delta t = CFL(2/\omega_{max}), \tag{3.31}$$

where for the Courant-Friedrichs-Lewy (CFL)-number we take $CFL \leq 1$. For linear wave dynamics, ω_{max} is the maximum wave frequency, which can be estimated using the linear potential-flow dispersion relation,

$$\omega_{max} \approx \sqrt{gk \tanh(kH_0)} \leq \sqrt{gH_0}k \tag{3.32}$$

the upper bound holding in the shallow-water limit. The maximum wavenumber $k = 2\pi/\Delta x$ depends on an estimate Δx of the minimum mesh size. For nonlinear wave dynamics, the CFL (again satisfying $CFL \leq 1$) needs to be estimated and tested. The current mathematical and numerical model cannot model wave breaking phenomenon, therefore, codes are anticipated to become unreliable when waves become too steep; this can be circumvented by implementing parametrised wave-breaking schemes such as the one in [81]. Wave-breaking parametrisation schemes have not been considered here.

3.4 Results and discussion

The described spatio-temporal discretisation of nonlinear wave dynamics results in algebraic systems of equations having both nonlinear and linear parts. These have been solved using the nonlinear solvers built into Firedrake, including Newton iteration and *PetSc* [5], which solvers can be made problem-specific via suitable preconditioners. Optimisation of the numerical solvers employed is in progress.

3.4.1 Comparison of driven long waves using shallow-water and potential-flow dynamics

Simulation results obtained from the presently developed numerical wavetanks based on the VPs for linear/non-linear shallow-water equations are compared with the potential-flow solver, from ([36]), in the small-amplitude limit for the case of waves generated from rest by a piston wavemaker. In the latter potential-flow solver, weak forms are implemented explicitly, after the vertical z-dependence in the VP is integrated out using one element in the vertical with a higher-order standard Lagrange polynomial expansion of the variables; the resulting VP then depends on only the horizontal coordinates and time. Corresponding variations yield algebraically-complex weak formulations that are subsequently implemented in (Firedrake), cf. ([34, 36]).

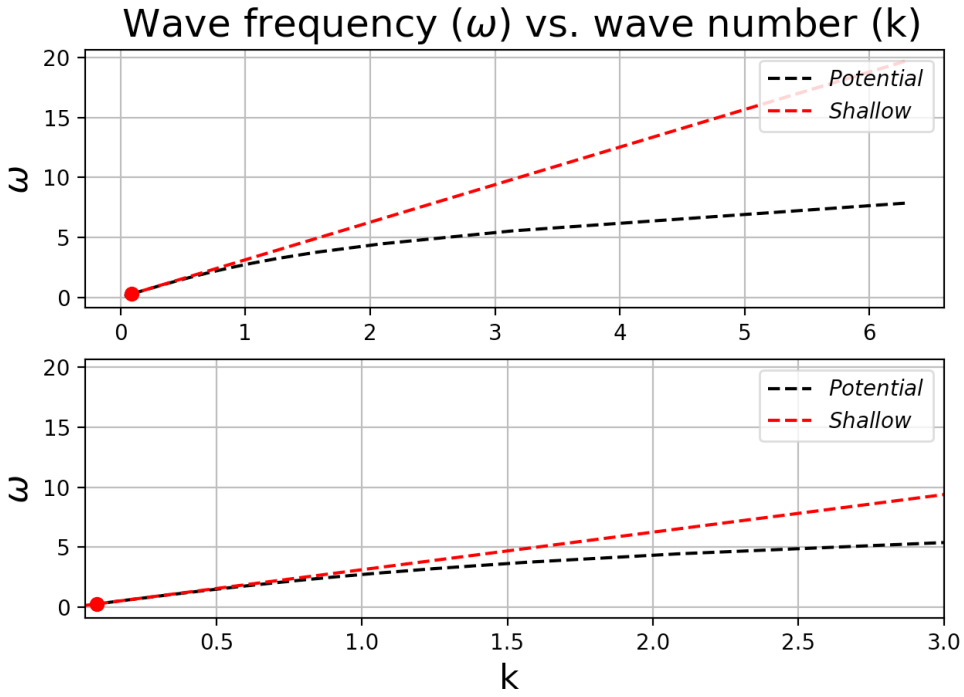


Figure 3.3: Wave frequencies for potential-flow (black) and shallow-water (red) cases. The red disc shows the chosen frequency of the wavemaker.

The angular frequency of the wavemaker is chosen in such a way as to generate long waves, thereby admitting a comparison of the results obtained from shallow-water equations with those obtained by solving the potential-flow equations. The dispersion relation for wave frequency ω, as a function of wave number k, for linear potential-flow and shallow-water motion is shown, for free waves, in Fig. 3.3, in which the chosen wavemaker frequency is indicated by a red disc.

The (static computational) domain has dimensions $L_x = 140\text{m}$ and $H_0 = 1\text{m}$; first-order

piecewise-linear continuous Galerkin (CG1) polynomials are used, with 200 elements and timestep $\Delta t = 0.02$s; we furthermore employ $n_z = 4^{\text{th}}$-order Lagrange polynomials across the transformed depth. The wavemaker motion chosen and controllable parameters used are:

$$R(t) = \begin{cases} \gamma \cos(\sigma t) & 0 \leq t \leq T_p \\ 0 & t > T_p \end{cases}, \tag{3.33}$$

$$\sigma = \sqrt{gH_0}k = \sqrt{gH_0}2\pi/\lambda, \lambda = 70\text{m}, T_p = 2\pi/\sigma, \gamma = 0.002\text{m}.$$

Simulations are undertaken over two time periods $0 \leq t \leq 2T_p$. Initially, the wavemaker and free surface of the water are at rest. The wavemaker gradually starts from rest and stops after completing one wave period.

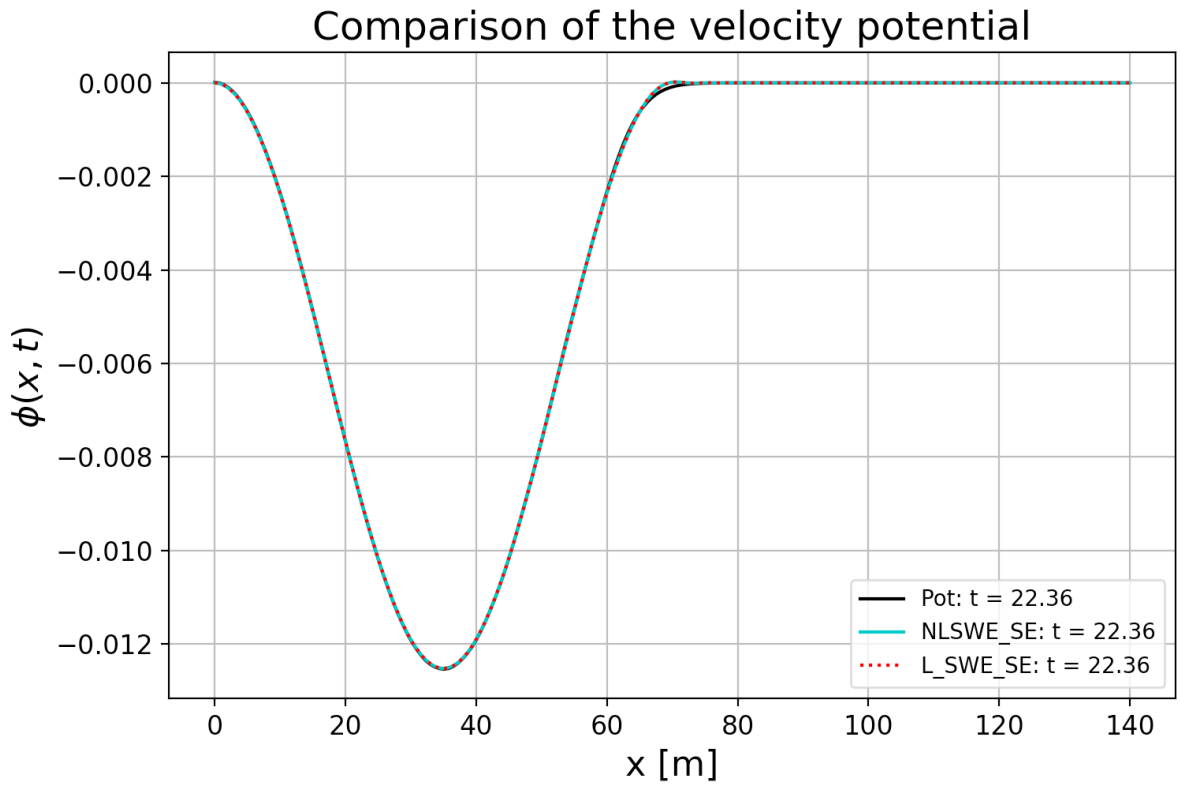


Figure 3.4: Free-surface velocity potential after one time period $t = T_p$. Black, cyan and red lines respectively correspond to potential, nonlinear and linear shallow-water solvers.

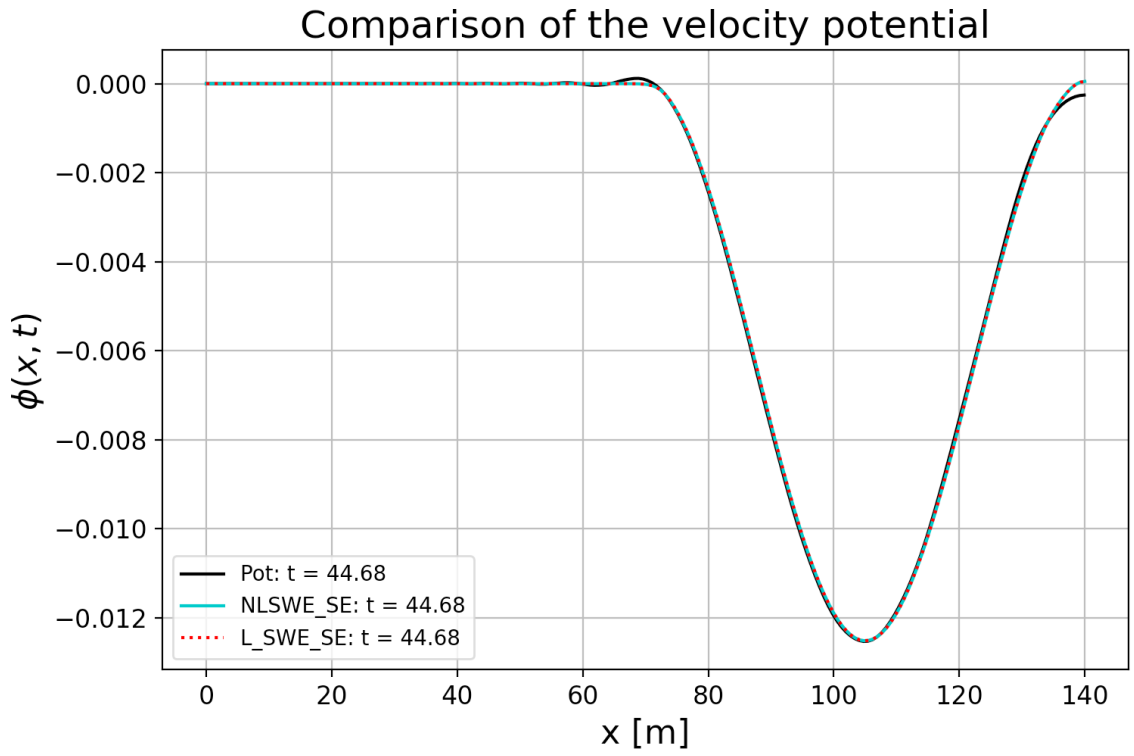


Figure 3.5: Free-surface velocity potential at final time $t = 2T_p$. Black, cyan and red lines respectively correspond to potential, nonlinear and linear solvers.

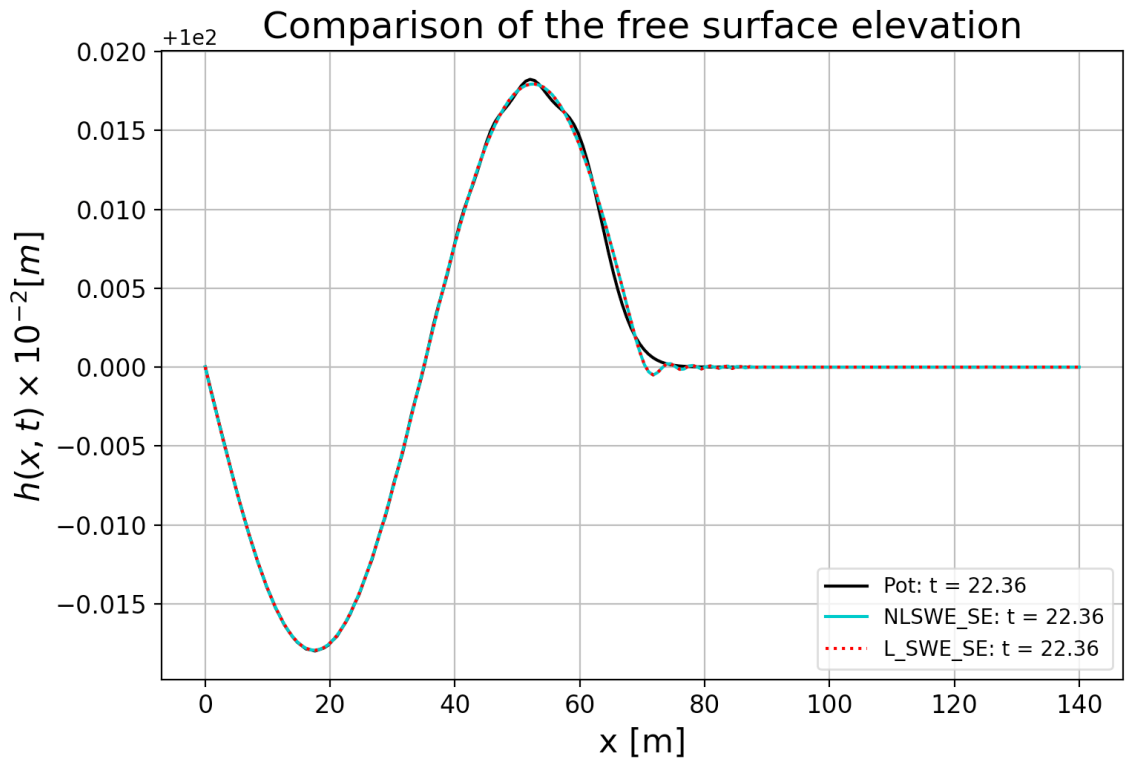


Figure 3.6: Free-surface elevation after one time period $t = T_p$. Black, cyan and red lines respectively correspond to potential, nonlinear and linear solvers.

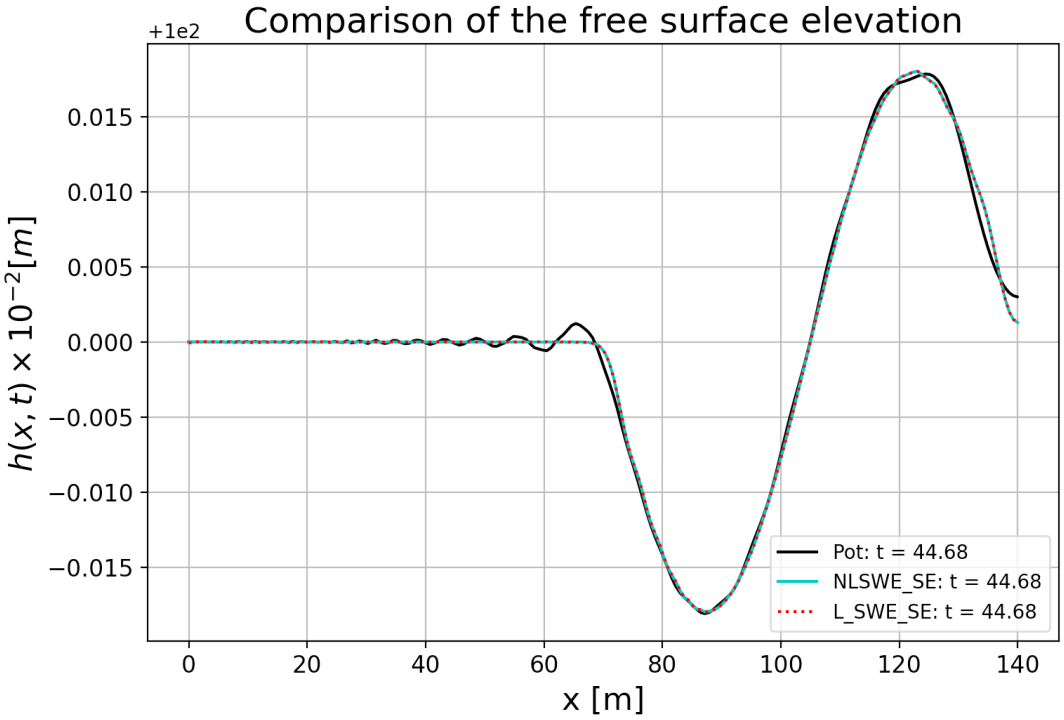


Figure 3.7: Free-surface elevation at final $t = 2T_p$. Black, cyan and red lines respectively correspond to potential-flow, nonlinear and linear solvers.

These simulations are seen to compare well, as evidenced by the near-coincident profiles of the free surfaces and surface-velocity potentials plotted at $t = T_p, 2T_p$ in Figs. 3.4 to 3.7. Finally, not shown here, we report that the potential-flow simulation directly implementing the VP (3.29) into Firedrake yields the same results as those obtained from this potential-flow simulation.

3.4.2 Test case: high-amplitude waves

In this test case, we generate waves of amplitude 0.3 m in the wavetank which is 25 m deep. The two-dimensional computational domain is 140 m long in x-direction and 25 m deep in z-direction. The domain is discretised into 1400 element in x-direction and 6 elements in z-direction, while the time step selected based on the Courant-Friedrichs-Lewy (CFL) condition, mentioned in (3.31), is 0.0159. Note that the comparison study explained in the previous study has waves of 1.5 mm amplitude. Therefore, we test the wavetank by increasing the wave amplitude and allowing them to reflect from the fixed wall. The wave period T_p is 6.7673 seconds. The total simulation time is $8T_p = 54.1445$ seconds; the wavemaker is turned off after $4T_p = 27.056$ seconds and the waves generated by the wavemaker are allowed to reflect. Fig. 3.8 and 3.9 show the generated and reflected wave heights and velocity potentials, respectively.

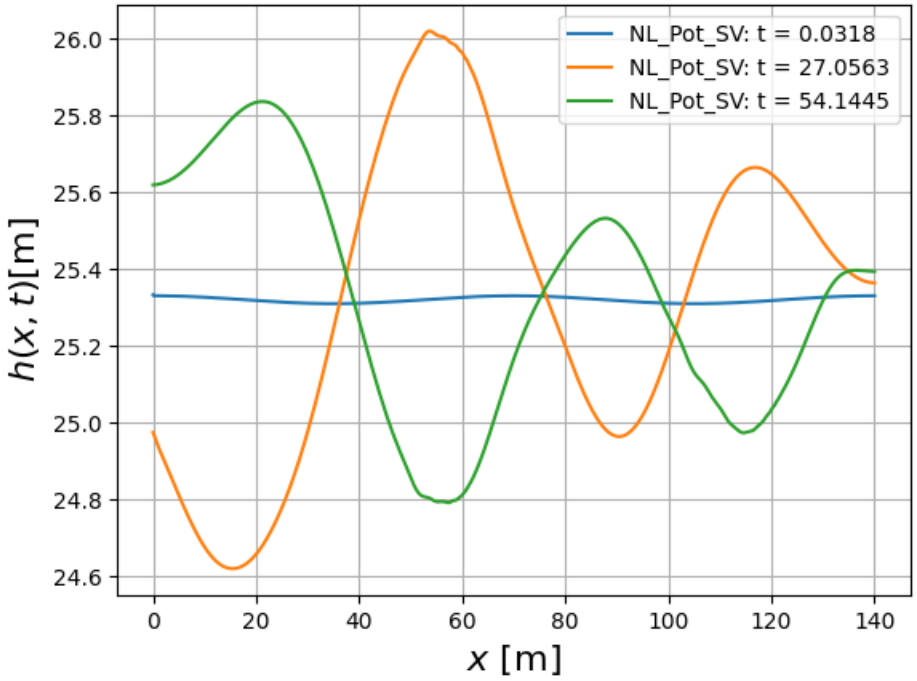


Figure 3.8: Evolution of free-surface elevation ($h = H_0 + \eta$) at different time steps. The wave generated by the wavemaker is shown in orange line while the reflected wave is shown in green line.

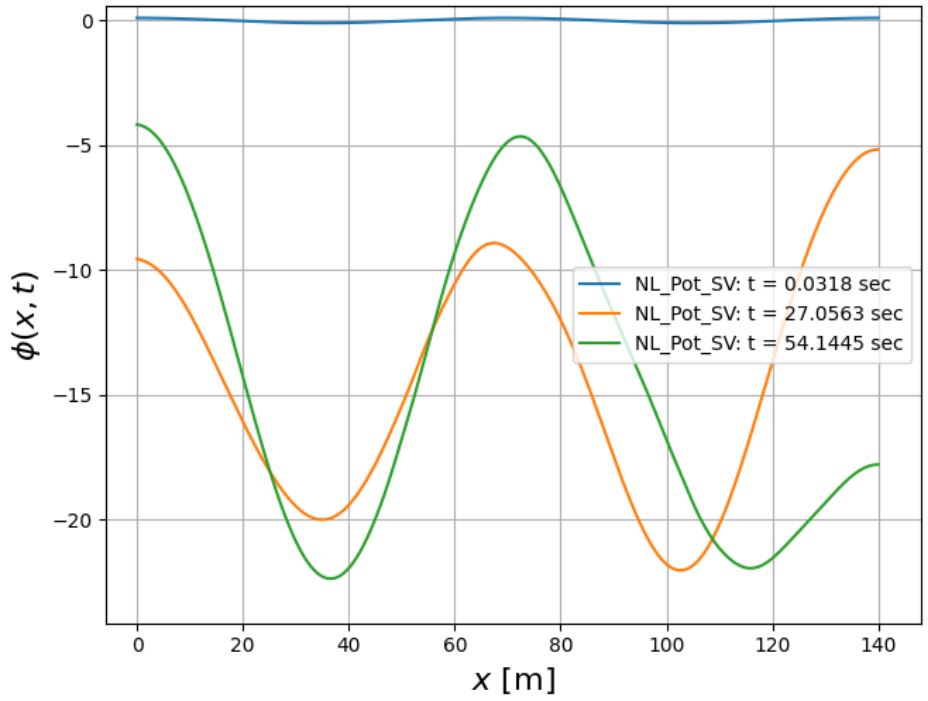


Figure 3.9: Evolution of velocity potential at different time steps.

Fig. 3.10 shows the evolution of the system's kinetic energy. It can been that the system's energy increases when the wavemaker is turned on, i.e. $0 < t < 27.056$ seconds, and oscillates at the same level when the wavemaker is turned off.

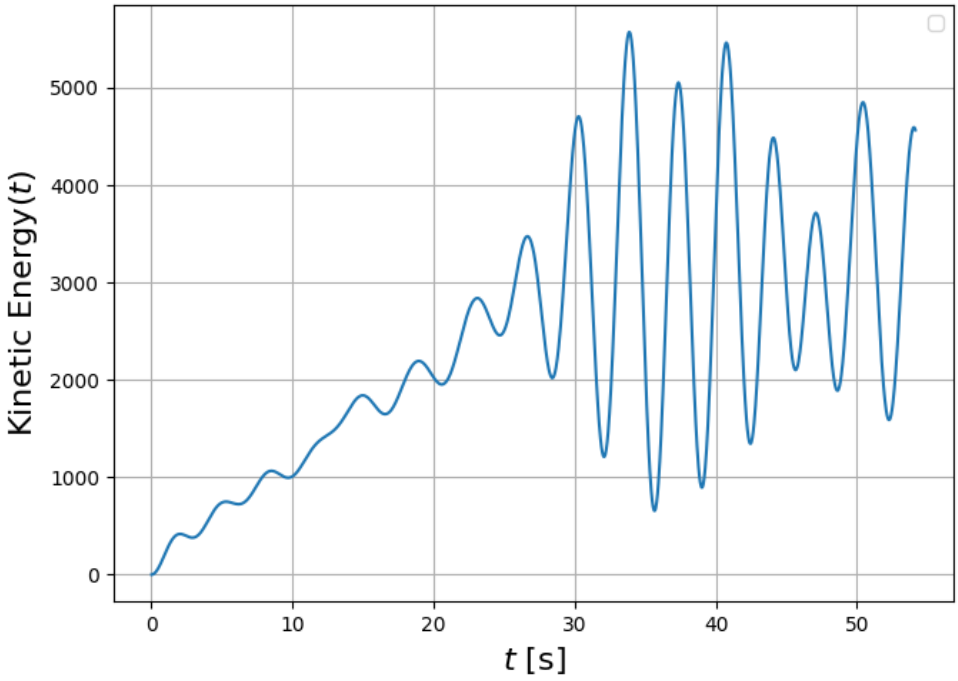
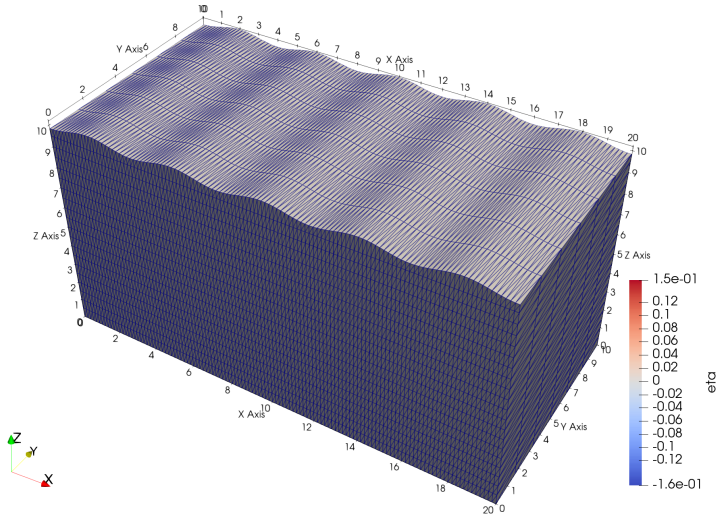


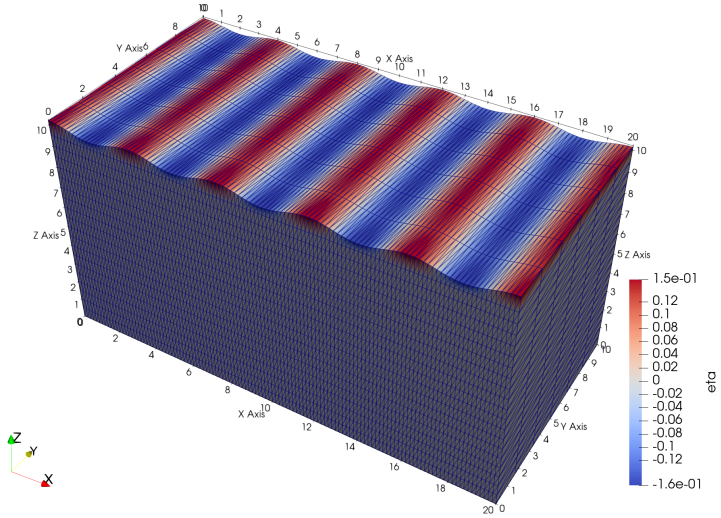
Figure 3.10: Evolution of the kinetic energy of the system.

3.4.3 Three-dimensional extension of two-dimensional wavetank

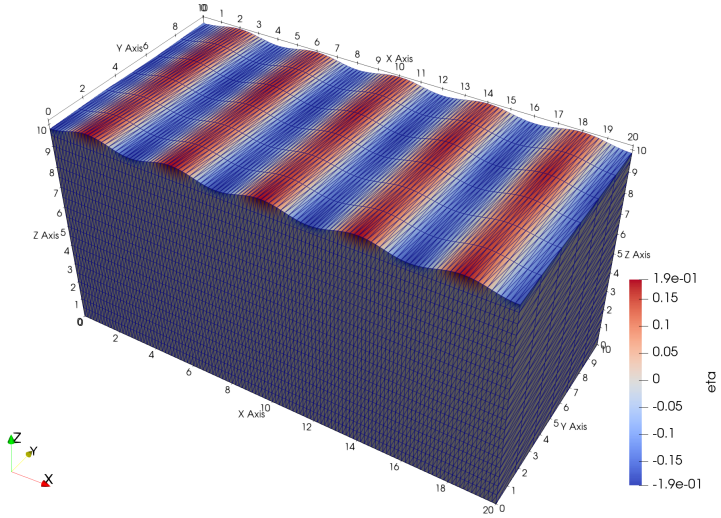
In Firedrake, a two-dimensional variational problem can easily be implemented in a three-dimensional computational domain which is generated by extruding the horizontal two-dimensional mesh in z-direction. The assignment of the boundary conditions to the three-dimensional domain slightly differs from the two-dimensional domain. For demonstration, we present a three-dimensional model of a numerical wavetank based on linear potential flow equations in Fig. 3.11. A similar framework can then be used to develop a three-dimensional wavetank based on nonlinear potential flow equations.



(a) The initial position (initial condition) of the free surface elevation at $t = 0$.



(b) Free surface elevation at $t = 25$s.



(c) Free surface elevation at $t = 50$s.

Figure 3.11: Demonstration of evolution of free surface elevation in a three-dimensional wavetank.

3.5 Extension of numerical wavetank to solve fluid-structure-interaction (FSI) problems

The variational principle for the numerical wavetank can be coupled with the variational principle of the hyperelastic structure equations to solve the problems of fluid-structure interactions. Salwa [72] has already developed a variational principle that couples water dynamics with hyperelastic beam equations, however, he has used the classical approach for implementing the variational principle and the water and beam dynamics are linearized. Therefore, in this section, we develop and implement a time-discrete variational principle for hyperelastic structure equations thus making it compatible with our time-discrete variational principle for numerical wavetank.

3.5.1 Variational modelling of hyperelastic beam for solving FSI problems in the numerical wavetanks

We start by considering the variational principle for the hyperelastic beam model developed by Salwa[72] (Chapter 2), which is given as

$$0 = \delta \int_0^T \iiint_{\Omega_O} \rho_0 \mathbf{U} \cdot \partial_t \mathbf{X} - \frac{1}{2}\rho_0 |\mathbf{U}|^2 - \rho_0 g Z - W(\underline{\mathbf{E}})\, \mathrm{d}a\, \mathrm{d}b\, \mathrm{d}c\, \mathrm{d}t, \tag{3.34}$$

where $\rho_0(\boldsymbol{a})$ is the material density, $\frac{1}{2}\rho_0(\boldsymbol{a})|\boldsymbol{U}|^2$ is the kinetic energy which depends on the velocity vector $\boldsymbol{U} \equiv \partial \boldsymbol{X}/\partial t$, gZ is the potential energy, and the internal energy $W(\underline{\mathbf{E}})$ depends on the position vector X, is given as

$$W(\underline{\mathbf{E}}) = \frac{1}{2}\lambda\, |tr(\underline{\mathbf{E}})|^2 + \mu\, tr(\underline{\mathbf{E}}^2). \tag{3.35}$$

Lamé parameters λ and μ are used to define the material properties. We consider a rectangular beam in the Eulerian framework in which Eulerian coordinates in space and time $\mathbf{X} = \mathbf{X}(a, b, c, t) = (X, Y, Z)^T = (X_1, X_2, X_3)^T$ are defined in terms of Lagrangian coordinates $\mathbf{a} = (a, b, c)^T = (a_1, a_2, a_3)^T$ and time t. The schematic of the beam is shown in Fig. 3.12.

The variations of (3.34) is taken with respect to velocity U and position X and end-point condition i.e., $\delta\mathbf{X}(\mathbf{a}, 0) = \delta\mathbf{X}(\mathbf{a}, T) = 0$ is applied while considering the Lagrangian-Green stress tensor $E_{ij} = \frac{1}{2}(F_{ki}F_{kj} - \delta_{ij}) = E_{ji}$, we have $E_{ij}(F_{ki}\delta F_{kj} + F_{kj}\delta F_{ki}) = 2E_{ij}F_{ki}\delta F_{kj}$ with

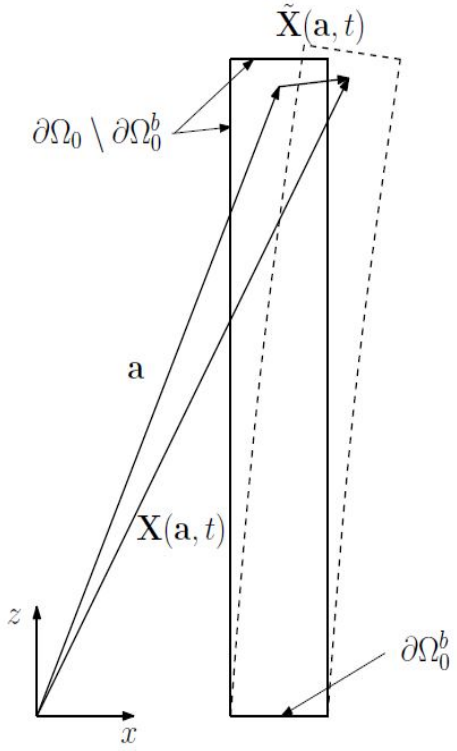


Figure 3.12: Cross section of a beam in the x–z plane is shown [72]. The solid line shows the reference state and the dotted line shows the position after deformation; $\mathbf{a} = \mathbf{X}(\mathbf{a}, 0)$ is the Lagrangian coordinate in the reference state while $\tilde{\mathbf{X}}(\mathbf{a}, t)$ in the deformed state. The movable boundary is denoted by $\partial\Omega_O = \partial\Omega_0$ while the fixed bottom is denoted by $\partial\Omega_O^b = \partial\Omega_0^b$.

deformation gradient $F_{kj} = \partial X_k/\partial a_j$, to obtain

$$\begin{aligned} 0 = \int_0^T \iiint_{\Omega_O} \rho_0(\partial_t \mathbf{X} - \mathbf{U}) \cdot \delta\mathbf{U} - \left(\rho_0 \partial_t U_l + \rho_0 g \delta_{3l} - \frac{\partial T_{li}}{\partial a_i}\right) \delta X_l \,\mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c\,\mathrm{d}t \\ - \int_0^T \iint_{\partial\Omega_O/\partial\Omega_O^b} n_i T_{li} \delta X_l \,\mathrm{d}S\,\mathrm{d}t, \end{aligned} \tag{3.36}$$

where $T_{li} \equiv \lambda \mathrm{tr}(\underline{\mathbf{E}}) F_{li} + 2\mu E_{ki} F_{lk}$ is stress tensor and n_i is the outward normal component. The equations of motions are obtained after using the arbitrariness of variations, as follows

$$\delta\mathbf{U}: \quad \partial_t \mathbf{X} = \mathbf{U} \quad \text{on } \Omega_O \tag{3.37a}$$

$$\delta X_l: \quad \rho_0 \partial_t U_l = -\rho_0 g \delta_{3l} + \frac{\partial T_{li}}{\partial a_i} \quad \text{on } \Omega_O \tag{3.37b}$$

$$\delta X_l|_{\partial\Omega_O/\partial\Omega_O^b}: \quad 0 = n_i T_{li} \quad \text{on } \partial\Omega_O/\partial\Omega_O^b. \tag{3.37c}$$

Hence, on the fluid-beam interface $a = L_s$, where L_s is the location of interface, with outward

normal $\hat{\mathbf{n}} = (-1, 0)$ and $c \in [0, c_s]$, the boundary equations are

$$0 = T_{l1}. \tag{3.37d}$$

The classical approach for VP implementation

After describing the mathematical model for hyperelastic structures, we describe the traditional approach for implementing the variational problem. In the classical approach, the equations are implemented in 'Firedrake' as weak formulations which are derived after taking the variations of the variational principle without using the integration by parts. The weak formulations for the hyperelastic beam are derived as

$$\delta X_k : \iiint_{\Omega_O} \rho_0 \partial_t U_k \delta X_k + \rho g \delta X_k \delta_{3k} + [\lambda \mathrm{tr}(\underline{\mathbf{E}}) F_{kj} + 2\mu E_{ij} F_{ki}] \frac{\partial(\delta X_k)}{\partial a_j} \,\mathrm{d}a \,\mathrm{d}b \,\mathrm{d}c = 0 \tag{3.38a}$$

$$\delta U_k : \iiint_{\Omega_O} \rho_0 (\partial_t X_k - U_k) \delta U_k \,\mathrm{d}a \,\mathrm{d}b \,\mathrm{d}c = 0. \tag{3.38b}$$

The function space to numerically solve the functions, i.e. X_k, $U_k, \delta X_k$, and δU_k, is the first order piecewise-linear continuous Galerkin (CG). The advantage of using CG function space is that it is a higher-order, local, easy to use with conforming adaptivity and to formulate Schur complement. The conventional approach to solving the equations in 'Firedrake' is to use the time discrete weak formations. The time integrator can either be based on the first-order Symplectic-Euler scheme or second-order Störmer-Verlet scheme. In the symplectic-Euler scheme, the updated velocity/ velocity at the next time step U^{n+1} of the beam is calculated by using the position at the current time step X^n as follows:

$$\iiint_{\Omega_O} \rho_0 \frac{(U_k^{n+1} - U_k^n)}{\Delta t} \delta X_k + \rho_0 g \delta X_k \delta_{3k} + [\lambda \mathrm{tr}(\underline{\mathbf{E}}^n) F_{kj}^n + 2\mu E_{ij}^n F_{ki}^n] \frac{\partial(\delta X_k)}{\partial a_j} \,\mathrm{d}a \,\mathrm{d}b \,\mathrm{d}c = 0 \tag{3.39a}$$

using the shorthand notation

$$E_{ij}^n = \frac{1}{2}(F_{ki}^n F_{kj}^n - \delta_{ij}), \quad F_{kj}^n = \partial X_k^n / \partial a_j. \tag{3.39b}$$

Then the position at the next time step X^{n+1} is calculated by using velocity U^{n+1} as follows

$$\iiint_{\Omega_O} \rho_0 \frac{(X_k^{n+1} - X_k^n)}{\Delta t} \delta U_k \,\mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c = \iiint_{\Omega_O} \rho_0 U_k^{n+1} \delta U_k \,\mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c. \tag{3.39c}$$

The novel approach for VP implementation

The second/novel approach is to solve the system of equations by using the time-discrete variational principle instead of time-discrete weak formations and use a 'Firedrake' function '*fd.derivative*' to take variations of the time-discrete variational principle with respect to $\delta\mathbf{U}^{n+1}$ and $\delta\mathbf{X}^n$ to obtain the time-discrete weak formations implicitly. This approach makes the implementation process efficient as time to type the equations and chances of human error are reduced. The time-discrete adjoint formulation is derived as follows (3.40)

$$\begin{aligned} 0 = \delta \iiint_{\Omega_O} & -\rho_0 \mathbf{X}^n \cdot \frac{(\mathbf{U}^{n+1} - \mathbf{U}^n)}{\Delta t} + \rho_0 \mathbf{X}^{n+1} \cdot \frac{\mathbf{U}^{n+1}}{\Delta t} \\ & - \frac{1}{2}\rho_0 |\mathbf{U}^{n+1}|^2 - \rho_0 g Z^n - W(\underline{\mathbf{E}}^n) \,\mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c \quad \text{with} \\ \underline{\mathbf{E}}^n = & \underline{\mathbf{E}}(\mathbf{X}^n) \end{aligned} \tag{3.40}$$

As a check, the variations of (3.40) with respect to $\delta\mathbf{U}^n$ and $\delta\mathbf{X}^{n+1}$ must yields the time-discrete weak formulations given in (3.39). The variations of (3.40) with respect to $\delta\mathbf{U}^n$ yields

$$\iiint_{\Omega_O} \rho_0 \frac{(U_k^{n+1} - U_k^n)}{\Delta t} \delta X_k + \rho_0 g \delta X_k \delta_{3k} + [\lambda \mathrm{tr}(\underline{\mathbf{E}}^n) F_{kj}^n + 2\mu E_{ij}^n F_{ki}^n] \frac{\partial(\delta X_k)}{\partial a_j} \,\mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c = 0 \tag{3.41a}$$

using the shorthand notation

$$E_{ij}^n = \frac{1}{2}(F_{ki}^n F_{kj}^n - \delta_{ij}), \quad F_{kj}^n = \partial X_k^n / \partial a_j, \tag{3.41b}$$

and variations with respect to $\delta\mathbf{X}^n$ yields

$$\iiint_{\Omega_O} \rho_0 \frac{(X_k^{n+1} - X_k^n)}{\Delta t} \delta U_k \,\mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c = \iiint_{\Omega_O} \rho_0 U_k^{n+1} \delta U_k \,\mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c, \tag{3.41c}$$

which proves that the second approach is mathematically equal to the first approach.

3.5.2 Hyperelastic beam with viscous structural damping

It is noticed that solving the set of equations (3.38) yields oscillating in the beam; therefore, the weak formulation (3.38a) is modified as follows

$$\delta X_k : \iiint_{\Omega_O} \rho_0 \partial_t U_k \delta X_k + \rho g \delta X_k \delta_{3k} + [\lambda \mathrm{tr}(\underline{\mathbf{E}}) F_{kj} + 2\mu E_{ij} F_{ki}] \frac{\partial(\delta X_k)}{\partial a_j} + \kappa \rho_0 U_k \delta X_k \,\mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c = 0, \tag{3.42a}$$

in which an additional damping term is added with a damping coefficient $\kappa(t) > 0$ for testing purposes. The damping term is introduced in the equations so that the oscillations of the beam can be numerically damped in an exponential manner to achieve the lithostatic state of rest. The numerical damping is essential to obtain the lithostatic state of rest for the beam because the coupled FSI problem involves dealing with the complex moving interface between the water-free surface and hyperelastic beam; as a result, it is important to use the lithostatic state as the initial condition for the efficient and accurate numerical treatment of the water and beam movements. The equations for the first-order symplectic-Euler time scheme with the damping term $\kappa(t)$ are stated as follows:

$$\iiint_{\Omega_O} \rho_0 \frac{(U_k^{n+1} - U_k^n)}{\Delta t} \delta X_k + \rho_0 g \delta X_k \delta_{3k} + [\lambda \mathrm{tr}(\underline{\mathbf{E}}^n) F_{kj}^n + 2\mu E_{ij}^n F_{ki}^n] \frac{\partial(\delta X_k)}{\partial a_j} + \frac{1}{2} \kappa \rho_0 (U_k^{n+1} + U_k^n)\, \delta X_k \,\mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c = 0 \tag{3.43a}$$

$$\iiint_{\Omega_O} \rho_0 \frac{(X_k^{n+1} - X_k^n)}{\Delta t} \delta U_k \,\mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c = \iiint_{\Omega_O} \rho_0 U_k^{n+1} \delta U_k \,\mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c, \tag{3.43b}$$

employing the shorthand notation

$$E_{ij}^n = \frac{1}{2}(F_{ki}^n F_{kj}^n - \delta_{ij}), \quad F_{kj}^n = \partial X_k^n / \partial a_j. \tag{3.43c}$$

3.5.3 Implementation in Firedrake

The first-order symplectic Euler time discretisation scheme is employed to solve the system of equations. At first, the equation (3.44) is solved – with the damping term in it – to calculate the updated version of velocity U^{n+1}. In the code, the terms are calculated sequentially before reaching the final step of calculating updated velocity U^{n+1} and then updated position X^{n+1}. Firstly, energy ($W(E) = \frac{1}{2}\lambda\,|tr(\underline{\mathbf{E}})|^2 + \mu\, tr(\underline{\mathbf{E}}^2)$) is computed for the nonlinear case by using

the appropriate value of Green Lagrange strain tensor (E) as stated in (3.43c). At this stage, the value of energy is scalar. In the next step, this scalar is converted into '*Foam*', which is a term used by Firedrake to denote that something has been integrated over the spatial domain. The step which follows after this is the derivative of the obtained '*Foam*' with respect to spatial coordinates denoted by X. It is done by using an inbuilt function of Firedrake denoted as '*fd.derivative*' in the software. This function automatically derives the time-discrete weak formulations in (3.44) and (3.45). After this step, the damping term is also converted into '*Foam*' by integrating it over the domain and then added to the previously calculated expression. The expression with the damping term is given in (3.43a), after separating the U^{n+1} terms from U^n terms and denoting the $\frac{1}{2}\lambda\,|tr(\underline{\mathbf{E}})|^2 + \mu\,tr(\underline{\mathbf{E}}^2)\,\mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c$ by a shorthand notation 'F_{expr}', the rearranged version of (3.43a) can be written as follows:

$$\iiint_{\Omega_O} U_k^{n+1}\delta X_k\,\mathrm{d}V = \iiint_{\Omega_O} \frac{2}{2+\Delta t\kappa}\Big[U_k^n\delta X_k - \frac{\Delta t}{2\rho_0}\Big(\rho g\delta X_k + F_{expr} + \frac{1}{2}\kappa U_k^n\delta X_k\Big)\Big]\,\mathrm{d}V. \tag{3.44}$$

It is worth noting, that when the damping coefficient κ is equal to zero it yields a time-discrete weak formulation of the equation of motion given in eq (3.43a). The updated value of velocity is then used to calculate the updated value of the coordinates of the beam, as follows:

$$\iiint_{\Omega_O} X_k^{n+1}\delta U_k\,\mathrm{d}V = \iiint_{\Omega_O} X_k^n\delta U_k + \Delta t\Big(U_k^{n+1}\delta U_k\Big)\,\mathrm{d}V. \tag{3.45}$$

When the above-mentioned set of equations is solved without a damping term, the solution shows oscillations in the beam because the constitutive laws for the nonlinear hyperelastic beam do not include a damping term and allow the beam to deform under its weight. Hence, these oscillations are damped numerically so the lithostatic state can be achieved. At the lithostatic state, the beam is at rest which means the outward normal force and the external normal force acting on the beam's surface are in balance, which is given by the following expression

$$0 = -\rho_0 g\delta_{3l} + \frac{\partial T_{li}}{\partial a_i} \quad \text{on } \Omega_O. \tag{3.46}$$

Once the weak formulations are derived, the next step is to select suitable parameters i.e. mesh size and time step Δt as they are crucial for the stability of the numerical results. For the symplectic-Euler scheme, the restriction on time step size is determined by the frequency 'ω',

which is given by the stability criterion as follows [72]:

$$|\omega \Delta t| \leq 2, \tag{3.47}$$

where $\omega = k\sqrt{(\mu + 2\lambda)/\rho_0}$ and $k = 2\pi\sqrt{1/\Delta a^2 + 1/\Delta b^2 + 1/\Delta c^2}$, while Δa, Δb and Δc are the smallest distance between mesh nodes in a, b and c directions, respectively. As a result, the stable time step for the scheme depends on the mesh resolution and it is computed as:

$$\Delta t = CFL(2/\omega), \tag{3.48}$$

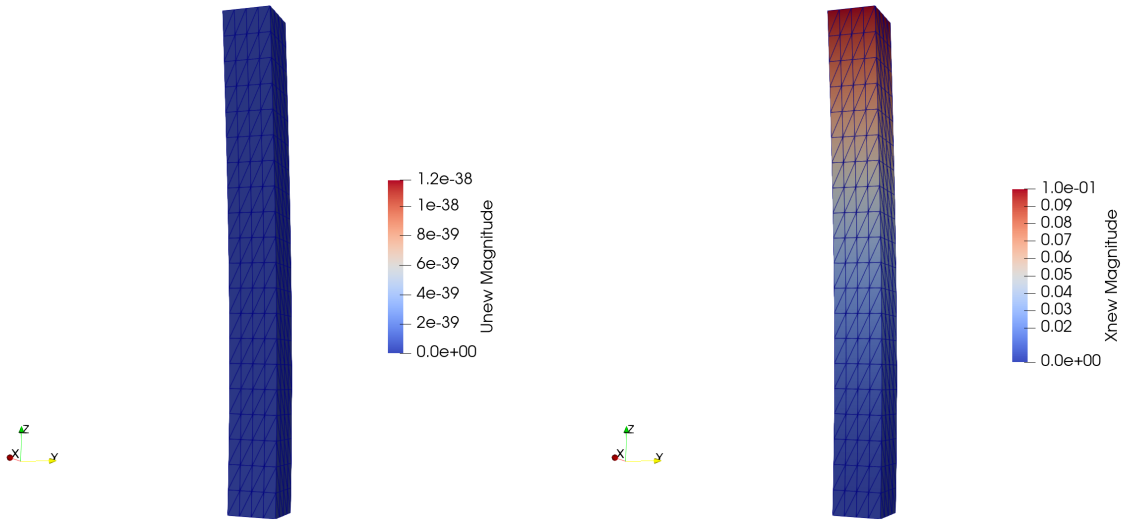
where for the Courant-Friedrichs-Lewy (CFL)-number we take $CFL \leq 1$. The simulation has been run with and without the damping term by setting the value of damping-coefficient κ equal to three and zero, respectively. To ensure that the results have reached a steady state, the comparison of both kinetic and potential energy evolution has been done by using different time durations. The comparison showed that the energy got stable after 1.8 seconds. Hence, the final time of 1.8 seconds is chosen to do further simulations.

3.5.4 Results and discussion

Comparison of classical and novel approaches for VP implementation

Next, we implement both approaches, i.e. the time-discrete weak formulations (traditional) and time-discrete VP (novel), in 'Firedrake' to compare the results. The codes for the implementation of the time-discrete VP problem are based on the extension of Salwa's [72] codes for the implementation of time-discrete weak formulations. The initial conditions for the velocity U and position X are depicted in Fig. 3.13.

A comparison is first made of the beam's velocity U and position X by solving the VPs obtained using two approaches: "case 1", a time-discrete VP (3.40), and "case 2", a weak formulation of equations of motions (3.38). This comparison is done in order to prove that both approaches are equivalent in that they yield results that agree with each other to machine error. The equations in both approaches are solved using the Symplectic-Euler time scheme, and quantification of the comparison is shown in Fig. 3.14.



(a) Initial condition for U at $t = 0$.

(b) Initial condition for X at $t = 0$.

Figure 3.13: The initial conditions for (a) velocity U and (b) displacement X.

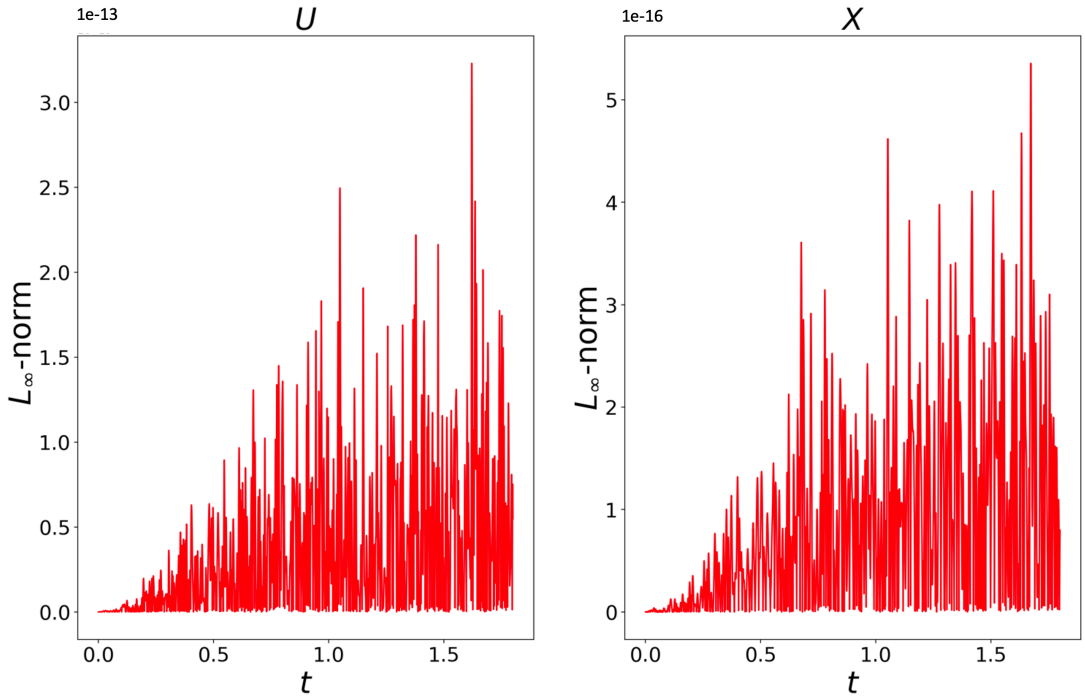
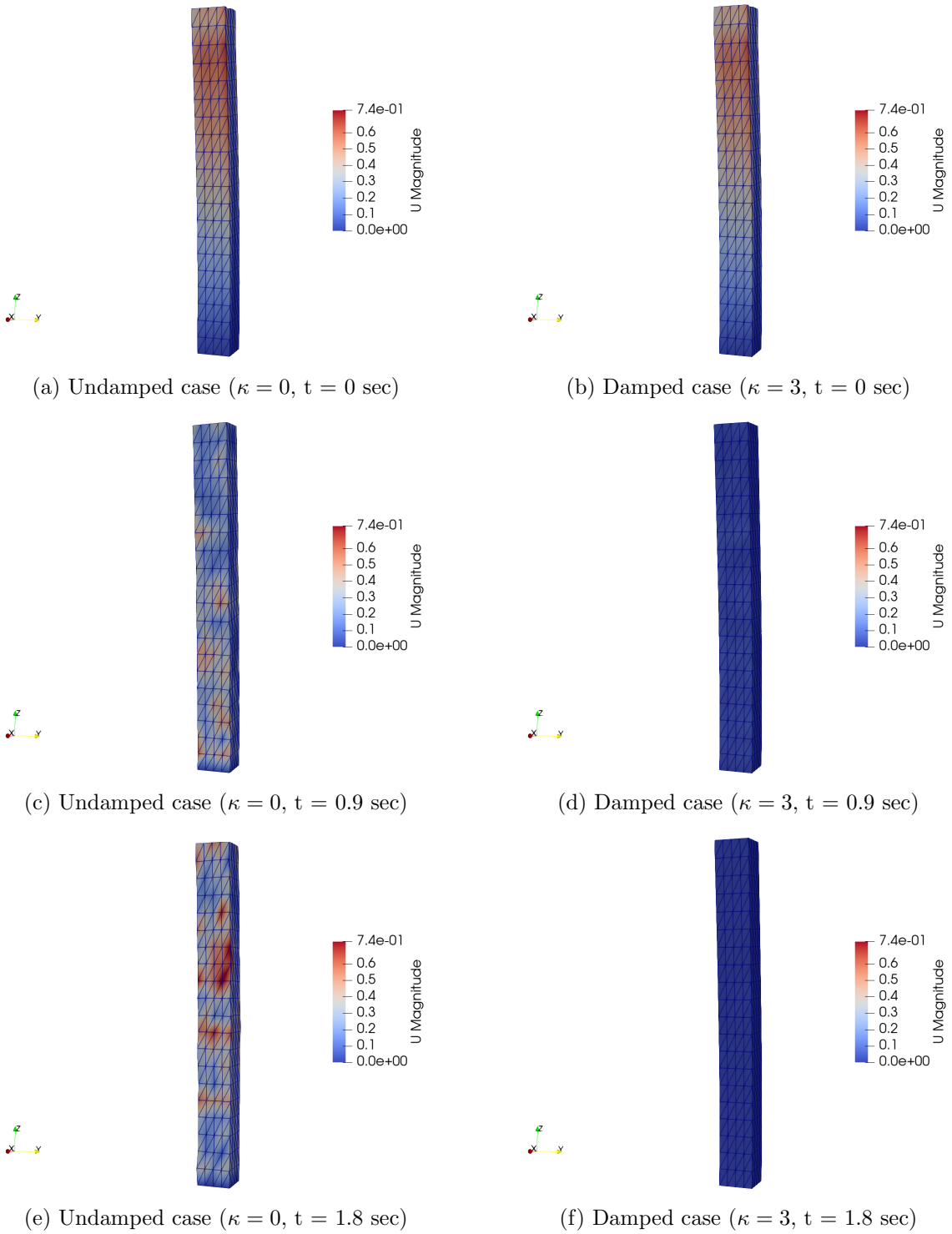


Figure 3.14: Time evolution of L_∞–norms of the difference, at a single label/Lagrangian point **a**, between numerical simulations for U (left) and X (right), generated for case 1 and case 2, of the 3D hyperelastic beam. The graphs confirm that the results of the two set-ups are, as expected, equivalent to within machine precision. Vertical axes display multiples of 10^{-13} and 10^{-16} in left- and right-hand plots respectively.

Since Fig. 3.14 demonstrates that the L_∞–norm of the difference between the results at one point computed from cases 1 and 2 is of the order of machine precision, it is confirmed that the two approaches are (numerically) equivalent.

Comparison of beam's equations with and without structural damping

The comparison of oscillations in the beam dynamics with and without damping (κ) at different times, i.e. initial, middle and final, is shown in Fig. 3.15.



(a) Undamped case ($\kappa = 0$, t = 0 sec)

(b) Damped case ($\kappa = 3$, t = 0 sec)

(c) Undamped case ($\kappa = 0$, t = 0.9 sec)

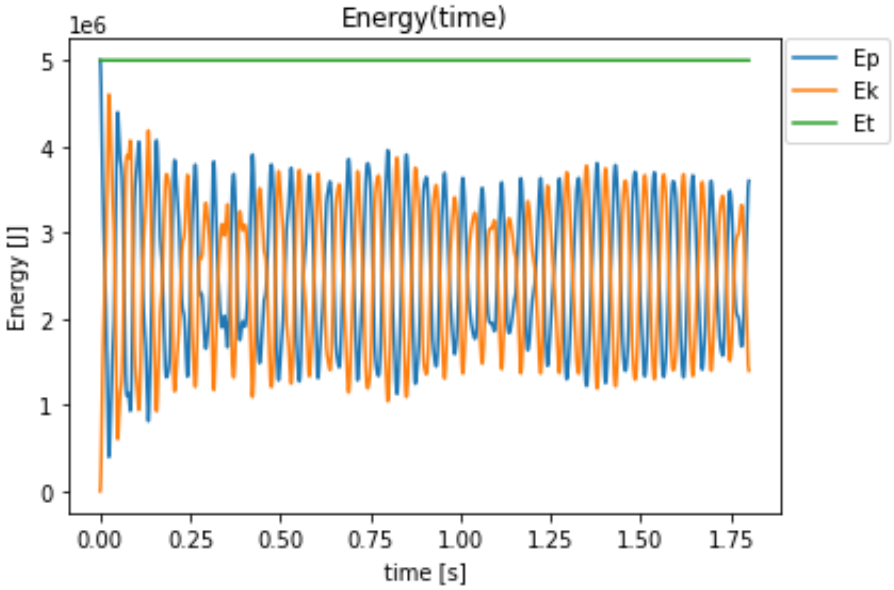
(d) Damped case ($\kappa = 3$, t = 0.9 sec)

(e) Undamped case ($\kappa = 0$, t = 1.8 sec)

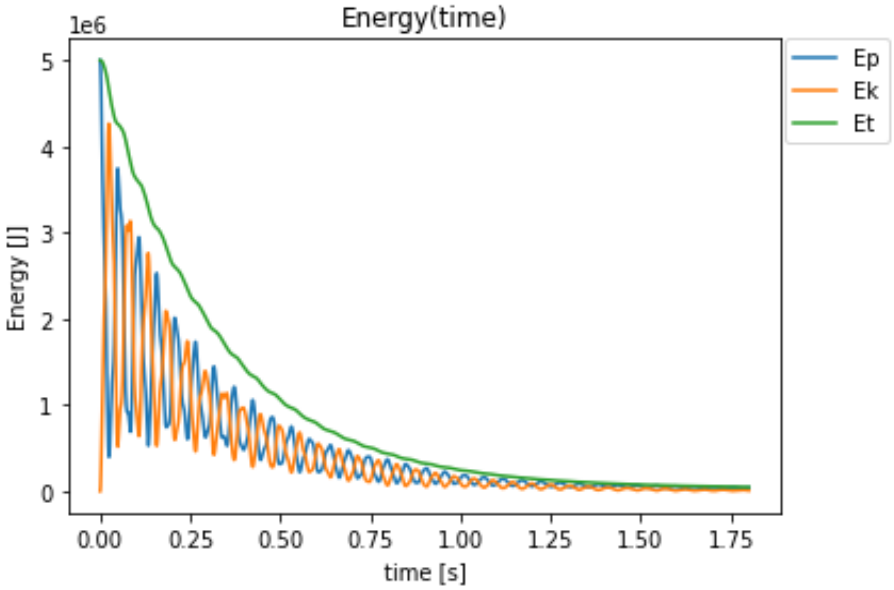
(f) Damped case ($\kappa = 3$, t = 1.8 sec)

Figure 3.15: Comparison of oscillations (magnitude of velocity) of the beam with and without using the damping coefficient κ. The results are shown at the final time step.

It is noticed that the oscillations have damped and the beam has reached the lithostatic state of rest which can be the initial condition for the coupled FSI problem. This case has been run without assuming any hydrostatic force acting on the beam. The energy plots for the energy evolution in both cases are shown in Fig. 3.16.



(a) Energy evolution in the undamped oscillations case ($\kappa = 0$)



(b) Energy evolution in the damped oscillations case ($\kappa = 3$)

Figure 3.16: Energy evolution plots over time in the case of undamped and damped oscillations of the beam are shown. Energy is fluctuating in a periodic manner in the undamped case while in the case of damped oscillations, energy converging to zero.

The comparison of energy evolution for the damped with the undamped case shows that as the oscillations of the beam are damping over time the fluctuations of energy are also reducing and

converging to zero.

The expression given in (3.50) can compute the hydrostatic pressure acting on the beam. Integrating the hydrostatic pressure along the beam's surface will convert it into force. As this pressure is the function of height which increases with the water depth; therefore, greater compressive force will be acting at the base of the mast as compared to the free surface. In this case, the lithostatic state would only be achieved when the outward component of the stress tensor at the beam's surface T^b_{li} is equal to the outward component of the stress tensor by the water T^w_{li}. Thus, when water is interacting with the beam at $a = L_s$, the boundary equation is

$$n_i T^b_{li} = n_i T^w_{li} \qquad \text{for} \qquad l, i = 1, 2, 3 \tag{3.49}$$

where T^w_{li} is defined as follows

$$\begin{aligned} T^w_{li} &= \rho g(H_0 - Z(L_s, c, t))\delta_{li}\Theta(H_0 - Z(L_s, c, t)) \\ &= \begin{cases} \rho g(H_0 - Z(L_s, c, t))\delta_{li}, & \text{if } Z \leq H_0 \\ 0, & \text{if } Z > H_0 \end{cases} \quad \text{on } \partial\Omega_O \text{ with } c \in [0, L_z > H_0]. \end{aligned} \tag{3.50}$$

To balance the increasing hydrostatic pressure T^w_{li}, the outward normal stress tensor T^b_{li} must increase with the water depth which is taken into account by adding a time-dependant variable $Z(L_s, c, t))$ in (3.50). Notice that both cases, i.e. with and without considering the hydrostatic pressure would yield a lithostatic state of the beam. However, solving the problem with hydrostatic pressure is significant because it can verify the results for the water-beam coupled case when considering that water is at rest.

3.6 Conclusion

We have developed a piston-driven numerical wavetank model to simulate intermediate- and deep-water dynamics by utilising our novel spatio-temporal discrete finite-element numerical schemes for potential-flow wave dynamics based on variational principles (VPs), including extended formulations of these principles in which waves are driven by piston wavemaker at one end of the wavetank. The finite-element methods have been implemented in the finite-element environment Firedrake, whose intrinsic domain-specific compiler-architecture has, through valida-

tion of automated derivations, allowed us to demonstrate a reduction in time-to-implementation, thereby offering promise to extend the approach to more complex coupled water-wave dynamics with FSI. A novel feature is that we have developed and tested the numerical implementations of the dynamics via consistent (i.e. stable and conservative) geometric time-discrete VPs, hence the discrete evolution equations are generated automatically within Firedrake and subsequently solved via its inbuilt (non)linear solvers. Preliminary numerical tests are therefore promising, and a full and further optimised implementation – through a time-discrete VP and verification/validation of potential-flow dynamics driven by waveflaps – is in progress.

Chapter 4

Experimental modeling of water-wave interactions with a flexible beam

4.1 Introduction

Fixed offshore wind turbines (FOWT) are considered an attractive alternative to onshore wind turbines because offshore wind flow is stronger and steadier than on land. Offshore installation additionally circumvents problems related to land availability, noise and interference with communication signals [49, 65]. However, building FOWT farms is capital-intensive as the design, fabrication and installation of structures in often-harsh ocean conditions are challenging. In addition, FOWTs are prone to higher risks of structural damage because they are larger than onshore wind turbines and have to endure hydrodynamic loading in addition to aerodynamic loading [45]. It is clearly of great importance to predict such loading accurately, which demands a better understanding of the physics of water-wave interactions with a fixed-bottom flexible monopile. The problem of water-wave interactions with such a beam is a complex multiphysics phenomenon known as a fluid-structure interaction (FSI). In FSI problems, the fluid flow interacts with the flexible structure in a way that deforms the structure and, as a result, the structural deformations change the initial fluid flow. Thus a FSI problem is a coupled, two-way problem of which, due to the complexity of the underlying physics, investigation is challenging in terms of experimentation, mathematical analysis and numerical modelling.

In the maritime industry, mathematical and numerical modelling is gaining significance because experimental scaled-model testing is not always feasible in early design stages due to time and budgetary constraints. Moreover, the experimental modelling of flexible structures at the model scale is not straightforward, motivating researchers to develop mathematical and numerical models for solving FSI problems. These models generally fall into two categories. First, they range from straightforward linear shallow-water equations and linear modal analysis to intermediate-complexity linear potential-flow solvers coupled to linear elastic structural equations [72, 71]. Second, there are more sophisticated approaches based on nonlinear potential flow, Navier-Stokes (NS) equations [86], and Smoothed Particle Hydrodynamics (SPH) [22] coupled with nonlinear hyperelastic structural equations. However, results generated by numerical models require validation using benchmark experimental data. The present research therefore concerns wave-basin experiments of FSI problems; specifically, the dynamic response of a flexible beam exposed to (controllably generated) water waves. The study's objective is the generation of a high-quality experimental data set to be used to validate diverse numerical models for solving FSI problems i.e., linear, nonlinear and high-fidelity.

The experimental set-up includes a vertically mounted flexible cylindrical beam equipped with six accelerometers, distributed evenly along its length, that record its dynamic response. Two probes placed at the free surface of the water close to the beam (forward of and to the side of the beam) measure the free-surface elevation of the incident and reflected water waves. The beam is fixed to a basin carriage that traverses horizontally along the wavetank at different speeds to control the frequency with which waves encounter the beam. The upper and lower parts of the beam are respectively in air and submerged in water. This model set-up was prompted by the basin depth (3.6m) which excludes the possibility of modelling a bottom-mounted beam. The FSI physics therefore do not exactly resemble those of a FOWT but have sufficient similarity to provide suitable validation material for FSIs of a FOWT. For example, a numerical model of the exact experimental setup can be created and validated with the experimental data and then the direction of gravitational acceleration and water loads can be reversed to match the physics with the actual FOWT model. Researchers [77, 3] have also studied experimentally the dynamic response of a bottom-fixed monopile turbine in waves. Note that we cover a very wide range of sea conditions ranging from regular-to-irregular and moderate-to-extreme wave height and steepness. The study of such a wide range of conditions makes the experiments suitable for providing reliable data in validating a suite of mathematical and numerical FSI solvers,

i.e., linear, nonlinear and high-fidelity. Furthermore, the data from the experiments are made publicly available through open-source data-sharing platforms.

Hammer tests have been performed on the beam in air and water to obtain the dry and wetted modes, natural frequencies and structural and hydrodynamic damping. The novelty of the experimental set-up is that it allows simultaneous measurement of beam deflections and their effect on the incident and reflected waves, rendering feasible a study of the FSI problem in diverse-yet-controllable conditions.

The experiments are divided into three cases, each of which is aimed at studying the dynamic response of the flexible beam to varying wave conditions ranging from regular-to-irregular and moderate-to-extreme wave height and steepness. Experimental Case 1 concerns interactions of regular waves with the flexible beam when the carriage is at rest; studying this case will facilitate the validation of linear FSI solvers in the non-resonant regime, since the non-linear dynamic response of beam is not excited by the incident-wave frequencies. Experimental Case 2 concerns interactions with the flexible beam when the carriage is moving at a constant speed. Moving the carriage changes the frequency of encounter between beam and waves, so that the dynamic response of the beam and its interaction with water waves, particularly at the onset of resonance, can be studied. By changing the steepness of the regular waves, both linear and nonlinear FSI solvers can be validated. In this case, the dynamic response of the beam results from an accumulated hydrodynamic loading that cannot be distinguished, by the current experimental set-up, into its consistuent wave- and current-induced components. Experimental Case 3 concerns steep, irregular-wave interactions with the flexible beam when the carriage is at rest. This is the most complex case and is designed to yield data on structural dynamics due to nonlinear wave-loading processes related to steep and breaking waves. This case will help to validate the high-fidelity FSI solvers.

Hence, the study covers a wide range of FSI problems that can be used to establish benchmarks for FSI-code validations.

4.2 Design of experimental set-up

The experimental set-up and laboratory facilities are now explained. The FSI set-up is designed to mimic the (simplified by neglecting the rotor effects) physics of a fixed-bottom offshore wind

turbine (OWT) mast; i.e. the focus is solely on the response of the flexible mast to water-wave loading and the concomitant changes in fluid flow due to the mast's deformations. Such a rotorless set-up will hopefully admit extensions aimed at broadening the application of the experimental data to other FSI problems; for example, in the design of vortex bladeless wind turbines [73]. Fixed-bottom OWTs occur in three forms, defined by their foundations, as shown in Fig. 4.1.

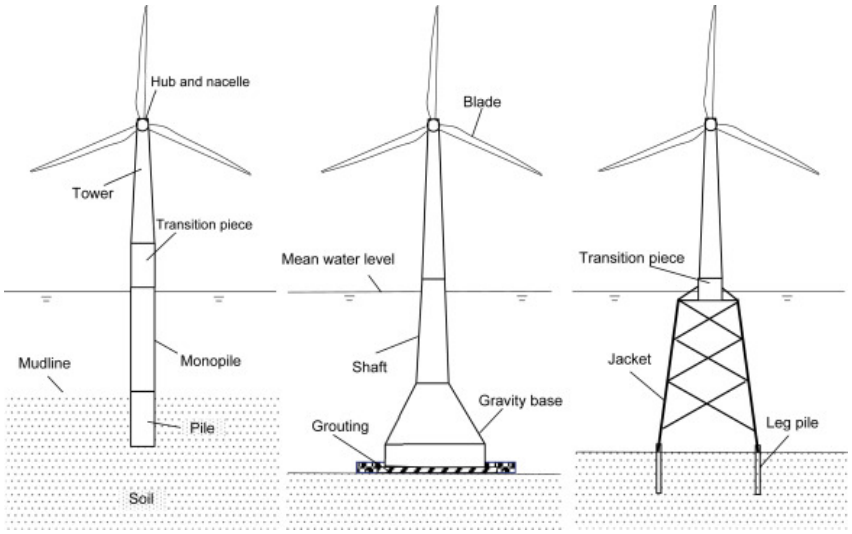


Figure 4.1: Schematic of different fixed-bottom OWT foundations; monopile, gravity-based and jacket. Copyright © 1969, Elsevier [42].

Of the three types of OWT foundation, the monopile is the subject of the present experimental study as it has the simplest design; one that comprises a single steel-tube pile. Before designing the experimental set-up, it is important to have a basic understanding of the (rotorless) dynamics of the mast of the monopile fixed-bottom OWT. Study of soil-structure interactions [7] confirms that the overturning moment generated at the mast's bottom, due to the wind and water-wave loading acting on the mast, causes angular movement of the buried (in soil) section of the foundation. Therefore, this behaviour should be incorporated into the experimental set-up.

The experimental set-up, a schematic side view of which is shown in Fig. 4.2, comprises an initially vertical flexible cylindrical beam, one (top) end of which is fixed to a basin carriage having a base made of PVC that is flexible enough to allow angular motion of the top of the beam, yet at the same time strong enough to keep the assembly intact. The other (bottom) free end – initially vertically below the fixed end – is submerged in water. The basin carriage can traverse along the basin's length at different speeds. There are six equidistant accelerometers attached along the beam's length for measuring the beam's acceleration. Five out of six

accelerometers are attached to the outer surface of the beam while the sixth one is attached to the inner surface of the submerged end of the beam. This is done to eliminate the interaction of the accelerometer with the water waves. Additionally, two probes (indicated by red discs in Fig. 4.2) are placed at the water free-surface, in the vicinity of the beam, to measure the wave elevation of the incident and reflected waves from the beam; the two probes are located (x, y, z) = (26.25, 1.475, 3.6)m and (30, 1.475, 3.6)m from the wavemaker, where x is the distance along the length of the wavemaker and y shows the distance in lateral direction from the centre of the wavemaker. This set-up admits simultaneous measurement of beam deflections and their effect on the incident and reflected waves and hence facilitates a quantifiable study of the FSI problem in a controlled environment.

Mathematically the experimental setup is analogous to a wind turbine's mast, when the user applies the hydrodynamic loads at the beam's fixed-end instead of the free-end. The experimental setup resembles the fixed-bottom monopile offshore wind turbine's mast which experiences the overturning moment generated at the mast's bottom, due to the wind and water-wave loading acting on the mast, causes angular movement of the buried (in soil) section of the foundation. Therefore, this behaviour is incorporated into the experimental setup by attaching a PVC base plate which allows the angular movement of the beam at the fixed end.

Experiments are conducted in the concept-design basin at the Maritime Research Institute Netherlands (MARIN). The concept basin is a 220m-long, 4.01m-wide and 3.6m-deep rectilinear basin filled with fresh water. It has a stiff carriage that can traverse along the basin's length at a maximum speed of 10m/s. At one end of the basin, there is a flap-type wavemaker that has eight contiguous paddles. The wave generator has the capacity to generate waves up to a significant wave height of 0.55m, at a peak period of 2.3s. A schematic plan view of the basin is shown in Fig. 4.3.

First, parameters for generating a required theoretical waveform are given to the wavemaker and the waves generated experimentally are measured by probes and compared with the required waveform. The difference between the experimental and required waves is used to adjust the wavemaker in order to obtain the required wave. However, a difference of up to 5% may still accrue between desired and iterated waveforms, but this is not an issue since the undisturbed waves ulitmately used in the experiments are recorded. The wave parameters, i.e. wavelength Λ and wave height H, can be calculated using the dispersion relation for deep-water dynamics

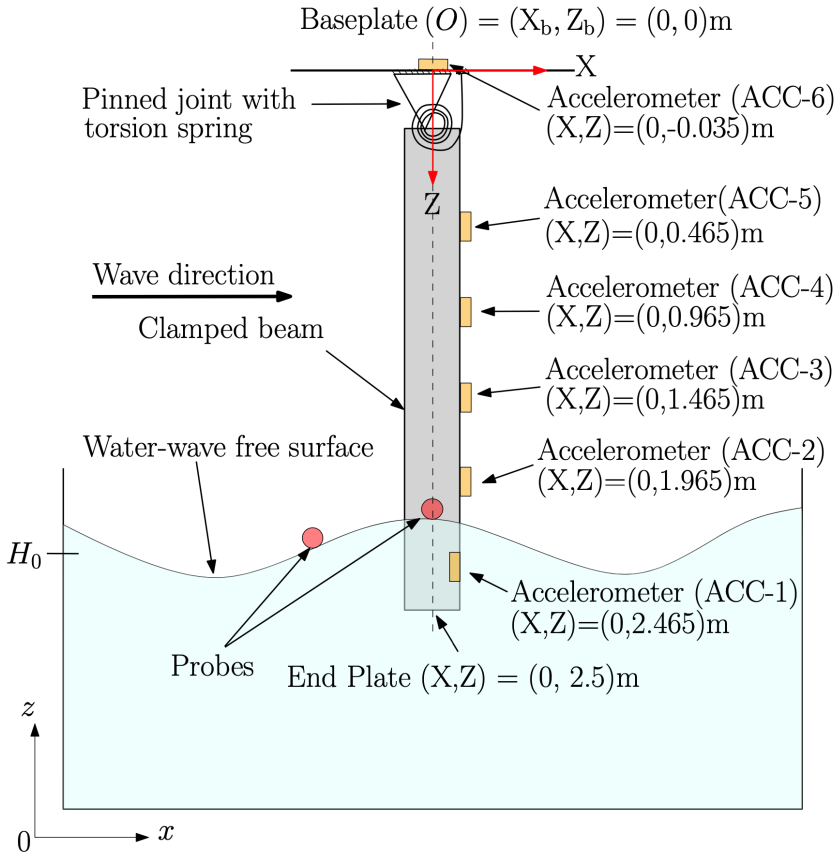


Figure 4.2: Schematic side view of the experimental set-up. An Eulerian-coordinate system (denoted by x, y and z) is used for the wavetank; its origin $(x, y, z) = (0, 0, 0)$m is located in the middle of the wavemaker at rest. A Lagrangian-coordinate system at rest (denoted by X, Y and Z) is used for the beam; its origin is at the base plate (labelled O in the figure) $(\mathrm{X}, \mathrm{Y}, \mathrm{Z}) = (0, 0, 0)$, which origin has fixed Eulerian position $(x_b, y_b, z_b) = (30, 2.05, 4.6)$m. At rest, the end plate at the free submerged end of the beam is located at $(\mathrm{X}, \mathrm{Y}, \mathrm{Z}) = (0, 0, 2.5)$m. The experiments are conducted for two submergence depths, i.e. 0.25m and 0.5m from the still-water level H_0. The base plate is flexible enough to allow rotation of the beam, represented by a pinned joint with a torsion spring. Moreover, the submerged accelerometer is internal. A more detailed CAD drawing of the set-up with exact dimensions and location of the sensors can be found on GitHub.

i.e., water depth $d > \Lambda/2$ (which applies here), given as

$$\omega^2 = gk, \tag{4.1}$$

where g is gravitational acceleration, and the wave number is $k = 2\pi/\Lambda$, so that wavelength and period are related via

$$\Lambda = \frac{g}{2\pi} T^2. \tag{4.2}$$

Formulae (4.1) and (4.2) are used to compute wave parameters Λ and ω, values of which are given in the following descriptions of experimental cases.

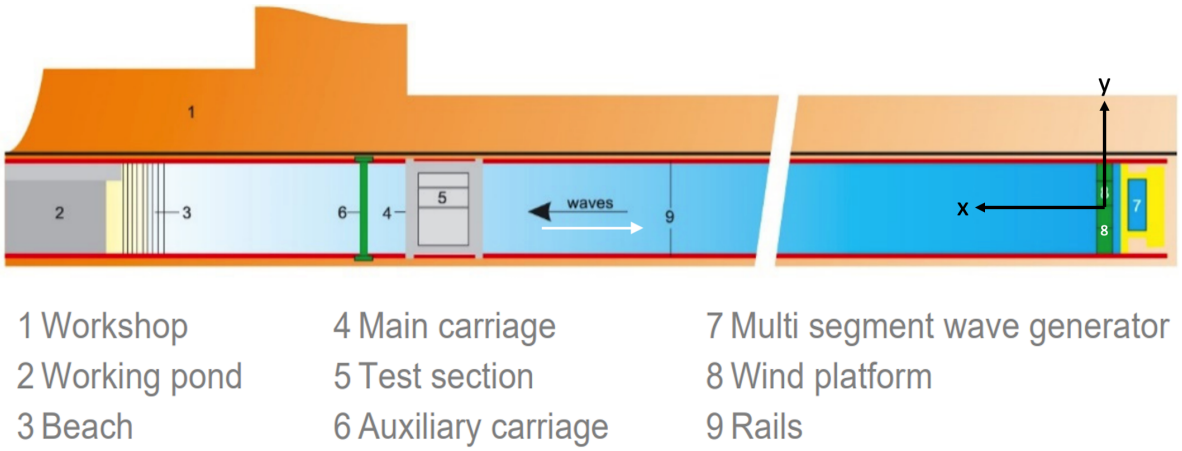


Figure 4.3: Schematic plan view of the concept wave basin at MARIN, The Netherlands [59].

4.2.1 Beam selection and procurement

The first significant experimental-set-up step is the selection of a beam flexible enough to model the FSI problem yet stiff enough to maintain a straight vertical position in the absence of external loading. After considering different material parameters, cost, and market availability, a cylindrical beam made of polyvinyl chloride (PVC) was selected. Beam dimensions were decided by calculating the natural beam frequency for different values of chosen parameters of length, wall thickness, and diameter. This parametric study is based on analysis of the horizontal cantilever beam shown in Fig. 4.4 and given as the clamped, free-beam case in [9, Table 8-1]. Note that the x, y coordinates in Fig. 4.4 differ from those used in the FSI experiments, the latter being used solely for referencing the beam geometry. Since the beam in the experimental set-up hangs vertically, any horizontal deflections from rest will be small, and the impact of water waves in the actual experiments will dominate over the restoring force of gravity, which is accordingly ignored in the analysis.

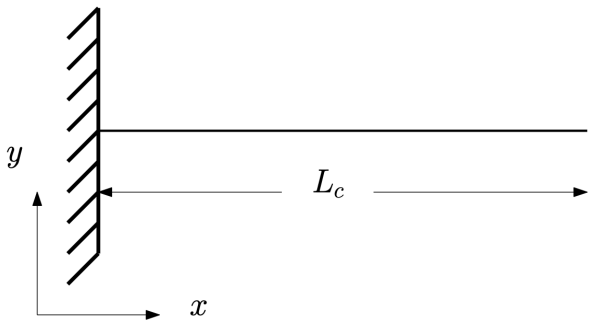


Figure 4.4: Two-dimensional view, in the x, y plane, of a one-dimensional cantilever beam of length L_c [9].

The goal being to select the values of the aforementioned parameters such that the beam's natural frequency lies outside the short, primary-wave regime. In this way, natural modes of the beam could not be excited by linear-wave effects, thereby admitting study of the nonlinear response of the beam. By moving the carriage into the waves, the wave-beam-encounter frequency of the waves can be tuned to match the natural frequency of the beam; in this way, the linear resonant response can also be studied. The empirical formula for the natural frequency f_i of the i^{th} mode of the cantilever beam is given by

$$f_i = \frac{\lambda_i^2}{2\pi L_c^2}\Big(\frac{E_c I_c}{M_c}\Big)^{1/2}; \quad i = 1, 2, 3, \ldots, \tag{4.3}$$

where L_c is the length of the cantilever beam; E_c is the elastic modulus and M_c its mass per unit length of the cantilever beam's material, and

$$I_c = \frac{\pi}{4}(a_c^4 - b_c^4) \tag{4.4}$$

is the area moment of inertia of the tubular beam with outer and inner radii a_c and b_c respectively. The modal profile $\tilde{y}$ corresponding to the i^{th}-mode of the cantilever beam is given by

$$\begin{aligned} \tilde{y}_i = &\cosh\frac{\lambda_i x}{L_c} - \cos\frac{\lambda_i x}{L_c} - \sigma_i\Big(\sinh\frac{\lambda_i x}{L_c} - \sin\frac{\lambda_i x}{L_c}\Big); \\ &i = 1, 2, 3, \ldots \end{aligned} \tag{4.5}$$

where x is the distance from the fixed end, and the eigenvalues λ_i and σ_i of the cantilever beam, corresponding to each mode number i, are real numbers calculated by Blevins [9] using the modal-analysis method of vibration response: their values for a cantilever beam are shown in Table 4.1, which is taken from [9]. That is, these formulae (4.3)-(4.5) and parameters (i.e. E_c, I_c, L_c and M_c) are for a cantilever-beam set-up that allows us to obtain initial guesses for the natural frequency and dimensional parameters of the beam that would be actually used in the study. The parameters chosen for the actual beam used in the experiments will be denoted without the subscript c, i.e. by E, I, L and M. Now the beam material has been selected, we continue considering the FSI set-up of Fig. 4.2. A PVC baseplate attached to the beam allows it to be mounted to the wavetank carriage. The baseplate additionally admits cables to be connected to the sensors in such a way that interaction with any beam displacements is

Table 4.1: Eigenvalues λ_i and σ_i of the cantilever beam, from Table 8-1 of [9].

Mode number (i)	λ_i	σ_i
1	1.87510407	0.73409551
2	4.69409113	1.01846732
3	7.85475744	0.9992245
4	10.9955407	1.00003355
5	14.1371684	0.99999855
$i > 5$	$(2i-1)\pi/2$	≈ 1

minimised as much as possible. The free (submerged) end of the beam is sealed with a PVC circular end plate such that water cannot enter the hollow beam. The described set-up is shown in Fig. 4.5.

Figure 4.5: Baseplate, wooden support, beam, accelerometers and cables of the beam. See text for details.

The masses and locations of accelerometers and end plate are given in Table 4.2. The total mass of the beam with accelerometers and baseplate is 4.66kg. The accelerometers (ACC) are numbered from 1 to 6 where ACC-1 is the accelerometer attached at the submerged free end and ACC-6 is attached at the fixed end of the beam. Calculations using these data give an effective mass per unit length of $M = 1.6048$kg/m . Parameters for the beam chosen for the

Table 4.2: Masses and locations of experimental furniture. The position of baseplate is used as a reference for the distances in the second column.

	Distance from baseplate [m]	Mass [kg]
ACC-6	-0.035	0.079
ACC-5	0.465	0.079
ACC-4	0.965	0.079
ACC-3	1.465	0.079
ACC-2	1.965	0.158
ACC-1	2.465	0.079
End plate	2.5	0.15

experiments are given in Table 4.3.

Table 4.3: Beam parameters in the FSI experiments.

Parameter	Value [mm]
Outer diameter (d_o)	125
Inner diameter (d_i)	120
Thickness ($a - b$)	2.5
Length (L)	2500

Although the purpose of these calculations is to obtain an estimate of the beam's material and dimensional parameters and dynamic response, the actual parameters and responses are better determined by performing hammer tests, as the calculations do not take into account factors such as the weight of sensors and cables, and unavoidable deviations in material properties accrued during manufacturing and fabrication processes. Hence, material parameters and dynamic responses of the beam assembly are determined by performing hammer tests, as described next.

4.3 Hammer tests on the beam

A hammer test is an experimental method for determining a structure's response and measuring its frequency response function (FRF). An impulse force is applied to excite the structure at a wide range of frequencies and the response is measured using accelerometers. The purpose of exciting the structure at a wide range of frequencies is to obtain its resonance frequencies. The obtained response can then be analysed in the frequency domain to determine dynamic parameters such as stiffness, mass and damping; modal parameters such as natural frequency and mode shapes; and, material properties of the structure. FRF, also known as the accelerance, is defined as the ratio of the output response (here accelerations) and input (impulse force) [27]. FRF therefore has dimensions of inverse mass and the units of the input and output signals determine the units of the FRF. For example, if the input signal is in units of force (N) and the

output signal is in units of acceleration (m/s^2), then the FRF will have units of kg^{-1}. However, in this article, we have computed neither FRF nor accelerance, their mention being only for information.

The response is obtained in terms of time-domain signals, here the sensor accelerations, that can be subsequently integrated to yield either velocity or displacement. The output is measured at different positions along the beam, while the input force is applied at a specific position. Hammer tests are performed to obtain the natural periods of the beam, which are used to calibrate the wave frequency required to excite the beam at that period. Exciting the beam at its natural period results in large deformations of the beam, which can be used to validate FSI solvers against nonlinear (hyperelastic) structural solvers.

Dry hammer tests of the beam assembly shown in Fig. 4.5 are conducted by lifting the beam in the air and applying an impulse force with a hammer, upon which dynamic responses (accelerations) of the beam are measured by the accelerometers. Wet hammer tests (of direct relevance to FSI studies) are performed in order to study the effect of submerged beam length on its response. It is found that the resonance time period of the beam increases with increasing submergence of the free end of the beam since the increasing submergence raises the hydrodynamic damping coefficient and added mass. The *added mass* refers to the inertia added to the system due to the fluid volume displaced by a submerged beam's motion. Based on the hammer-test study, two submergence depths, of 0.25m and 0.5m, are used in the experiments since the resonance time periods of the beam for these depths are achievable using the waveflap wavemaker at the facility.

4.3.1 Results from dry and wet hammer tests

Time-domain beam responses obtained from both dry and wet hammer tests (the latter, at two different submergence depths) are presented in Fig. 4.6. Each test comprised three hammer strikes on the beam. Hence, there are 9 peaks in total; three for each test. For the purpose of graphical comparison, the extra signal before the first peak is manually excluded so all peaks can coincide at the start of the signal. A zoomed portion of the comparison is shown in Fig. 4.6. The second blue reading is hidden behind the second red peak. Comparison of the three initial peaks appears in the expanded "early" inset in Fig. 4.6, revealing the dependence on the degree of submergence.

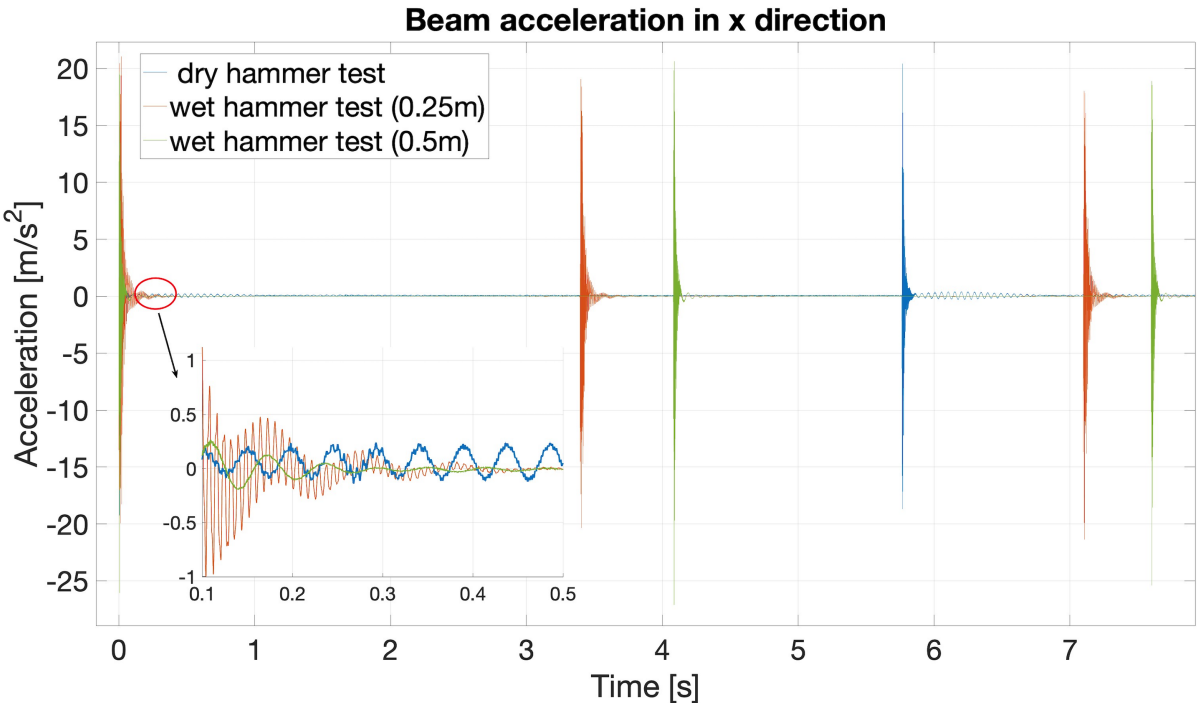


Figure 4.6: Time-domain beam responses (accelerations in x direction) for the three hammer tests. Dry (blue), wet (red, 0.25m-deep) and wet (green, 0.50m-deep) tests.

Fig. 4.7 shows the frequency-domain dynamic beam response for the three hammer tests. The peaks show the frequencies of the dominant modes of the beam for each test. The reduction in peak frequency in the hammer-test sequence dry (blue), wet (red, 0.25m-deep) and wet (yellow, 0.50m-deep) is clearly consistent with the above-mentioned increase, with submerged depth, of both damping and effective beam mass. However, the impact of the added mass surpasses that of the damping. In addition, Fig. 4.8 shows the first three modes of the beam calculated by integrating the accelerations obtained in the hammer tests. See Earth Arxived version [1].

Finally, we compute the time periods and natural frequencies of the beam responses, measured in the hammer tests, by converting the time-domain signal into the frequency domain using the MATLAB functions for Fast-Fourier Transform (FFT), Direct Fourier Transform (DFT), and Cross Spectral Density (CSD) methods. Each type of hammer test (one dry and two wet) was performed twice and the values of frequency and time period of the measured accelerations over time are shown in Table 4.4, each augmented by an error tolerance. Error tolerance is calculated by taking the standard deviation σ of the fundamental frequency which is calculated by using six values, the three "FFT"s for two repeat tests.

Table 4.4 confirms that the resonant time period of the beam increases when the beam is

[1] https://eartharxiv.org/repository/view/6586/

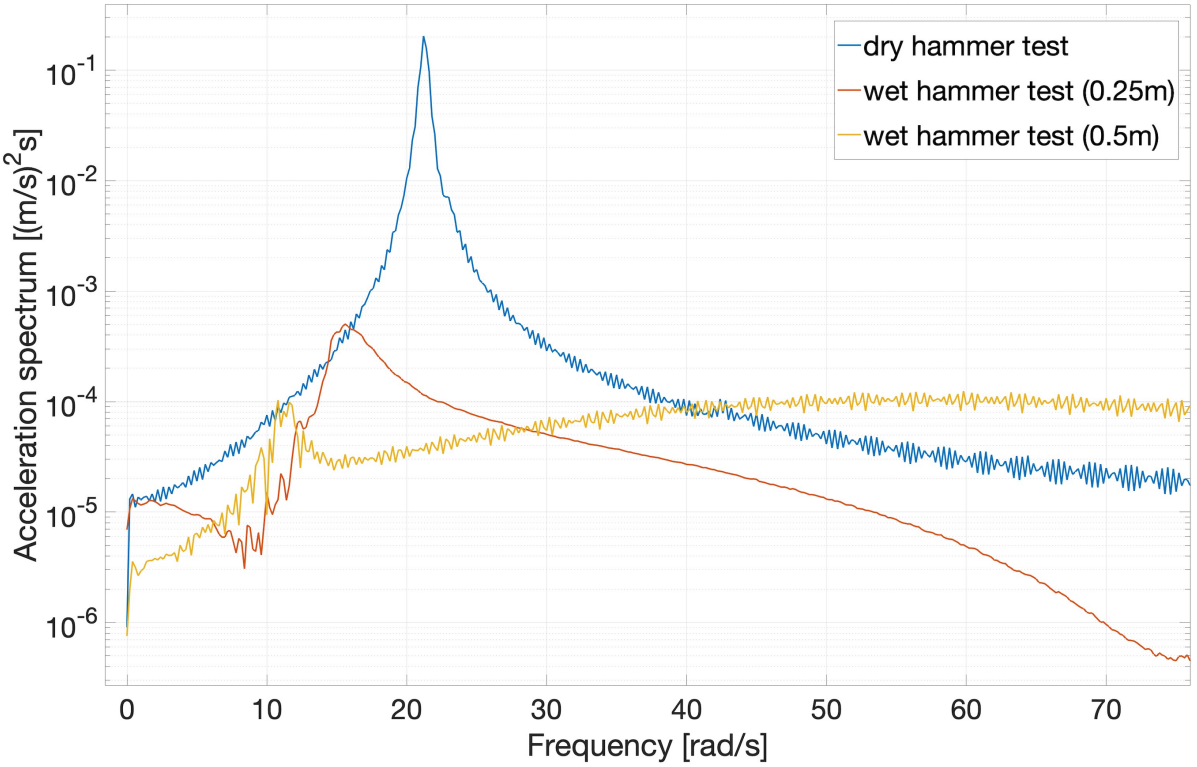


Figure 4.7: Frequency-domain beam-response spectra for the three hammer tests.

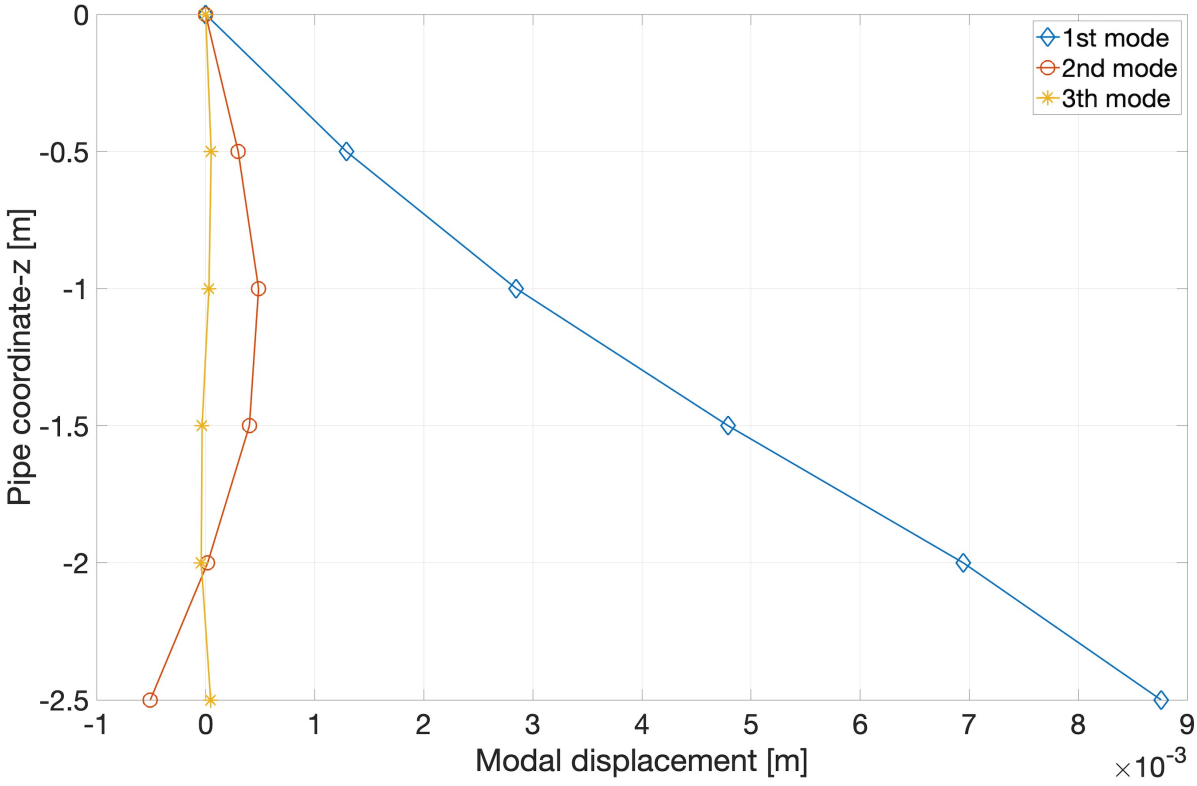


Figure 4.8: Profiles of first three beam modes, integrated from sensor accelerations measured in dry hammer tests.

submerged in the water due to an increase in added mass and damping coefficient. Measurement errors are propagated into subsequent calculations by using the mean, of the implied extreme values, on which a symmetric error range is centred.

Table 4.4: Natural frequency and time period of the beam's first mode, from accelerometer data in hammer tests.

	Period(T_p)	Natural Frequency (f)	Natural Frequency (ω)
	[s]	[s^{-1}]	[rad/s]
Dry hammer test	0.28 ± 0.002	3.6 ∓ 0.03	22.62 ∓ 0.19
Wet hammer test (0.25m)	0.43 ± 0.037	2.34 ∓ 0.2	14.70 ∓ 1.26
Wet hammer test (0.5m)	0.58 ± 0.028	1.72 ∓ 0.09	10.81 ∓ 0.53

Since the time domain response (accelerations) of the beam has been obtained from the hammer tests, the logarithmic decrement method [60] can be used to calculate the damping ratios of the beam in dry and partially submerged cases. The given time domain signals include a range of frequencies. Therefore, to apply the logarithmic decrement method, we first convert the time domain signal into the frequency domain and then apply the band-pass filters around each identified dominant frequency to isolate each component. See the decomposition of the time domain accelerations signal, obtained from the dry hammer test, based on natural frequencies in Fig. 4.9.

The damping ratios (Υ) of the beam's first three fundamental frequencies in dry and partially submerged cases are given in Table 4.5.

Table 4.5: Damping ratios Υ_1, Υ_2, and Υ_3 corresponding to the beam's first (f_1), second (f_2), and third (f_3) natural frequencies are given, respectively.

	Υ_1	Υ_2	Υ_3
	[s^{-1}]	[s^{-1}]	[s^{-1}]
Dry hammer test	0.0131	0.0057	0.0014
Wet hammer test (0.25m)	0.0470	0.0004	0.0006
Wet hammer test (0.5m)	0.0614	0.0037	0.0029

The elastic modulus of the beam is measured experimentally by performing the bending test with the beam while its fixed end is clamped. The bending test consists of applying a gradually increasing known force $F_i = g \times m_i$ at a point $L_p = 2.0$m from the clamped end of the beam and then measuring the beam's increasing deflection as 1kg masses are sequentially stacked on top of each other on a string attached to the beam's free end. The schematic of the bending test is shown in Fig. 4.10.

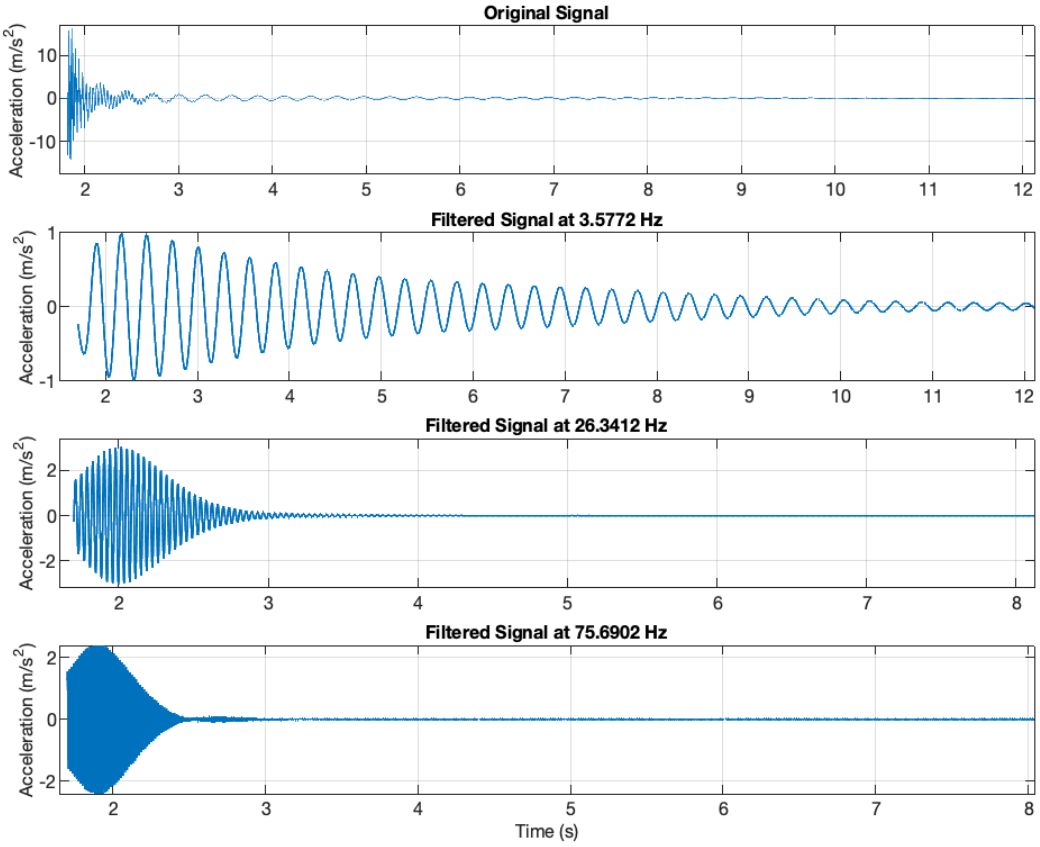


Figure 4.9: Natural frequency-based decomposition of the beam's accelerations obtained from the dry hammer tests are plotted.

Each distance D_i in Table 4.6 is measured from the bottom of the plate at which mass m_i is placed. These masses are in the form of circular iron disks that are stacked on top of each other as described above. The deflection or static offset of the beam is the difference between the two consecutive values of measured distances, i.e.

$$\zeta_i = D_{i-1} - D_i; \quad i = 0, 1, \ldots, 11. \tag{4.6}$$

The measured distances and deflections corresponding to the applied point loads are listed in Table 4.6.

The flexural rigidity EI of the beam is given by

$$EI = \frac{F_{max} L_p^3}{3\delta_{max}}, \tag{4.7}$$

where F_{max} is the maximum applied point force and δ_{max} is the corresponding maximum deflection or static offset. Using the experimentally determined natural frequencies (and hence

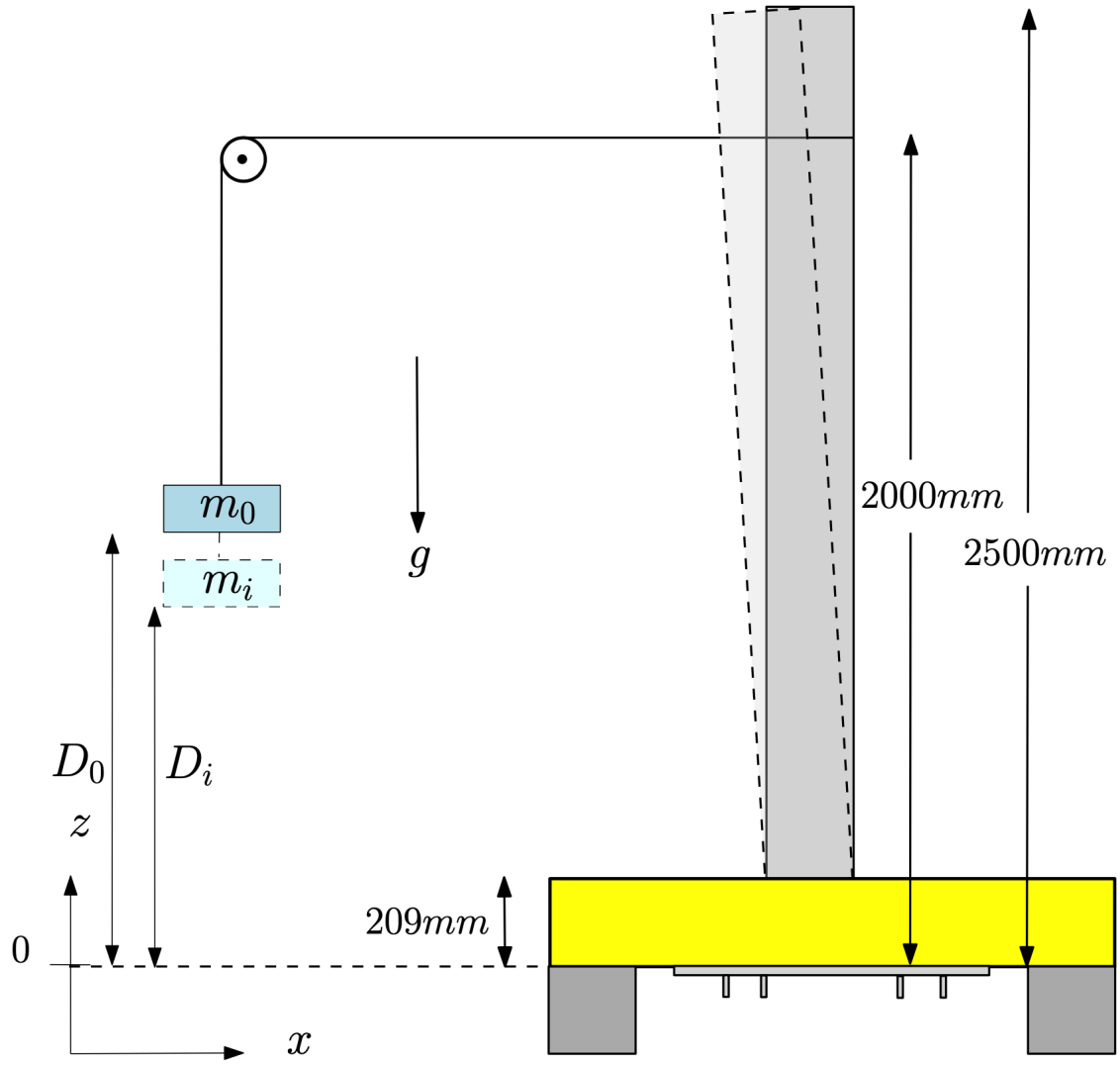


Figure 4.10: Schematic diagram of bending test. The beam before deflection is shown as a dark-grey rectangle. The beam deflected by loading of mass m_i appears as a the light-grey curvilinear quadrilateral. Movement of the base is precluded by clamping the base plate with rigid wooden blocks (shown in yellow) in such a way that the beam can move freely in the x-direction.

periods) given in Table 4.4 , and the elastic modulus E of the material computed using (4.7) , the spring constant k of the torsional spring shown in Fig. 4.2 can now be calculated, by using the procedure formulated by Blevins [9], as follows. The expression for the natural frequency of the pinned free beam with a torsion spring at the pinned joint is given in [9] as

$$f_i = \frac{\lambda_i^2}{2\pi L^2}\left(\frac{EI}{M}\right)^{1/2}; \quad i = 1, 2, 3, \ldots, \tag{4.8}$$

where f_i is the fundamental frequency of the i^{th} mode (computed via hammer tests), M is the mass per unit length (computed via the mass-distribution information in Table 4.2), L is

Table 4.6: Dependence of deflection ζ_i and maximum static offset $\delta_{i,max}$ of beam on increasing mass-loading m_i.

i	Mass	Force	Distance	Deflection	Maximum deflection
	m_i	F_i	D_i	ζ_i	$\delta_{i,max}$
	[kg]	[N]	[mm]	[mm]	[mm]
0	0	0	519	0	0
1	1	9.81	512	7	7
2	2	19.62	506	6	13
3	3	29.43	500	6	19
4	4	39.24	495	5	24
5	5	49.05	488	7	31
6	6	58.86	481	7	38
7	7	68.67	476	5	43
8	8	78.48	470	6	49
9	9	88.29	464	6	55
10	10	98.1	458	6	61
11	11	107.91	452	6	67

the length (measured), EI is flexural rigidity (computed via a bending test), and λ_i is a non-dimensional frequency parameter which is a function of $kL/(EI)$ and obtained from Table 4.7, which displays the data given in Blevins [9].

Table 4.7: Natural frequencies of a pinned free beam with a torsion spring at a pinned joint. λ_i is a function of $kL/(EI)$. Table reproduced from [9], in which data are provided to 4 significant figures.

kL/EI	$\lambda_i(kL/EI)$ $i=1$
0	0
0.01	0.4159
0.1	0.7357
1	1.248
10	1.723
100	1.857
∞	1.875

For the given material parameters, the stiffness k of the moving base, represented by the torsional spring in Fig. 4.2, is derived as follows; first, λ is calculated by rearranging (4.8)

$$\lambda = \sqrt{\frac{f2\pi L^2 M^{1/2}}{(EI)^{1/2}}} = 1.65 \pm 0.01, \tag{4.9}$$

which value of λ is then used to find the corresponding value of kL/EI from Table 4.7 via linear interpolation when the ratio kL/EI is converted onto a logarithmic scale, as shown in Fig. 4.11.

This yields

$$\log\left(\frac{kL}{EI}\right) = 1.96 \pm 0.03 \tag{4.10}$$

from which the logarithmically interpolated kL/EI is 7.13 ± 0.24. Finally, the stiffness or torsional spring constant k is then computed as

$$\begin{aligned} k &= (7.13 \pm 0.24)\frac{EI}{L} \\ &= (12.24 \pm 0.41) \times 10^3 \text{Nm/rad}. \end{aligned} \tag{4.11}$$

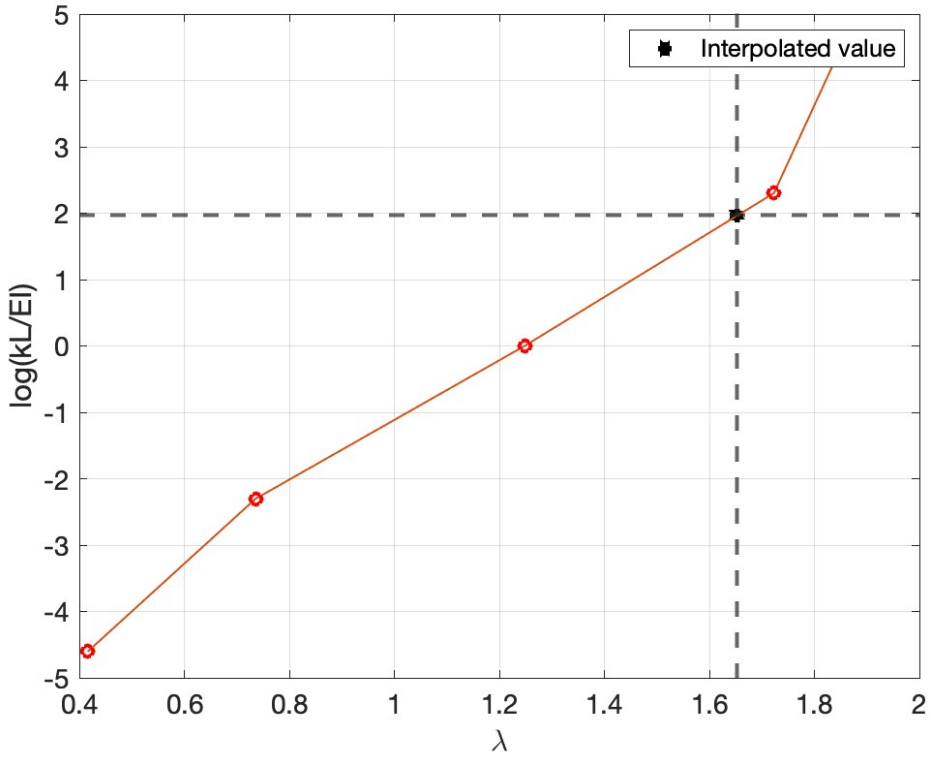


Figure 4.11: Semilog plot of data in Table 4.7 on which linear interpolation of kL/EI is performed, as described after (4.9) in the text.

Finally, all material parameters of the beam are summarised in Table 4.8.

Table 4.8: Material parameters of the beam used for the FSI experiments. Error tolerances are not available for all parameters.

Parameters	Units	Values
Spring Stiffness (k)	[Nm/rad]	$(12.24 \pm 0.41) \times 10^3$
Elastic Modulus (E)	[N/m^2]	2.378×10^9
Mass per length (M)	[kg/m]	1.6048
Length (L)	[mm]	2500
Density	[kg/m^3]	1668

In addition to material parameters, there are flow-based non-dimensional parameters significant in determining the flow-induced loading on the beam [48, 16], e.g. Froude number (Fr), Reynolds number (Re), and Keulegan-Carpenter number (KC)[46, 24]. The Froude number (Fr) is the ratio of inertial forces (flow velocity) to gravity forces (wave propagation speed or gravity waves). It is crucial for understanding the wave-induced forces on the beam and given as

$$Fr = c/\sqrt{gd_o}, \tag{4.12}$$

where the phase velocity of the wave, i.e. $c = \Lambda/T$, g is gravitational acceleration, and d_o is the outer diameter of the beam. Reynolds number is the ratio of inertial forces (flow velocity around the beam's outer diameter) to viscous forces (dynamic water viscosity) in a fluid flow. The Reynolds number characterizes the flow regime (laminar or turbulent) around the beam.

$$Re = \rho U d_o/\nu, \tag{4.13}$$

where the peak oscillatory flow velocity $U = \pi H/T$, ρ is water density, and $\nu = 1 \times 10^{-6}\mathrm{m}^2/\mathrm{s}$ is dynamic water viscosity. Keulegan-Carpenter number (KC) [46, 24] is given as the following ratio

$$KC = \pi H/d_o. \tag{4.14}$$

When KC>10, flow separation occurs, viscous effects and drag forces dominate, and diffraction and radiation forces are negligible. On the contrary, when KC<2, the flow remains attached to the body, diffraction and radiation forces are significant, and the effects of drag forces are important at resonance only.

4.4 Case-1 experiments: interactions of regular waves with the flexible beam when the carriage is at rest

Fig. 4.12 depicts the set-up for Case 1, which is further divided into two subcases corresponding to different submerged beam lengths. Subcases 1 and 2 respectively have 0.25m and 0.5m of the beam submerged, and the wave parameters for each subcase are shown in Tables 4.9 and 4.10 respectively, in which H denotes the wave height, T the wave period and Λ the wavelength; the last column gives the (dimensionless) wave steepness, defined as H/Λ. Waves of steepnesses

0.08, 0.04 and 0.03 are generated to interact with the flexible beam. We remark that the wave parameters in Tables 4.9 and 4.10 are those relating to experimental input; parameters gleaned from the actual waves generated in the wavetank were observed to differ from the input ones by up to 5%, as discussed in more detail in section 4.7 below.

Figure 4.12: Interactions of regular waves with the beam.

Table 4.9: Input parameters and characteristics of regular waves when the carriage is at rest and 0.25m of the beam is submerged in water.

H	T	Λ	Steepness (H/Λ)
[m]	[s]	[m]	[-]
0.126	1	1.56	0.081
0.282	1.5	3.51	0.080
0.016	0.5	0.39	0.041
0.062	1	1.56	0.040
0.14	1.5	3.51	0.040
0.25	2	6.239	0.040
0.39	2.5	9.748	0.040
0.016	0.58	0.525	0.031

Case 1 aims to validate the linear FSI solvers in the non-resonant regime, as the natural frequencies of the beam are higher than those of the wave. The time domain plots of the incident wave signal and the beam's response (accelerations) recorded by all accelerometers are shown in Fig. 4.13. It can be noticed that the beam's response is highest at the free end and lowest at the

Table 4.10: Input parameters and characteristics of regular waves when the carriage is at rest and 0.5m of the beam is submerged in water.

H	T	Λ	Steepness (H/Λ)
[m]	[s]	[m]	[-]
0.032	0.5	0.39	0.082
0.126	1	1.56	0.081
0.282	1.5	3.51	0.080
0.016	0.5	0.39	0.041
0.062	1	1.56	0.040
0.14	1.5	3.51	0.040
0.25	2	6.24	0.040
0.016	0.58	0.52	0.030

fixed end. Similarly, plots were checked for accelerations in X, Y, and Z directions and a greater response was found in $X-$coordinate which is also the direction of incoming waves. Therefore, the beam's free-end response in the $x-$direction will be considered for further analysis.

In Fig. 4.13, it can be noticed that the wave and acceleration signals show an irregular pattern after 200 seconds due to the interactions of the incoming waves with the waves reflected from the other end of the wavetank. For signal analysis, we will consider the part of the signals unaffected by the reflected waves.

Fig. 4.15 and Fig. 4.14 show a part of the time domain signals in the form of two subplots: (upper) the undisturbed incident water waves in solid line and refracted water waves from the beam in dashed line; (lower) acceleration, in the x-direction, of the submerged end of the beam.

Some tests with high waves were also performed that excited the beam's natural frequency due to nonlinear (sum-frequency) effects, as shown in Fig. 4.16 and Fig. 4.17, whose two subplots show: (upper) the undisturbed incident water waves in solid line and refracted water waves from the beam in dashed line; (lower) acceleration, in the x-direction, of the submerged end of the beam.

4.5 Case-2 experiments: interactions of regular water waves with the flexible beam when the carriage is moving at a constant speed

Case-2 experiments are divided into two subcases, distinguished as in Case 1: wave parameters for the first and second subcases are now shown in Tables 4.11 and 4.12 respectively.

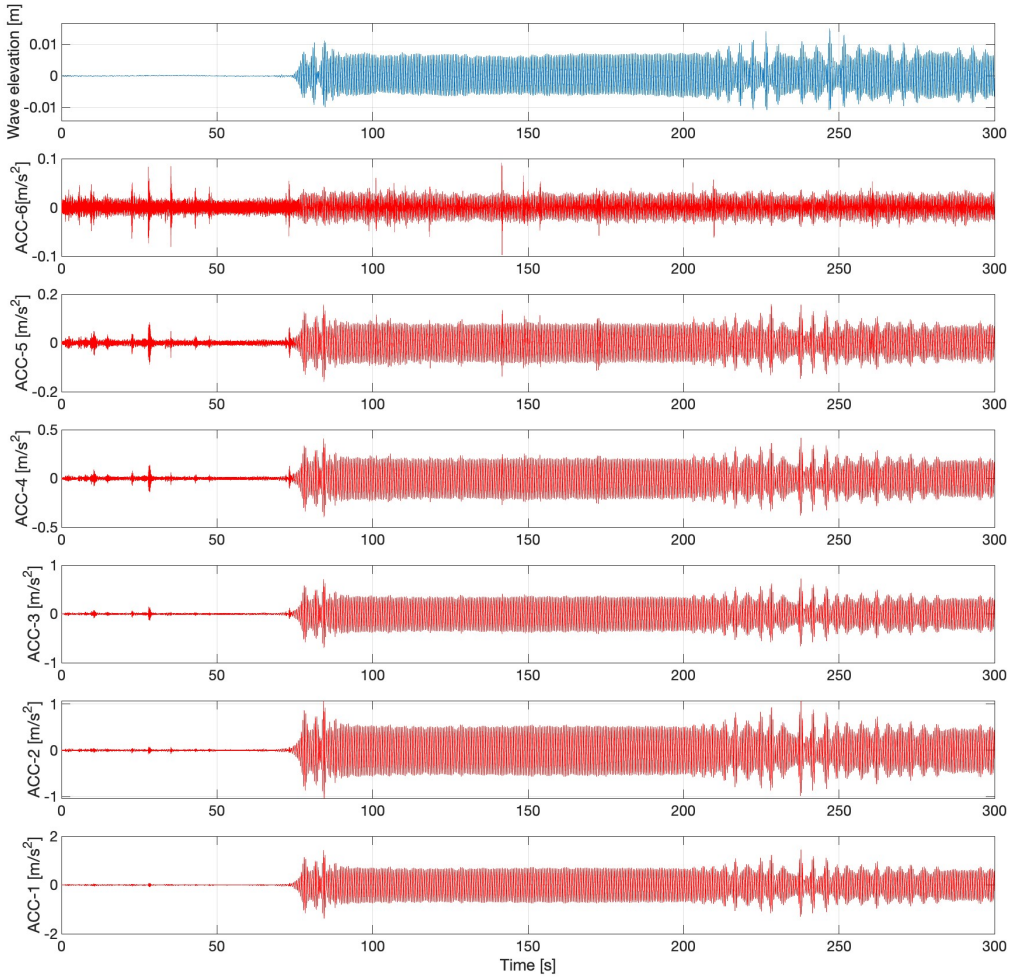


Figure 4.13: The time domain plots of the incident wave signal and the beam's response (accelerations) recorded by all accelerometers are shown. These plots are corresponding to the third test case in Table 4.9, i.e. wave height H =0.016 m and T = 0.5 s.

Table 4.11: Input parameters and characteristics of regular waves when the carriage is moving at a constant speed and 0.25m of the beam is submerged in water.

H	T	Λ	Steepness (H/Λ)	u_0	ω_e
[m]	[s]	[m]	[-]	[m/s]	[rad/s]
0.126	1	1.560	0.081	0.297	7.480
0.016	0.5	0.390	0.041	0.149	14.967
0.062	1	1.560	0.040	0.297	7.480
0.14	1.5	3.509	0.040	0.446	4.987

Moving the carriage changes the frequency with which waves encounter the beam, so that the dynamic response of the beam and its interaction with water waves, particularly at the onset of resonance, can be studied. Changing the steepness of regular waves allows both linear and

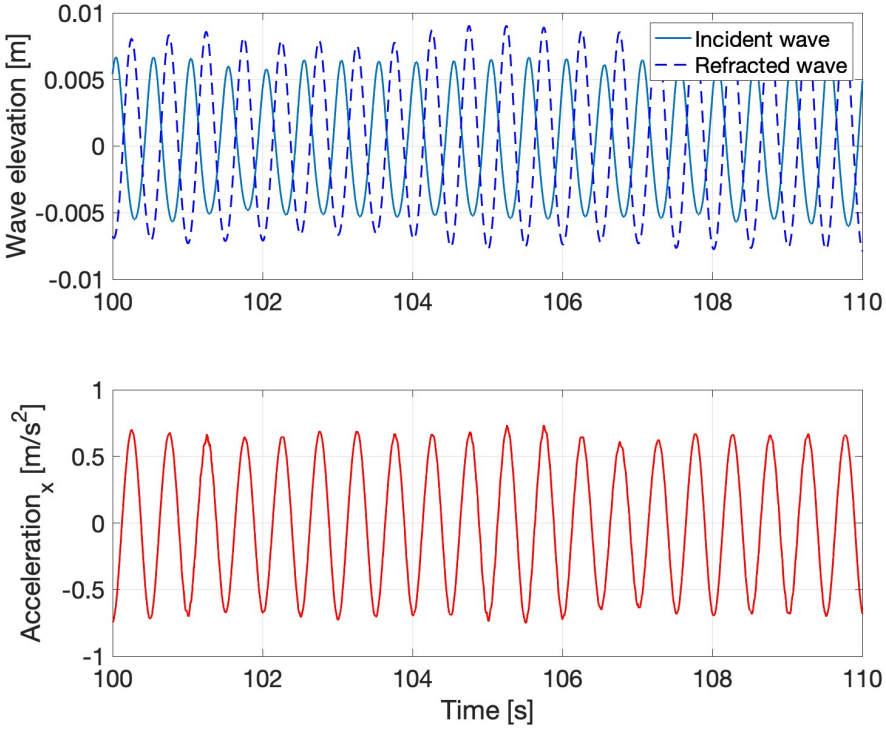


Figure 4.14: Single-frequency response of the flexible beam to regular water waves. These plots are corresponding to the third test case in Table 4.9, i.e. wave height H =0.016 m and T = 0.5 s.

Table 4.12: Input parameters and characteristics of regular waves when the carriage is moving at a constant speed and 0.5m of the beam is submerged in water.

H	T	Λ	Steepness (H/Λ)	u_0	ω_e
[m]	[s]	[m]	[-]	[m/s]	[rad/s]
0.126	1	1.56	0.081	-0.215	5.417
0.016	0.5	0.39	0.041	-0.1077	10.831
0.062	1	1.56	0.040	-0.2154	5.415
0.14	1.5	3.51	0.040	0.6864	5.418

nonlinear FSI solvers to be validated. The encounter frequency ω_e of the waves is calculated as

$$\omega_e = \omega_0 \pm u_0 \frac{\omega_0^2}{g}, \tag{4.15}$$

where $\omega_0 = 2\pi/T$ is the earth-bound frequency of the waves, u_0 is the velocity of the carriage (designated as positive/negative when the carriage moves against/with the waves) and g is the gravitational acceleration. Tests are conducted for cases where the carriage moves both with and against the waves. The speed was selected such that the natural frequency was an integer multiple (1,2 or 3) of the encounter frequency. The speed was limited to 0.7m/s because higher speeds introduce loads that would have damaged the experimental set-up. The complete time-

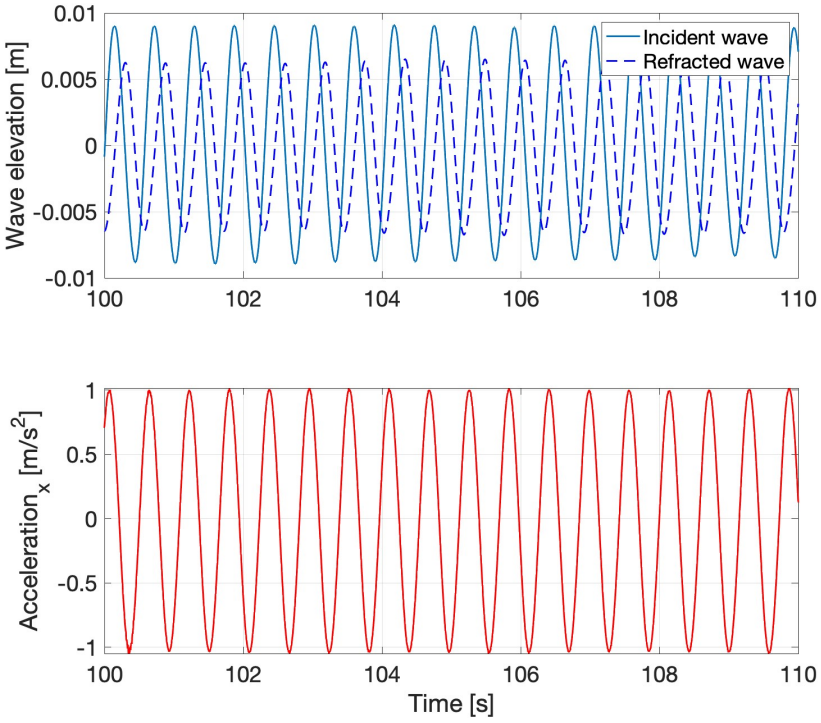


Figure 4.15: Single-frequency response of the flexible beam to regular water waves. These plots are corresponding to the last test case in Table 4.10 when wave height H =0.016 m and T = 0.58 s.

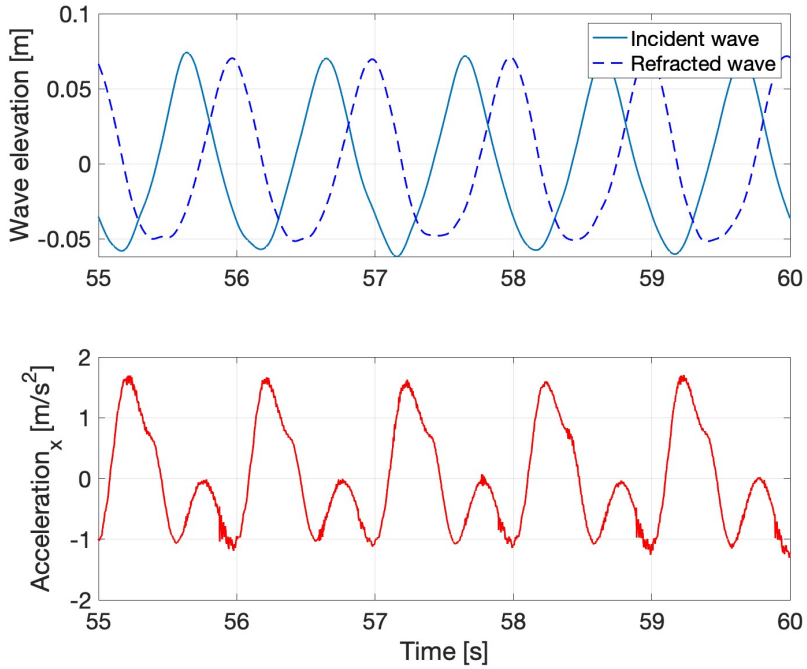


Figure 4.16: Multi-frequency response of the flexible beam to regular water waves. These plots are corresponding to the second test case in Table 4.10 when wave height H =0.126 m and T = 1 s.

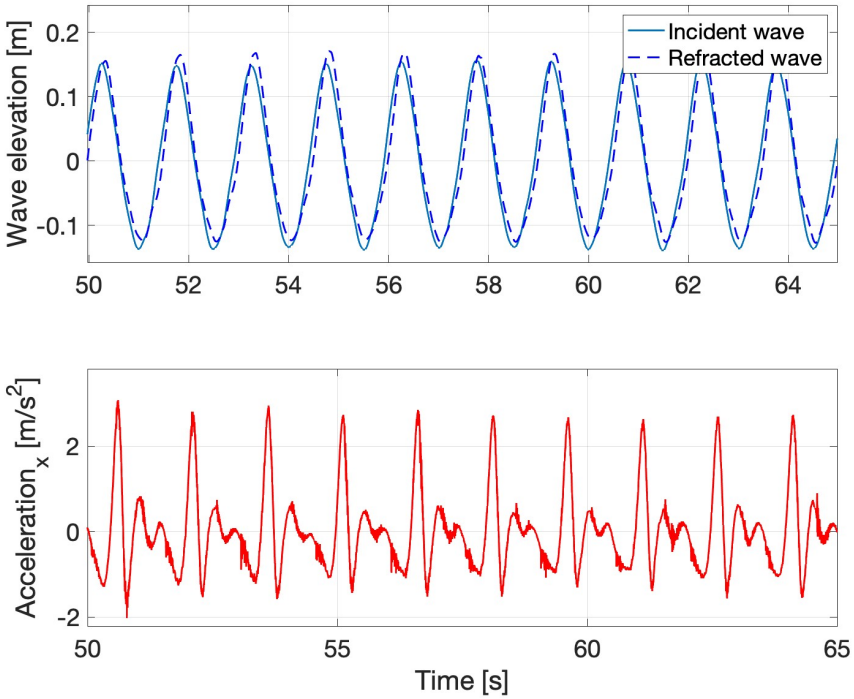


Figure 4.17: Multi-frequency of the flexible beam to regular water waves.These plots are corresponding to the third test case in Table 4.10 when wave height H =0.282 m and T = 1.5 s.

domain signal of the flexible beam's response to regular waves with 0.5m of the beam submerged is shown in Fig. 4.18 while Fig. 4.19 shows a zoomed-in section of the complete time domain signals.

4.6 Case-3 experiments: interactions of irregular water waves with the flexible beam when the carriage is at rest

Case-3 experiments are divided into two subcases, distinguished as in Cases 1 and 2: wave parameters for the first and second subcases are now shown in Tables 4.13 and 4.14 respectively.

Case 3, whose experimental set-up is shown in Fig. 4.20, is the most complex of the cases considered and is designed to yield data on structural dynamics due to nonlinear wave-loading processes related to steep and breaking waves. Irregular waves are modelled in the experimental facilities by using already-developed wave-spectrum models, which were developed to replicate oceanographic waves and are given in the form of parameterised functions. There are different models to represent waves in different regions of the world and conditions, i.e. deep seas [38],

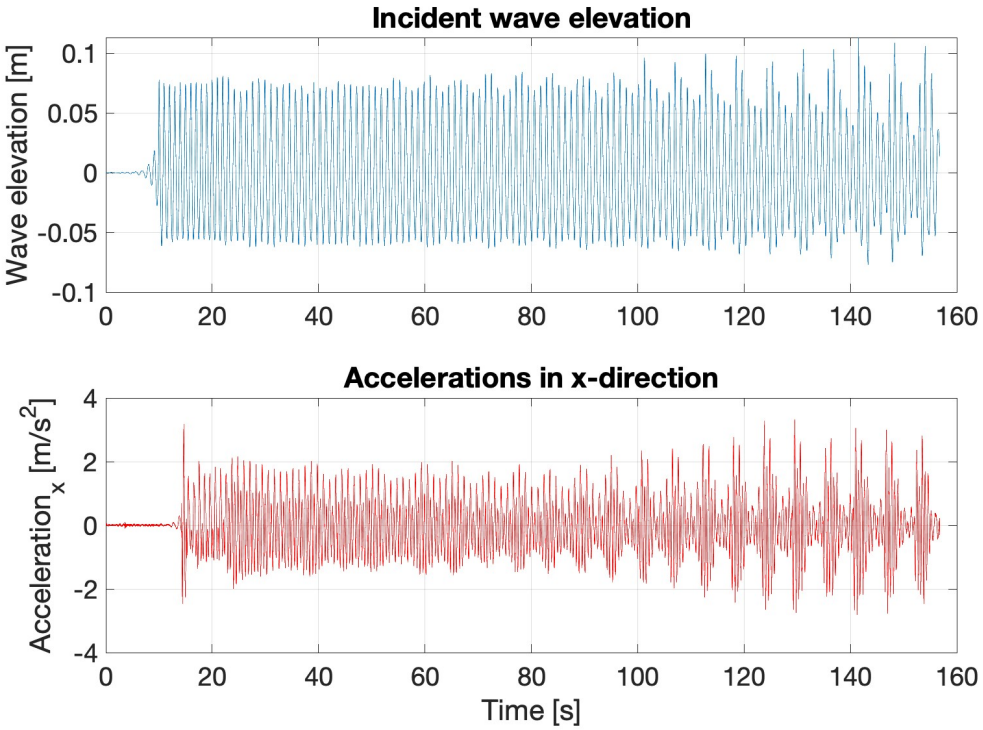


Figure 4.18: Response of the flexible beam (0.5m submerged) to regular water waves when the carriage is moving at a constant speed. The wave height and time period are 0.126 m and 1.56 s.

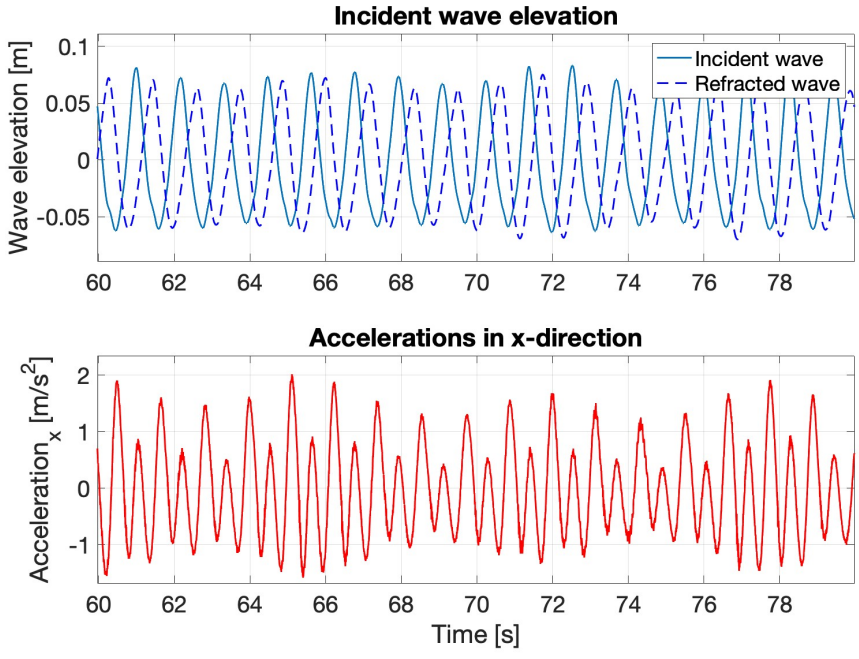


Figure 4.19: A zoomed-in part of the time domain signals of beam's response (0.5m submerged) to regular water waves when the carriage is moving at a constant speed. The wave height and time period are 0.126 m and 1.56 s.

Figure 4.20: Interactions of irregular waves with the beam.

shallow water[40], and fully developed seas [64]. In this study, we have experimentally modelled the JONSWAP (Joint North Sea Wave Observation Project) spectrum [38], which represents irregular wave patterns in the North Sea. The parametric equation for the JONSWAP spectrum is given as follows:

$$S(f) = \frac{\alpha g^2}{(2\pi)^4 f^5} \exp\Big(-\frac{5}{4}\Big(\frac{f}{f_m}\Big)\Big) \gamma^{\exp\Big(\frac{-(f-f_m)^2}{2\sigma^2 f_m^2}\Big)}, \tag{4.16}$$

$$\sigma = \begin{cases} \sigma_a = 0.07 & \text{for} \quad f \leq f_m, \\ \sigma_b = 0.09 & \text{for} \quad f > f_m, \end{cases}$$

where f_m is the maximum frequency of the spectrum; g is gravitational acceleration; α is a coefficient, known as the Philips parameter, that scales the overall magnitude of the spectrum and is taken as 0.0081; γ is the peak-enhancement factor whose value is region dependent[53], e.g. 3.43 to 3.70 for the Jiangsu waters in China[28]. This case aims to validate high-fidelity

FSI solvers.

Table 4.13: Input parameters and characteristics of irregular waves when the carriage is at rest and 0.25m of the beam is submerged in water.

MARIN Test No. 70065_02CB_02	Environment	Time	Irregular-Sea Characteristics			
			JONSWAP Type Spectrum			
			H_s	T_p	Dir.	γ
		[s]	[m]	[s]	[deg]	[-]
North Sea state						
011_001_01	Gain 1.0	1781	0.34	2.25	180	2.9
011_001_01	Gain 0.25	1781	0.085	2.25	180	2.9
011_001_01	Gain 0.5	1781	0.17	2.25	180	2.9

Table 4.14: Input parameters and characteristics of irregular waves when the carriage is at rest and 0.5m of the beam is submerged in water.

MARIN Test No. 70065_02CB_02	Environment	Time	Irregular-Sea Characteristics			
			JONSWAP Type Spectrum			
			H_s	T_p	Dir.	γ
		[s]	[m]	[s]	[deg]	[-]
North Sea state						
011_001_01	Gain 1.0	1781	0.34	2.25	180	2.9
011_001_01	Gain 0.5	1781	0.17	2.25	180	2.9

In Tables 4.13 and 4.14, the Environment parameter Gain 1.0 represents the actual wave spectrum of the North Sea state, whereas Gain 0.25 generates scaled waves up to a quarter of the actual wave height and Gain 0.5 generates waves scaled up to half the actual wave height. H_s is the significant wave height and T_p is the wave period. We report one interesting event that occurred when a steep breaking wave interacted with the beam, shown in Fig. 4.21.

The response of the beam to the breaking wave is recorded and plotted in terms of the time- and frequency-varying data which is shown in Figs. 4.22 to 4.24.

The nonlinear dynamic beam response clearly shows multiple modes, which are further investigated by performing Fourier transform in the frequency domain, as shown in Fig. 4.23.

Furthermore, frequency analysis is also performed in which the time-domain response is first filtered (using proprietary Matlab software from MARIN) and then decomposed into higher and lower time-domain response-frequency components. The actual and filtered time-domain responses are compared in Fig. 4.24, which reveals that the impact wave excited multiple natural frequencies in the beam. The filter frequency is 25 rad/s.. The nonlinear response of the beam is due to the excitation of higher frequencies: in particular, it can be seen that the

Figure 4.21: Impact of steep irregular waves with the beam.

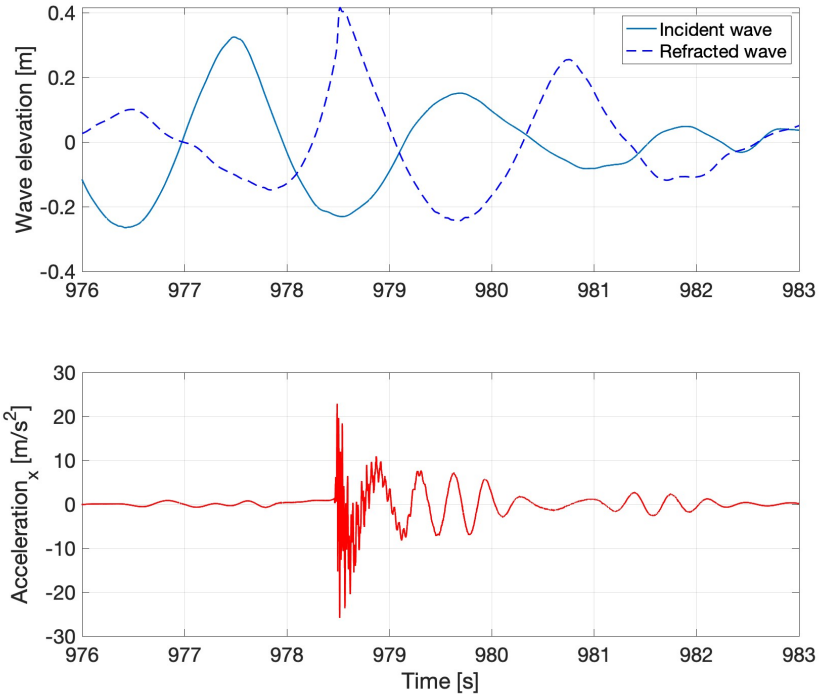


Figure 4.22: Response of the flexible beam to irregular waves.

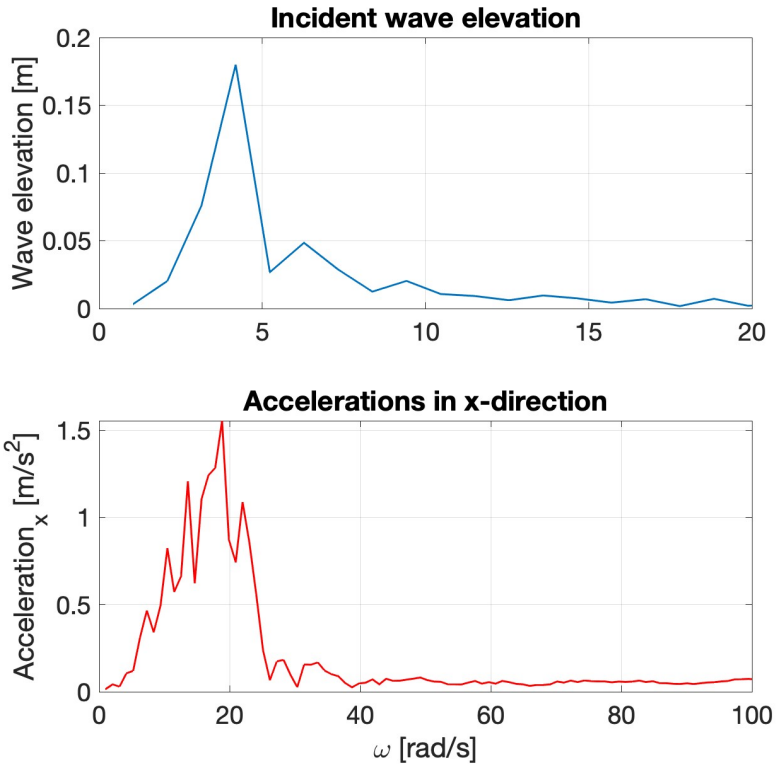


Figure 4.23: The time-domain experimental signals are plotted in the frequency domain. The top plot shows the wave signal while the bottom plot depicts the beam response.

high-frequency response (yellow) decays faster than the low-frequency response (red) as a result of structural and hydrodynamic damping.

4.7 Experimental uncertainty

To assess the accuracy and reliability of the experimental data recorded, it is essential to quantify the error at each stage of the experiments. Therefore, this section explains the different types of errors that may affect the measurements, and the precautionary steps taken to minimise them. The experimental campaign can be divided into three stages: designing and fabricating the experimental set-up; performing the experiments and recording the measurements; and, processing the recorded data. Each stage incurs associated specific errors. In the first stage, errors may accrue through defects in the design and manufacturing. Cooke *et al.*[20] presents three case studies that are useful to understand error occurrence during the design stage. Therefore, to minimise this error the set-up was designed and fabricated by the team of researchers and technicians at MARIN and the material was procured from certified providers. The sensors were tested and calibrated, and the set-up was inspected before deploying in the wavetank. During

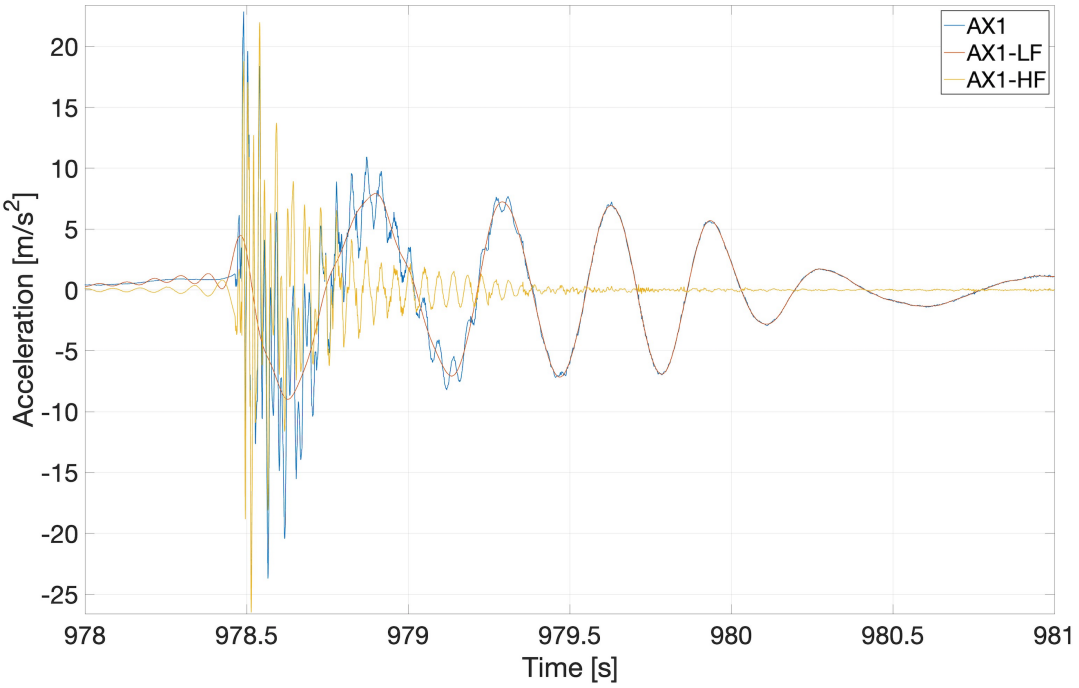


Figure 4.24: Frequency analysis of the response of the flexible beam to irregular waves. The original signal (blue) is decomposed into higher (yellow) and lower (red) frequency responses. In the legend, AX1 represents the original signal, while AX1-HF and AX1-LF are the respective high-frequency and low-frequency parts of the original signal.

this stage, we found that one of the accelerometers (ACC-2) was defective and was hence replaced by the team. ACC-2 was placed on the external surface of the beam which interacted with the water in the event of high-amplitude waves and hence a source of uncertainty. The setup could be improved by placing the accelerometer (ACC-2) at the internal surface of the beam. Also note that the accelerometers rotate with the beam as they are fixed to the beam. Therefore, the measurement of the acceleration component normal to the beam, i.e. along the x-direction, contains a gravitational term $g\sin(\phi)$, where ϕ is the angle between the local rotation of the beam with the original vertical position of the beam. However, the contribution of the gravitational term is negligible as compared to the accelerations due to beam vibrations. We performed several hammer tests during the experiments to ensure that the structure's resonance period after the experiments was the same as the initial resonance period before applying the loading. The purpose was to ensure that the beam was not damaged by the water-wave interactions.

The second stage is when actual experiments are performed during the generation of water waves in the wavetank. Before recording actual experimental measurements, the six accelerometers and two probes (see Fig. 4.2 and section 4.2) were calibrated. Experimental wave-generation

is an iterative process and, from previous experiments, the experts observed that the actual generated-wave parameters in the wavetank can differ by 1% to 5% from the input-wave parameters. These variations occur due to basin effects, i.e., wave reflections. To quantify the discrepancy, we have compared the input-wave parameters and actual waves measured by the probe for the first subcase of experimental case 1. The percentage relative error between the input parameters, i.e. wave amplitude, time period and wavelength and those of the actual wave measured in the basin are listed in Table. 4.15. We note that the percentage relative error ranges from 0 to 6.98. To obtain the values of wave amplitude presented in Table. 4.15,

Table 4.15: Percentage relative error between the input wave parameters and those of the actual wave generated in the wavetank; here, for the first subcase of experimental case 1.

A	T	Λ
[m]	[s]	[m]
0.00%	-0.50%	-0.99%
2.17%	-0.86%	-1.71%
2.56%	1.01%	2.03%
-5.20%	0.05%	0.10%
1.45%	-0.17%	-0.33%
1.63%	0.03%	0.06%
0.52%	0.40%	0.80%
-6.98%	0.52%	1.04%

we performed a harmonic analysis of measurements obtained from the probe located in front of the beam. That is, an uninterrupted wave-signal window was selected from the measured wave elevation and then analysed in the frequency domain by performing a Fourier transform. Next, the signal's amplitude at the corresponding time period was recorded. To minimise this error, we experimentally simulated water waves without the beam set-up and ensured that the generated waves were within the acceptable range i.e. 5% to 6%, in keeping with the above percentage-relative error quantification.

Another type of error, arising at the second stage, is the intrinsic instrument error of the measuring equipment. The sensors involved are the two wave probes that measure the incident and reflected waves, and the six accelerometers that measure the beam's accelerations at six equidistant points. To quantify this type of error, we took measurements twice and then computed the relative difference. Table 4.16 shows the relative difference in the first fundamental frequency ($f^{(1)}$) of the beam, measured by the accelerometers, when a dry hammer test was performed twice. The fundamental frequency is computed by taking the Fourier transform of

the time-domain signal by using the MATLAB function FFT. Results shown in Table 4.16 show that the relative difference of the frequencies from the two hammer tests is less than 1%, which confirms that the instrumental errors are dominated by those accruing from wave-generation effects.

Table 4.16: Accelerometer-measurement errors.

	Test 1	Test 2	Relative error %
	$f^{(1)}$ [1/s]	$f^{(1)}$ [1/s]	
AX1	3.6	3.59	0.28%
AX2	3.6	3.59	0.28%
AX3	3.6	3.59	0.28%
AX4	3.58	3.59	-0.28%
AX5	3.56	3.59	-0.84%
AX6	3.6	3.58	0.56%

In the third stage, errors arise in the time-domain processing of experimentally measured data, for example, the signal's amplitude and frequency. The data-processing error depends upon the algorithm used to analyse the data, e.g. discrete Fourier transform. Other common examples of this type of error are truncation error, overflow error, and rounding error.

In addition to the above-mentioned errors, human error also contributes towards total error; this can be minimised by re-examining both set-up and measurements. Accordingly we ensured that specialised teams of experts performed relevant parts of the experiments, i.e. design, fabrication, bending test, hammer tests and the actual FSI experiments in the wavetank. Moreover, through our numerical model of the beam (ultilising the parameters given in Table. 4.8), we found that the relative error between the experimentally- and numerically-computed dry resonance period is 0.3%.

See Appendix C for the details of data availability and post-processing codes.

4.8 Conclusion

This experimental study tested the dynamic response of a flexible beam subjected to a wide range of simulated sea states, namely, mild-to-extreme and regular-to-irregular. The experimental data obtained herein will be useful for mathematical, engineering and computational research communities in the validation of FSI numerical solvers ranging from linear to high-fidelity. In the next two chapters, we will explain the development of the numerical models for this specific experimental setup and perform the validation of MARIN's in-house linear and

high-fidelity FSI solvers.

Chapter 5

Linear fluid-structure interaction modelling of regular water waves with the flexible beam

5.1 Introduction

A series of experiments, divided into three experimental cases, have been performed with two goals in mind [69]: to understand fluid-structure interactions (FSIs) of waves impacting on a flexible beam with simultaneous measurements of beam accelerations and incident and reflected waves; and, to use the acquired data set for validation of a hierarchy of FSI numerical models. The experimental campaign has successfully been carried out at the concept basin facility of the *Maritime Research Institute Netherlands* (MARIN) and high-quality data has been recorded and shared to enhance our understanding of the FSI process. To achieve the second goal of the study, i.e. validation of linear, nonlinear and high-fidelity numerical FSI solvers that are commonly employed by the maritime industry in the design of fixed-foundation offshore wind turbines, it is essential to make experimental data publicly accessible. Hence, an accessible online platform, specifically a GitHub[1] public repository, is created to make the data available to the public. Furthermore, we have shared the details of the experiments with the scientific community in the form of a published research article [69]. Now that the experiments are performed and shared publicly, the next step is to perform a validation study which is the

[1] https://github.com/EAGRE-water-wave-impact-modelling/FSI_Experiments

concern of this chapter.

This comparison study uses SEACAL [58] to simulate the flexible beam's response to the water-wave conditions that are tested in the first experimental case. The numerical results obtained from the SEACAL are then compared to the experimental data. The wave conditions tested in experimental case 1 are explained in the next sections.

5.2 Experimental case 1

The experimental setup includes a flexible beam which is attached to the basin's carriage at one end while the other free end is submerged in the water. The beam is equipped with six accelerometers which are equally distributed along the beam's length. These accelerometers measure the beam response. There are two probes to measure the incident and the reflected water-wave elevation. The probe in the front of the beam measures the undisturbed incident wave and the probe parallel to the beam measures the change in the wave, which is caused by the beam's motion. The setup is designed such that it admits the simultaneous measurements of incident waves and the beam's response. Hence, it is suitable for studying FSI problems [69].

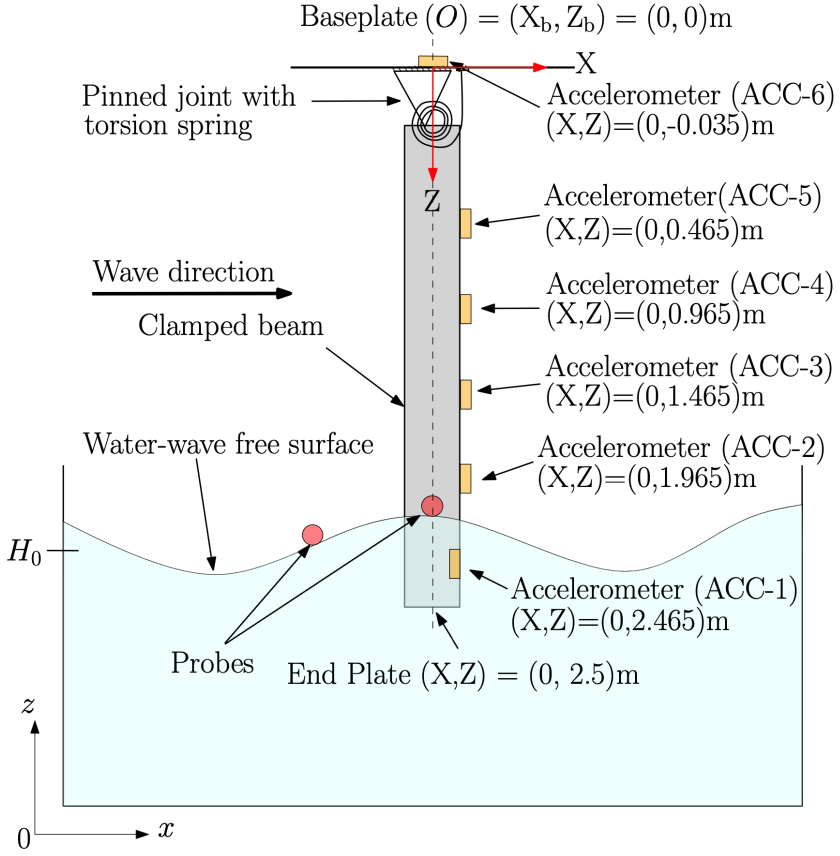


Figure 5.1: Schematic side view of the experimental set-up [69].

Experimental Case 1 concerns interactions of the regular waves with the flexible beam when the carriage is at rest. This case aims to validate the linear FSI solvers in the non-resonant regime, as the beam's natural frequencies are higher than those of the wave. However, some tests with high waves were also performed that excited the beam's natural frequency due to nonlinear (sum-frequency) effects.

This experimental case is further divided into two subcases corresponding to different submerged beam lengths. Subcases 1 and 2 respectively have 0.25m and 0.5m of the beam submerged.

5.2.1 Subcase 1: beam submerged at 0.25m

At first, the beam is submerged at 0.25m and its response towards the different wave parameters and characteristics is studied. The waves with different parameters and characteristics, which are used in this study, are listed in Table 5.1.

Table 5.1: Regular-wave parameters and characteristics when the carriage is at rest and 0.25m of the beam is submerged in water.

H	A	T	ω	Λ	Steepness (H/Λ)
[m]	[m]	[s]	[rad/s]	[m]	[-]
0.126	0.063	1	6.28	1.56	0.081
0.282	0.141	1.5	4.19	3.51	0.080
0.016	0.008	0.5	12.56	0.39	0.041
0.062	0.031	1	6.28	1.56	0.040
0.14	0.070	1.5	4.19	3.51	0.040
0.25	0.125	2	3.14	6.239	0.040
0.39	0.195	2.5	2.51	9.748	0.040
0.016	0.008	0.58	10.83	0.525	0.031

In Table 5.1, H denotes the wave height, A wave amplitude, T the wave period, ω the encounter frequency, and Λ the wavelength; the last column gives the (dimensionless) wave steepness, defined as H/Λ. Waves of steepnesses 0.08, 0.04 and 0.03 are generated to interact with the flexible beam. The plot of wave height corresponding to the encounter frequency is plotted to explain the selection criteria of different wave parameters. The plot is shown in Fig. 5.2.

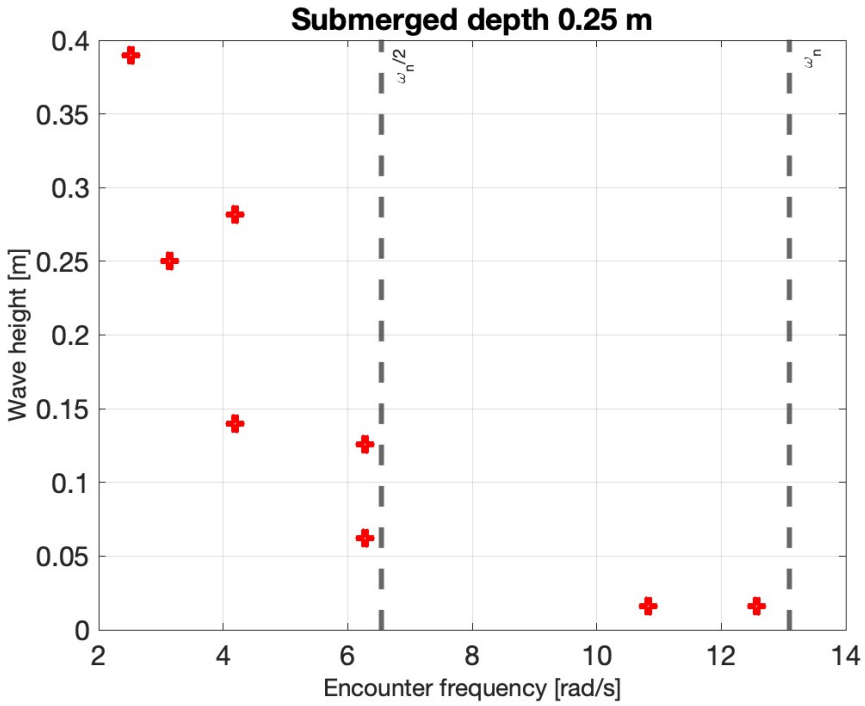


Figure 5.2: A plot of the wave heights corresponding to the wave frequencies which are used to study the beam's response when the beam's submerged depth is 0.25m.

The red markers show the wave height corresponding to the wave frequency which is also the beam's encounter frequency. There are two grey-coloured vertical dashed lines; the first dashed line is plotted at half of the first resonant frequency of the beam while the second one is plotted at the first resonant frequency of the beam. The resonant frequency of the beam, when submerged depth is 0.25m, is measured experimentally by performing the hammer tests. Some wave parameters are selected such that the resonance frequency and sum-frequency effect can be studied on the beam's response.

5.2.2 Subcase 2: beam submerged at 0.5m

After subcase 1, the beam's submerged depth was increased up to 0.5m to study how different submerged depths will affect the beam's response. This study of the different regular-wave interactions with a beam, when the beam's submerged depth is 0.5m, is explained in subcase 2. Similar to subcase 1, regular waves of different parameters and characteristics are chosen to study the beam's response in diverse wave conditions. The wave parameters and characteristics for the second subcase are listed in Table5.2.

Similar to Table 5.1, in Table 5.2, H denotes the wave height, A wave amplitude, T the wave

Table 5.2: Regular-wave parameters and characteristics when the carriage is at rest and 0.5m of the beam is submerged in water.

H	A	T	ω	Λ	Steepness (H/Λ)
[m]	[m]	[s]	$[rad/s]$	[m]	[-]
0.032	0.016	0.5	12.56	0.39	0.082
0.126	0.063	1	6.28	1.56	0.081
0.282	0.141	1.5	4.19	3.51	0.080
0.016	0.008	0.5	12.56	0.39	0.041
0.062	0.031	1	6.28	1.56	0.040
0.14	0.070	1.5	4.19	3.51	0.040
0.25	0.125	2	3.14	6.24	0.040
0.016	0.008	0.58	10.83	0.52	0.030

period, ω the encounter frequency and Λ the wavelength; the last column gives the (dimensionless) wave steepness, defined as H/Λ. Waves of steepnesses 0.08, 0.04 and 0.03 are generated to interact with the flexible beam.

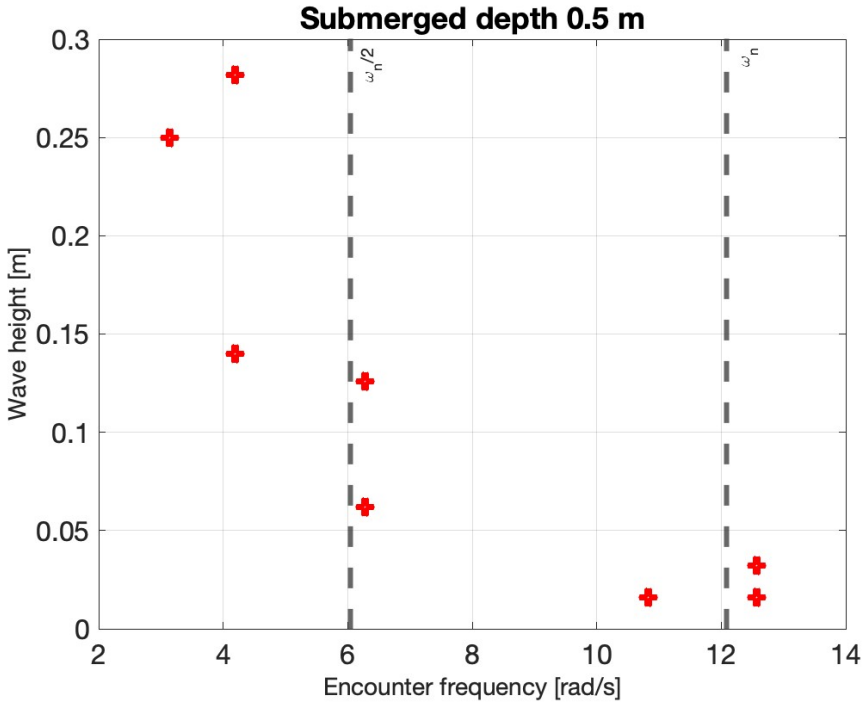


Figure 5.3: A plot of the wave heights corresponding to the wave frequencies which are used to study the beam's response when the beam's submerged depth is 0.5m.

Similar to subcase 1, the red markers in Fig. 5.3 show the wave height corresponding to the wave frequency which is also the beam's encounter frequency. There are two grey-coloured vertical dashed lines. The first line is plotted at half of the first resonant frequency of the beam while the second one shows the first resonant frequency of the beam. The resonant frequency of the beam is measured experimentally by performing the hammer tests. Some wave parameters

are selected such that the resonance frequency and sum-frequency effect can be studied on the beam's response. It can be noticed that the beam's natural frequency decreases as we submerged the beam deeper in the water. This is due to an increase in the added mass and damping introduced by the medium.

5.3 Harmonic analysis of the experimental data

The output data from the experiments are in the form of time domain signals which require further signal processing to extract essential information, i.e. magnitude and phase of the fundamental frequency, and harmonics of the fundamental frequency. This information identifies whether the structure's response is linear or nonlinear. Before analysing the structure's response, it is necessary to determine whether the structure corresponds to a linear or a nonlinear system. One way to determine this is by applying impact load on the structure, i.e. hammer test, and analysing the structure's response in the frequency domain. Therefore, we performed dry and wet hammer tests and examined the beam's response in the frequency domain. Based on the response, we concluded that the beam used in this study corresponds to a linear system [69]. However, a linear system can also exhibit a nonlinear response in certain loading conditions; therefore, it is essential to distinguish a linear response from a nonlinear response. The characteristics associated with the linear response are: a linear system obeys the principle of superposition, i.e. if a system's response to two inputs $x_1(t)$ and $x_2(t)$ are $y_1(t)$ and $y_2(t)$, respectively, then superposition holds if input $ax_1(t)+bx_2(t)$ results in a response $ay_1(t)+by_2(t)$ where a and b are constants [75]. A linear response will either show one fundamental frequency or harmonics that are integer multiples of the fundamental frequency. When harmonics appear in the response of a linear system, the magnitude of the harmonics will decrease with increasing harmonic order, and the phase relationship between the harmonics should be consistent, i.e. an integer multiple of the phase angle of the fundamental frequency. As an example of a linear response, the output signal of the incident wave recorded by the wave probe located at the front of the beam and accelerations of the submerged end of the beam in x-direction are shown in Fig. 5.4. This plot refers to the first subcase of experimental case 1 in which the beam is submerged at 0.25m, and the wave amplitude is 0.031m with a time period of $1s$.

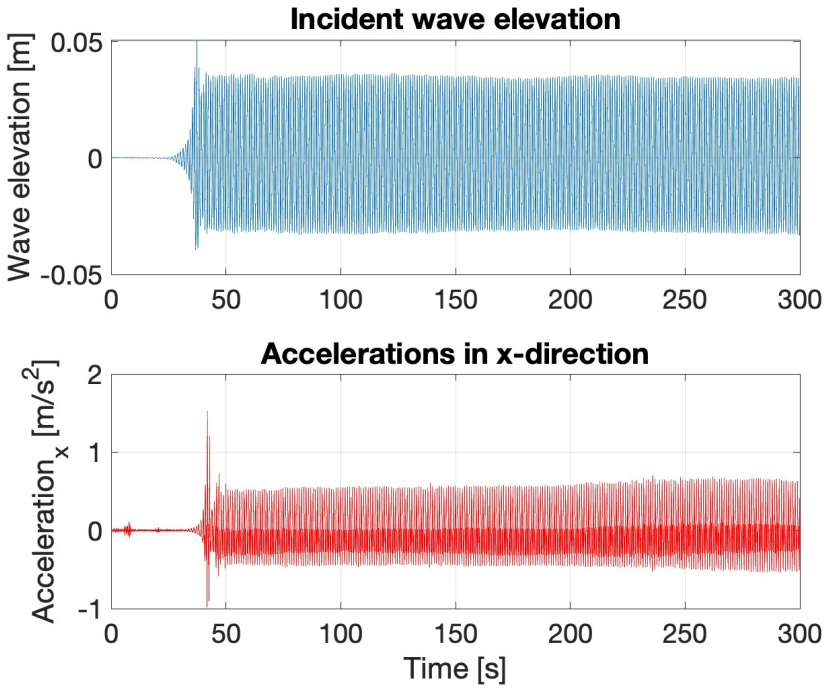


Figure 5.4: The top plot shows the incident wave signal and the bottom plot shows the response, i.e. accelerations in x direction, measured by the accelerometer located at the submerged end of the beam.

The beam's response is composed of different frequencies as the response is also affected by the waves reflected from the beach in addition to the incident waves. This study is about the validation of a linear solver, therefore, we need to extract the first fundamental frequency of the signal. For this, we select a signal window which is not affected by the reflecting waves and find the first fundamental natural frequency/mode by taking the Fourier transform. The Fourier transform of the signals shown in Fig. 5.4 from time 100 to 250 seconds is shown in Fig. 5.5.

The top plot shows the dominant frequency and amplitude of the wave signal. In this test case, a regular wave of 0.031m amplitude is generated with an angular frequency of 6.28 rad/s, therefore, the Fourier transform shows one peak of 0.03m amplitude at 6.32 rad/s. The difference in the angular frequency is due to the phase shift. Similarly, the Fourier transform of the beam's response signal shows that multiple high-order frequencies or harmonics got excited due to wave interaction. These harmonics are the integer multiples of fundamental frequency and their amplitude decreases as the frequency increases. This observation shows that the beam's response is linear to this wave condition. In this study, we will only consider the beam's response at the first natural frequency because the software that needs to be validated is a linear FSI solver. Once the amplitudes of the input (incident wave) signal and the output (beam's

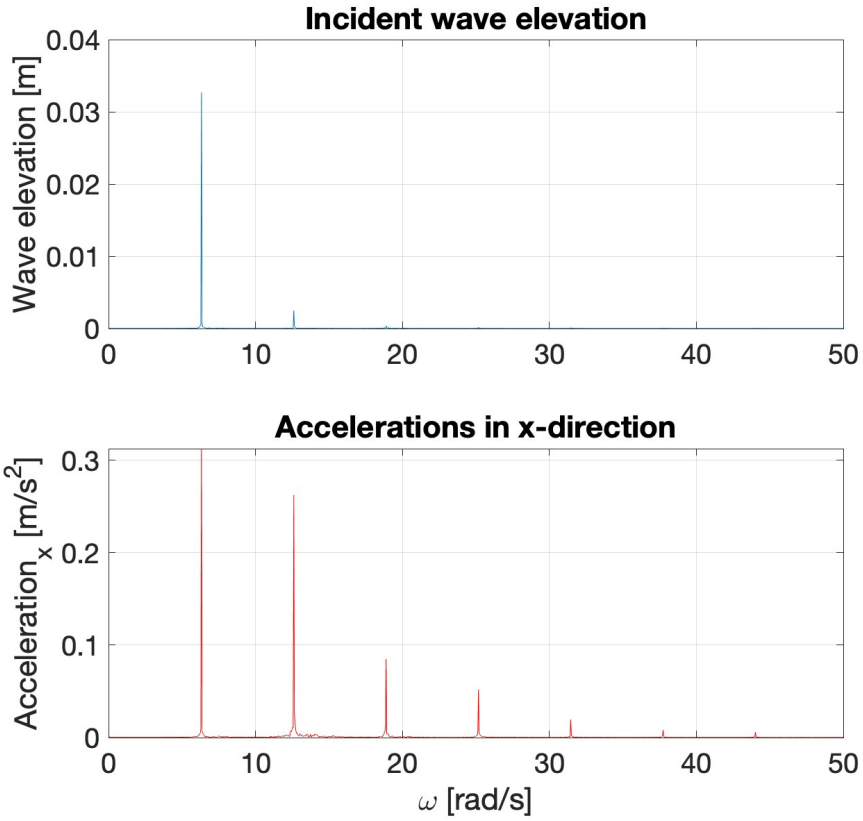


Figure 5.5: The top plot shows the amplitude and frequency of the dominant mode of the incident wave while the bottom plot shows the beam response,i.e. amplitude and frequency of the incoming wave harmonics, that got excited due to the wave interaction.

accelerations in x direction) signal are known for the first mode, the next step is to compute the response amplitude operator (RAO). RAO is defined as the ratio of the amplitude of the output to the amplitude of the input signal in the frequency domain. RAO is given as

$$RAO(f) = \frac{O(f)}{I(f)} = \frac{A_{acc}}{A_{wave}}, \tag{5.1}$$

where A_{acc} is the amplitude of the beam's acceleration in x direction and A_{wave} is the amplitude of the incident wave. The output from SEACAL is in the form of RAO. The wave amplitude is a known input and the response is calculated by using the relation given in (5.1). Finally, the RAO and beam's response obtained from the experiments and SEACAL are compared.

5.4 SEACAL: A linear FSI solver

This section is dedicated to the explanation of mathematical and numerical models used in SEACAL to carry out the FSI simulation. These explanations are the summary of the details

mentioned in SEACAL's theory manual [58]. SEACAL is a linear FSI solver which calculates the behaviour of floating structures, mainly ships, in water waves. This software is developed in the Cooperative Research Ships (CRS) SEACAL WG (2021-2023). It has three sub-programs which are:

1. HYDMES, to create the free-surface mesh around the floating structure and describe the structure's surface by using 3D panels. It also performs hydrostatic calculations and determines the flexural modes.
2. HYDCAL, to calculate the hydrodynamic coefficients by solving the linearised boundary value problem.
3. RESCAL, to calculate the response of the structure in waves and added resistance.

5.4.1 Generalized Modes

A finite element (FE) model of the structure is created in FE software which is not a part of SEACAL. The objective is to compute the structure's generalized modes. The generalized modes of the structure consist of six rigid body modes and E elastic modes. Hence, a $6\times(6+E)$ mode-shape matrix is defined at each node P of the structure's FE model. For this purpose, the structure is discretised by using finite elements with N total number of nodes. The global rigid body modes of the structure are defined at each node P of the model. The global modes consist of three rigid translations and three rigid rotations. The rotations are defined around an arbitrary point G that is taken at the structure's global centre of gravity. However, in this case, rigid-body modes are not considered because the beam is clamped.

The elastic/ dry natural modes of the structure are computed by solving the following eigenvalue problem

$$\left[\hat{M}\right]\left[\ddot{\hat{\xi}}\right]+\left[\hat{K}\right]\left[\hat{\xi}\right]=0, \tag{5.2}$$

where $[\hat{M}]$ is a $(6N)\times(6N)$ mass matrix, $[\hat{K}]$ a $(6N)\times(6N)$ structural stiffness matrix, and $\hat{\xi}$ a $6N\times1$ motions vector. The FE solver outputs the eigenvectors for a given mode i at each node P of the model as

$$[h_i]_p=[h_{i1}h_{i2}h_{i3}h_{i4}h_{i5}h_{i6}]_P^{-1}=[\hat{\xi}_i]_P, \tag{5.3}$$

and the corresponding forces at each node as

$$[F_i]_P = \begin{bmatrix} \hat{F}_i \\ \hat{M}_i \end{bmatrix}_P = \omega_i^2 \left[\left[\hat{M} \right] \left[\hat{\xi}_i \right] \right]_P. \tag{5.4}$$

The generalized modes shapes at each node are given as:

$$[h]_P = \begin{bmatrix} [h]_P^G & [h]_P^E \end{bmatrix}. \tag{5.5}$$

For demonstration, the FEM model created for the SEACAL simulations is shown in Fig. 5.6.

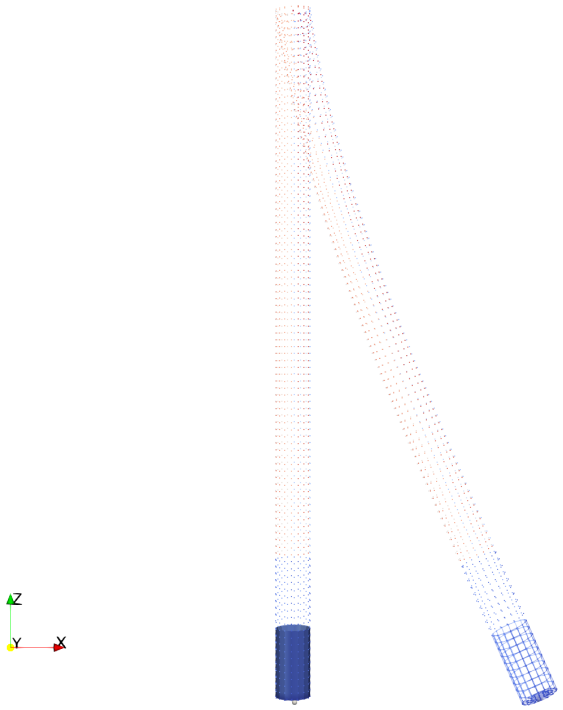


Figure 5.6: The straight beam shows the initial state of the beam and the deformed beam shows the beam shape when the first fundamental mode is excited.

The beam's model is created through a combination of structural and hydro mesh elements. Hydro mesh elements are used for the submerged part of the beam, while structural mesh elements are utilized for the part above water. The submerged part of the beam is distinguished by the dark blue bottom in Fig 5.6 while the part at the top of the dark blue part is in the air. The elastic modes that were calculated by the FE solver are then transferred to the SEACAL tool in terms of nodal points and wetted elements. The transferred variables are then used as generalized mode shapes in the above formulas for the calculation of the flexible response.

5.4.2 Fluid solver

SEACAL's fluid solver, HYDCAL, is based on potential flow theory, therefore, the considered fluid flow is non-viscous, irrotational, and incompressible and it satisfies the Laplace equation:

$$\boldsymbol{\nabla}^2\phi = 0. \tag{5.6}$$

The set of equations obtained from Laplace equation (5.6) and boundary conditions, i.e. kinematic free-surface and dynamic, are solved by using the boundary element method (BEM). Once the velocity potential is obtained, the pressure p and wave elevation ζ respectively can be computed using the Bernoulli equation and kinematic free-surface boundary condition. Since SEACAL solves the first-order linear quantities, therefore, the first-order variations of pressure and free surface elevation respectively are given as

$$p^{(1)} = -\rho\Big(\frac{\partial\phi^{(1)}}{\partial t} + \hat{\boldsymbol{\nabla}}\phi_s \cdot \hat{\boldsymbol{\nabla}}\phi^{(1)}\Big), \tag{5.7}$$

and

$$\zeta^{(1)} = -\frac{1}{g}\Big(\frac{\partial\phi^{(1)}}{\partial t} + \hat{\boldsymbol{\nabla}}\phi_s \cdot \hat{\boldsymbol{\nabla}}\phi^{(1)}\Big), \tag{5.8}$$

where ϕ_s is the steady part steady part while ϕ is the unsteady (time-dependent) part. In this case, there is no steady flow as the beam is not moving with forward speed therefore we are neglecting ϕ_s. After calculating the pressure, the global fluid forces on the structure are then computed by integrating the pressure over the structure's surface. Once the pressures and forces are known, the next step is to compute hydrostatic and hydrodynamic coefficients which are required to solve the equation of motion. The hydrostatic restoring (stiffness) matrix C has a contribution from the hydrostatic pressure C_h and mass elements C_M.

$$C = C_h + C_M \tag{5.9}$$

The contribution due to the hydrostatic pressure is calculated either by summation over the panels (hydrodynamic mesh) or by summation over the Gauss points (integration mesh).

The hydrodynamic coefficients, i.e. added mass and damping coefficients, are calculated from the radiation forces. The radiation forces are obtained by integrating the pressure over the

structure's surface S after taking into account the local normal vector $\vec{n}$ and local modal displacement H, as follows:

$$\vec{F}^{(rad)} = \int\int_S p^{(rad)}(\vec{n} \cdot \vec{H})\mathrm{d}S. \tag{5.10}$$

After computing the radiation force, we can calculate added mass and damping coefficient by using the known encounter frequency ω_e, as follows

$$A_{ij}(\omega_e) = \frac{\mathrm{Re}(F_{ij}^{(rad)}(\omega_e))}{\omega_e^2} \quad \text{and} \quad B_{ij}(\omega_e) = \frac{\mathrm{Im}(F_{ij}^{(rad)}(\omega_e))}{\omega_e}. \tag{5.11}$$

Once all the required input matrices and coefficients to compute the structure's response are known, the next step is to solve the equation of motion, which is described in the next subsection.

5.4.3 Beam's response calculation

SEACAL has a subprogram, RESCAL, to compute the structure's response. RESCAL solves the linear equation of motion in the frequency domain to compute motion and internal loads. The general form of which is given as

$$\Big[-\omega_e^2(M + A_L) + i\omega_e B_L + C_L\Big]x = F_L, \tag{5.12}$$

where ω_e is encounter frequency; x is the $N \times 1$ motion response vector and N is the number of degrees of freedom; M is the $N \times N$ mass matrix; A_L is $N \times N$ linear added mass matrix; B_L is $N \times N$ linear damping matrix. The linear added mass and damping matrices are calculated by using potential flow equations. C_L is $N \times N$ linear restoring matrix, which is calculated from hydrostatics; and F_L is $N \times 1$ linear excitation force vector. Equation (5.12) can be written as

$$S(x)x = F(x), \tag{5.13}$$

where $S(x)$ represents the linear system of matrices at the left-hand side of (5.12) and $F(x)$ represents the linear right-hand side vector of (5.12).

5.5 Results and comparison

In this section, we present the comparison of the results obtained from SEACAL with the experimental tests. Before proceeding with the FSI simulation, it is important to ensure the

accuracy of the beam's FEM model by comparing the numerically obtained natural period and fundamental frequency with the experimental results. Therefore, the natural period of the FEM model should be close to the natural period of the actual beam when it is in the air. The results obtained from such comparison are shown in Table. 5.3.

Table 5.3: Comparison of the beam's natural periods and angular frequencies corresponding to the first natural mode are obtained from the experiments and SEACAL calculations when the beam is in the air and submerged in the water.

	Experiment		SEACAL	
	T_p	ω	T_p	ω
	[s^{-1}]	[rad/s]	[s^{-1}]	[rad/s]
Dry hammer test	3.6	22.62	3.606	22.657
Wet hammer test (0.25m)	2.34	14.70	2.390	15.017
Wet hammer test (0.5m)	1.72	10.81	1.697	10.664

The relative percentage error between the numerical and experimental results for the natural periods of the beam in the air is only 0.16%, which shows that the FEM model is an accurate representation of the actual beam's structure. However, it can be noticed that the relative error is -2.09% and 1.369% when the beam's submerged depth is 0.25m and 0.5m, respectively. The error is slightly larger as compared to the dry mode values when the beam is submerged because the added mass values computed by SEACAL are different from the experimental ones. Furthermore, we can notice that, in both experimental and numerical results, the natural period of the beam increases as it moves from a less dense medium (air) to a more dense medium (water) and is then submerged deeper in the more dense medium (water). This is due to an increase in the added mass and damping introduced by the medium. In the next subsections, we will show the comparison of SEACAL results with the experimental data. For the beam's response, we choose to compare the results obtained from the accelerometer at the submerged free-end of the beam because it shows maximum accelerations.

5.5.1 Regular-wave and beam interactions when the beam is submerged at 0.25m.

All the wave conditions tested in the first subcase of the experiments are simulated in SEACAL. SEACAL computes the RAO values for all six accelerometers at the input values of angular wave frequencies or encounter frequencies, and then the relation mentioned in (5.1) can be used to compute the beam's response. In this report, we consider the RAO values and response of

the accelerometer which is attached to the submerged end of the beam because the accelerations of the submerged end are of higher order as compared to the other locations, as shown in Fig. 4.13. The results obtained from the experiments and SEACAL are listed in Table 5.4.

Table 5.4: Comparison of the results obtained from the linear FSI numerical solver are compared with the experimental results.

	Given Initials			SEACAL		Experiments	
	A	T_p	ω	RAO	Response	RAO	Response
	[m]	[s]	[rad/s]	$[1/s^2]$	$[m/s^2]$	$[1/s^2]$	$[m/s^2]$
1	0.063	1.000	6.283	10.697	0.674	9.619	0.606
2	0.138	1.500	4.189	2.117	0.292	1.964	0.271
3	0.0074	0.500	12.566	102.380	0.758	87.838	0.650
4	0.0327	1.000	6.283	11.415	0.373	9.083	0.297
5	0.069	1.500	4.189	2.117	0.146	1.739	0.120
6	0.123	2.000	3.142	0.710	0.087	0.423	0.052
7	0.195	2.500	2.513	0.298	0.058	0.077	0.015
8	0.0069	0.580	10.833	75.914	0.524	53.623	0.370

A comparison of the RAO values obtained from the SEACAL and experiments at the beam's submerged end is plotted in Fig. 5.7 while Fig. 5.8 shows the comparison of RAO values obtained from accelerometers at other locations of the beam.

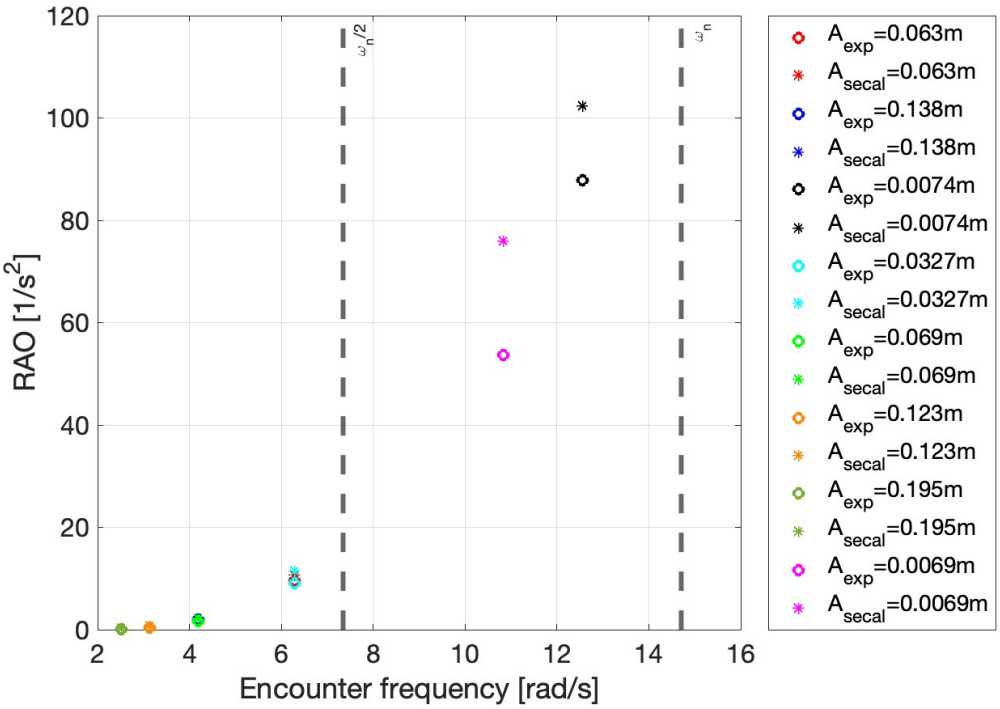


Figure 5.7: The comparison of response amplitude operator (RAO) obtained from SEACAL and experimental data when submerged depth is 0.25m is presented. The values of the first fundamental frequency, i.e. ω_n and $\omega_n/2$, are represented by vertical grey dashed lines.

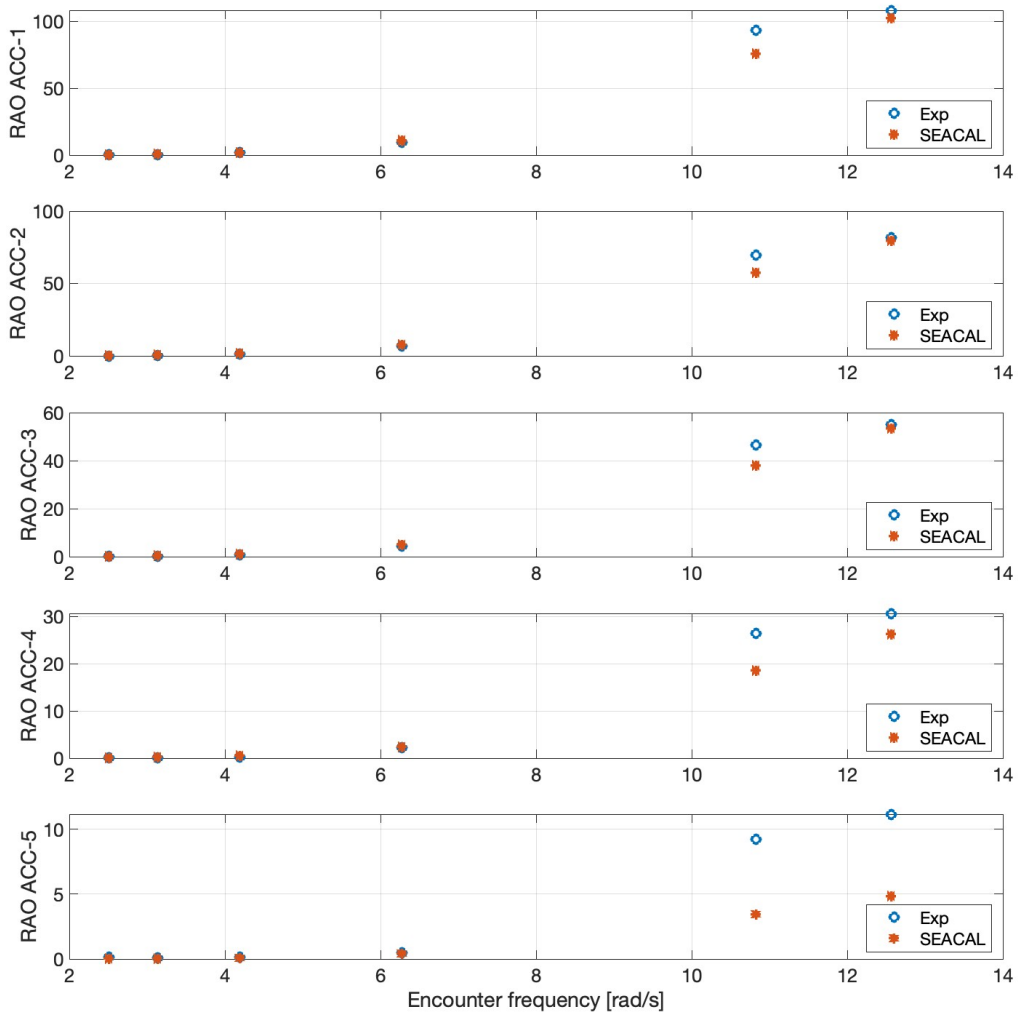


Figure 5.8: The comparison of response amplitude operator (RAO) $1/s^2$ obtained from SEACAL and experimental data for all accelerometers when submerged depth is 0.25m is presented.

It can be noted in Fig. 5.7 that results agree well when the wave encounter frequency is lower than half of the beam's resonance frequency, i.e. from 2.513 to 6.28 rad/s. Through harmonic analysis, it is found that the beam's response is linear for the mentioned range of frequencies, therefore SEACAL's predictions are accurate. An example of linear harmonic response is demonstrated in Fig. 5.5. However, as the wave encounter frequency approaches the resonance frequency, the beam resonates at the resonance period and the deformations get large, as a linear solver, SEACAL's computations deviate from experimental data. Fig. 5.9 shows the harmonic analysis of the experimental results of the incident wave and beam response at the encounter frequency of 10.83 rad/s and 12.56 rad/s. The amplitudes of the

response signals are significantly higher than the amplitude of the incident wave signals. This shows that these incident waves excited the resonance frequency of the beam.

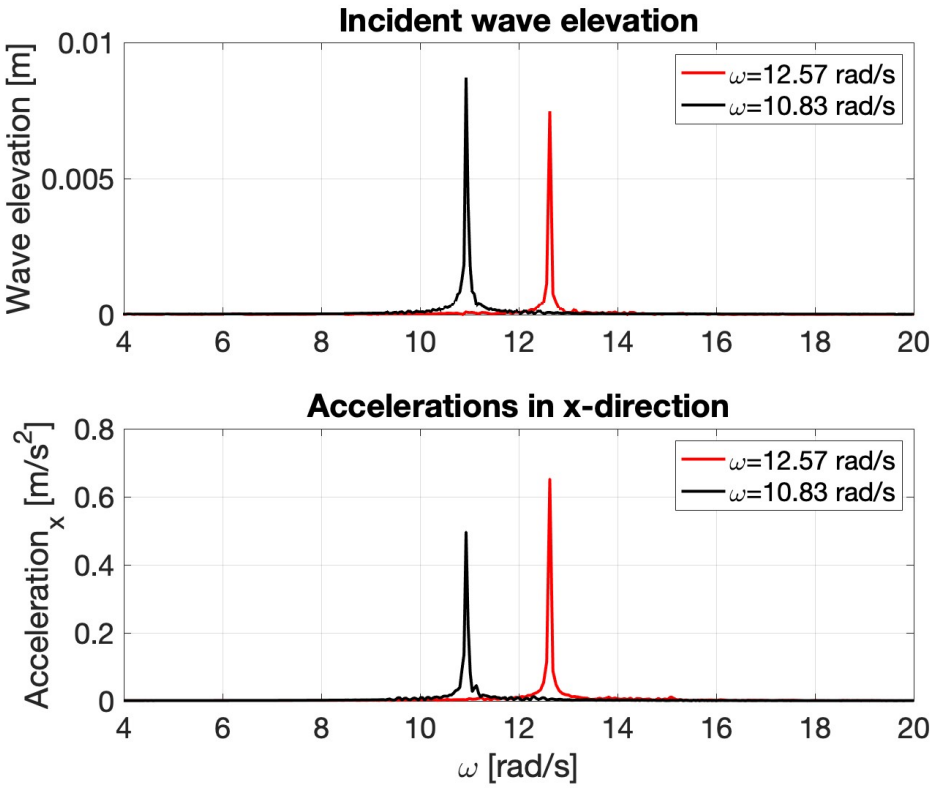


Figure 5.9: Harmonic analysis of incident wave signals (top) and corresponding beam's response (bottom) in the frequency domain are shown. The beam's response towards two waves of the same amplitude but different frequencies, i.e. amplitude 0.007m and the encounter frequencies of 10.83 rad/s and 12.56 rad/s, is shown.

Fig. 5.10 shows the comparison of the beam's response obtained from SEACAL and experimental data. After visual inspection of Fig. 5.10, it can be noticed that the response predicted by SEACAL is higher than in actual experiments. The reason is, SEACAL's fluid solver is based on potential flow theory, therefore, it neglects the viscous effect and predicts higher fluid loads on the beam thus resulting in higher values of RAO and the beam's response. A mere visual inspection of the results could be misleading and therefore one needs to check the relative error, which is presented in Fig. 5.11.

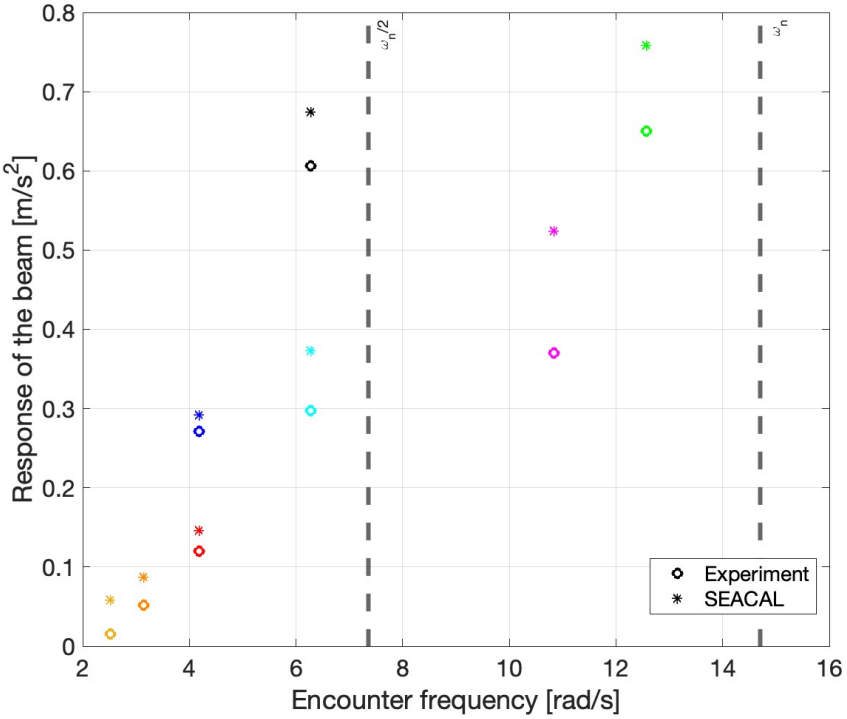


Figure 5.10: Comparison of the beam's response calculated from SEACAL with the measurements obtained from the experiments is shown. Each colour represents a specific wave height corresponding to wave encounter frequency.

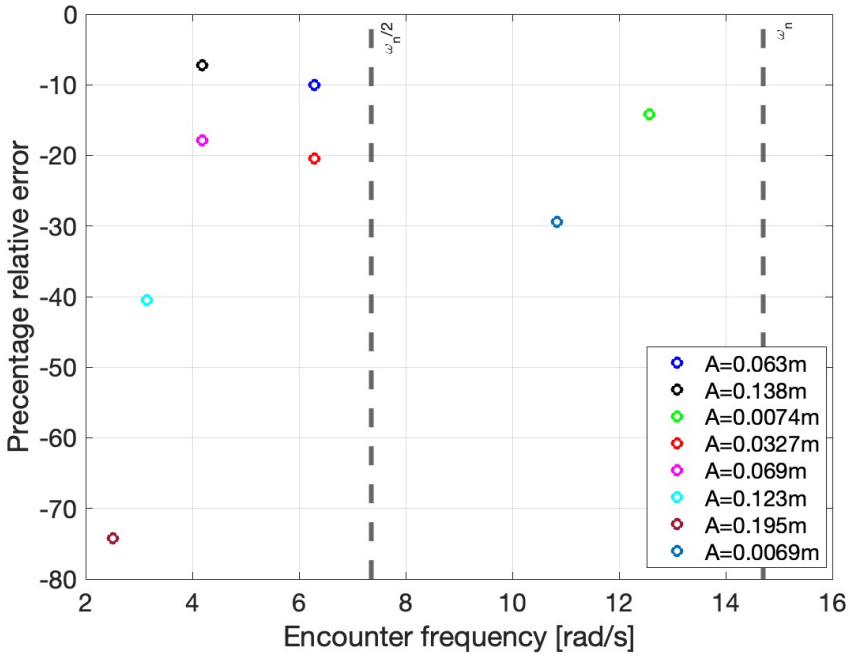


Figure 5.11: Percentage relative difference of experimental values of beam's response with respect to SEACAL's results for the corresponding wave amplitudes and encounter frequencies is shown.

Fig. 5.11 shows the percentage relative difference in the experimental and numerical results. This way the effect of wave height and wave encounter frequency on the result's discrepancy becomes apparent. The plot in Fig. 5.11 depicts that five of eight test cases have a relative error of less than 20%. The two test cases which showed the highest discrepancy, i.e. percentage relative error of -40% and-74%, are the waves with amplitudes of 0.123m and 0.195m with the encounter frequencies of 3.14 rad/s and 2.51 rad/s interact with the beam. Therefore, harmonic analysis in the frequency domain is performed to investigate the reason for the high discrepancy. The harmonic analysis is shown in Fig. 5.12.

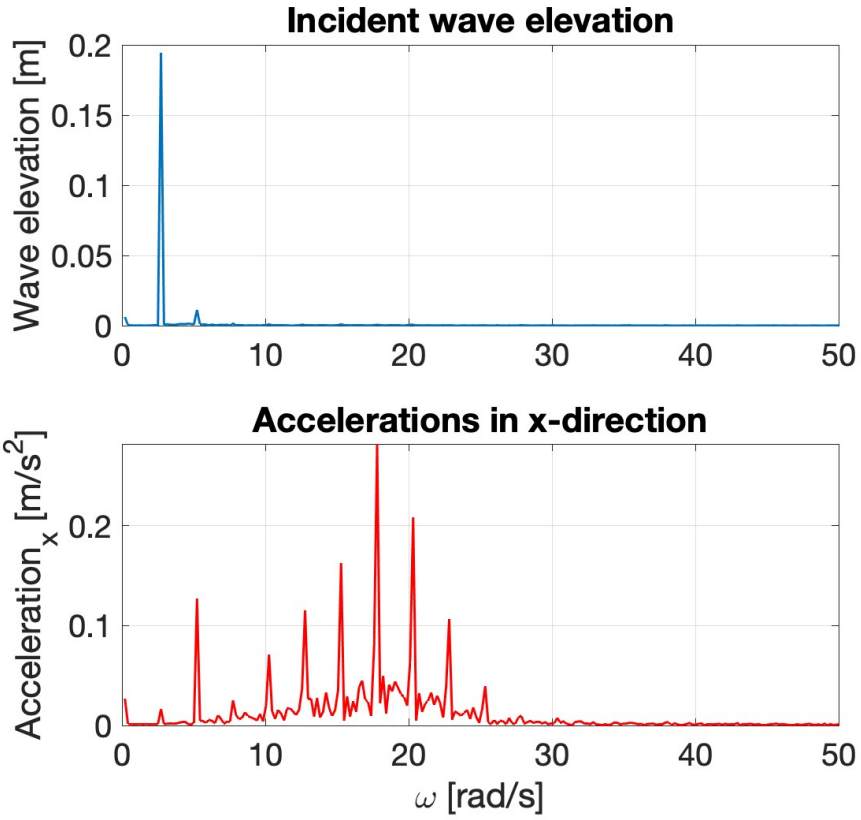


Figure 5.12: Harmonic analysis of incident wave signals (top) and corresponding beam's response (bottom) in the frequency domain are shown. The nonlinear response of the beam towards the wave with the highest relative error, i.e. amplitude 0.195m and the encounter frequency of 2.51rad/s is shown.

In Fig. 5.12, the top plot shows the amplitudes and encounter frequencies of the incident waves while the bottom plot shows the amplitudes and frequencies of the beam's response. The input wave signal is linear as there is one dominant regular wave with an angular frequency of 2.51rad/s. However, the beam's response is highly nonlinear, as multiple high-order modes got excited and the amplitudes of the high-order modes are much higher than the first mode.

To further elaborate, in Fig. 5.13, time domain signals of the incident wave and the resulting beam's response (accelerations in x directions) measured from all accelerometers are plotted. The beam's multiple-frequency response can be seen in the time-domain plots.

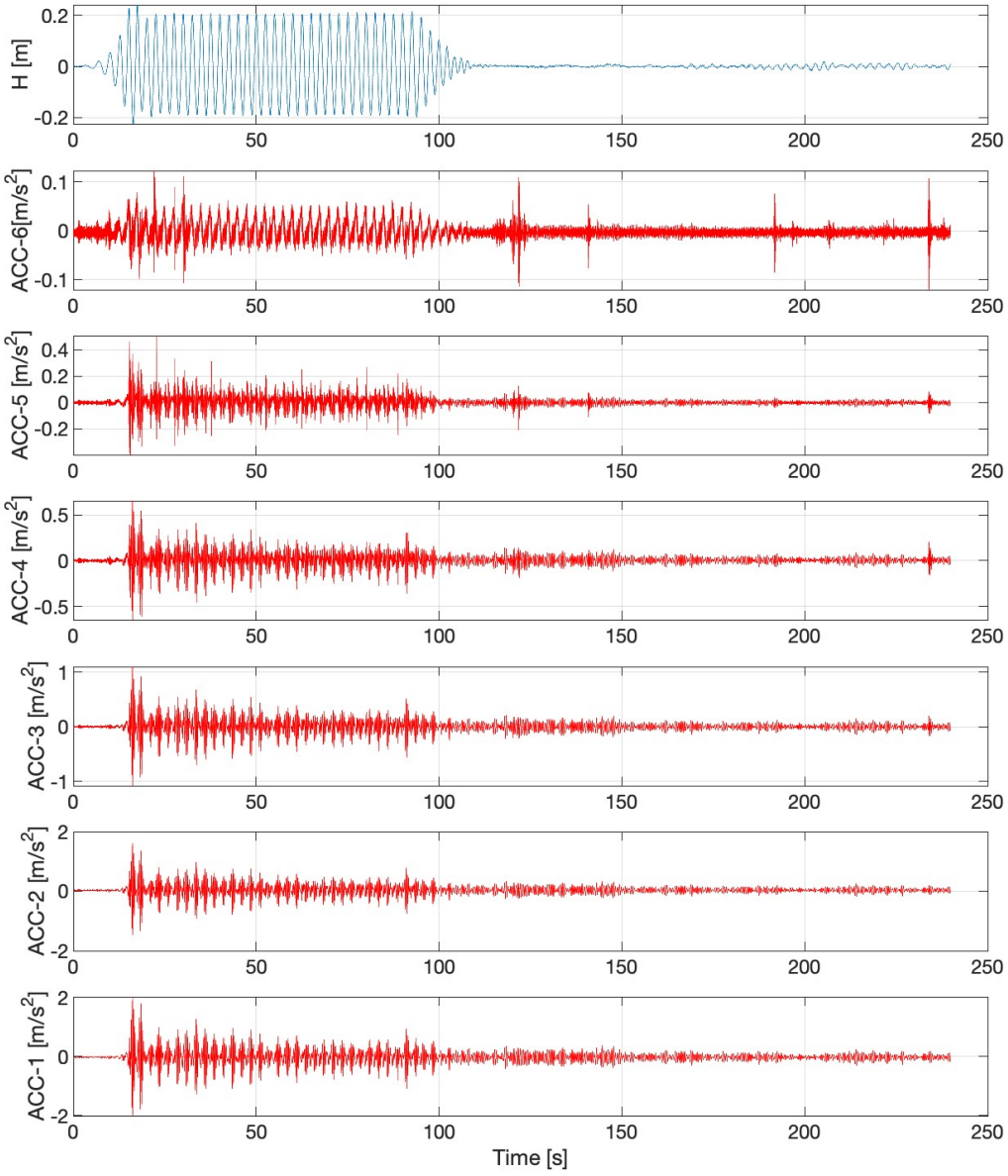


Figure 5.13: The time domain signals of the incident wave and beam's response (accelerations in x directions) measured from all accelerometers are plotted.

Hence, as a linear solver, SEACAL is incapable of computing such a response. A similar kind of nonlinear beam response is seen upon harmonic analysis of the wave with 0.123m amplitude

and 3.14 rad/s frequency.

5.5.2 Regular-wave and beam interactions when the beam is submerged at 0.5m.

All the wave conditions tested in the second experimental subcase are simulated in SEACAL to obtain the RAO values for the range of given angular wave/encounter frequencies and then the relation mentioned in (5.1) is used to compute the beam's response. Due to an increase in the beam's submerged depth, from 0.25m to 0.5m, the beam's resonance frequency decreases respectively from 14.70rad/s to 10.81rad/s. Hence, making it possible to study the regular-wave interaction with the beam when wave-encounter-frequency is larger than the beam's resonance frequency. The wave parameters and the results obtained from experiments and SEACAL are listed in Table 5.5.

Table 5.5: Comparison of the results obtained from the linear FSI numerical solver are compared with the experimental results.

	Given Initials			SEACAL		Experiments	
	A	T_p	ω	RAO	Response	RAO	Response
	[m]	[s]	[rad/s]	$[1/s^2]$	$[m/s^2]$	$[1/s^2]$	$[m/s^2]$
1	0.0044	0.500	12.560	72.920	0.321	80.682	0.355
2	0.064	1.000	6.280	14.872	0.952	14.313	0.916
3	0.12	1.500	4.190	3.312	0.397	3.692	0.443
4	0.007	0.500	12.560	72.920	0.510	94.143	0.659
5	0.033	1.000	6.280	14.872	0.491	13.818	0.456
6	0.057	1.500	4.190	3.312	0.189	3.579	0.204
7	0.12	2.000	3.140	1.166	0.140	0.867	0.104
8	0.0062	0.580	10.830	147.025	0.912	154.516	0.958

A detailed comparison of the RAO values obtained from the experiments and SEACAL is shown in Fig. 5.14. Fig. 5.14 has four sub-plots and each subplot corresponds to a different accelerometer. The topmost subplot shows the RAO values of all test cases calculated from the accelerometer located at the submerged free end of the beam. SEACAL did not yield any RAO value for the two accelerometers located around the beam's fixed end, therefore, comparison plots for ACC-5 and ACC-6 are not shown.

Fig. 5.15 and 5.16 and show the comparison of numerical results, obtained from SEACAL, with the experimental data. Fig. 5.15 shows the comparison of RAO values while the comparison of beam's response is shown in Fig. 5.16.

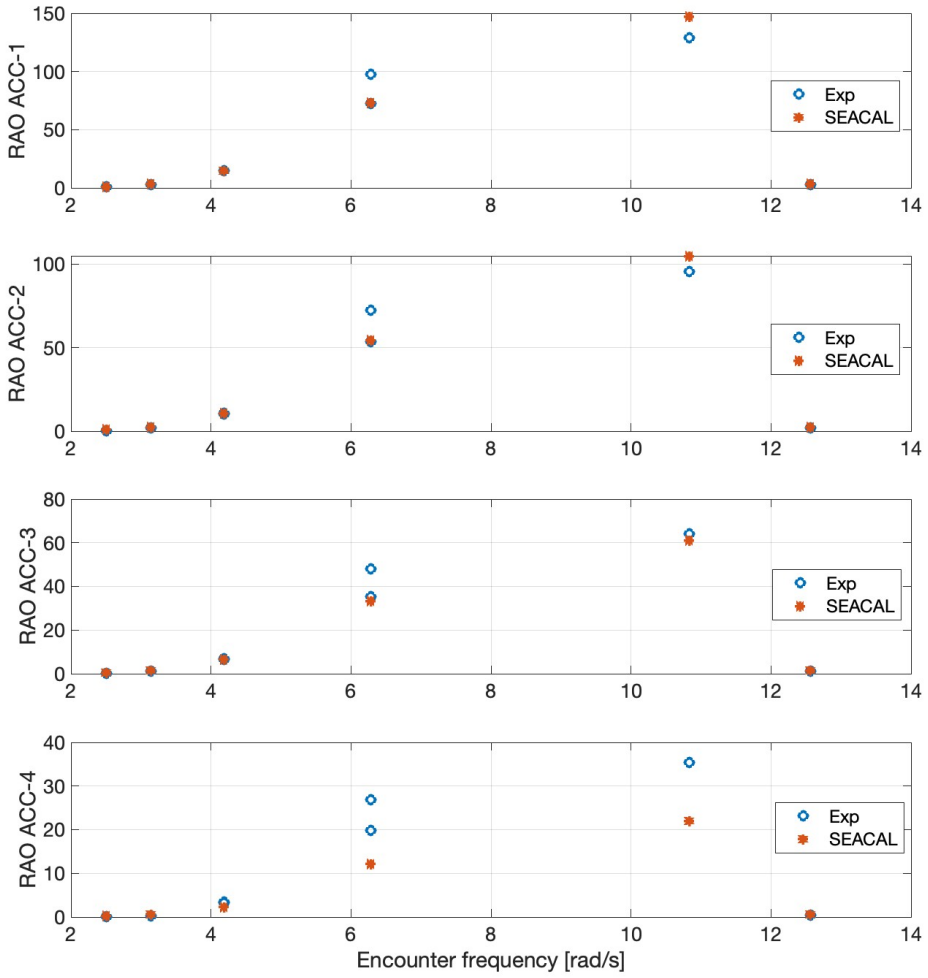


Figure 5.14: The comparison of response amplitude operator (RAO) $1/s^2$ values obtained from SEACAL and experimental data at different wave amplitudes and encounter frequencies is shown.

Unlike the first subcase and our expectations, we noticed from Fig. 5.16 that SEACAL under-predicted the value of RAO and thus beam's response for five of eight test cases. The difference in this subcase is that the beam is submerged deeper and therefore the effects due to hydrodynamics are more significant, which might have excited the high-order frequency response in the beam which SEACAL is unable to predict because of the limitations of the linear model. Further in this section, we will perform a harmonic analysis of the test cases with the highest deviation and testify our deduction. Hence, the first step is to quantify the relative error and identify the cases with the highest value of relative error. Fig. 5.17 shows a plot of percentage relative error.

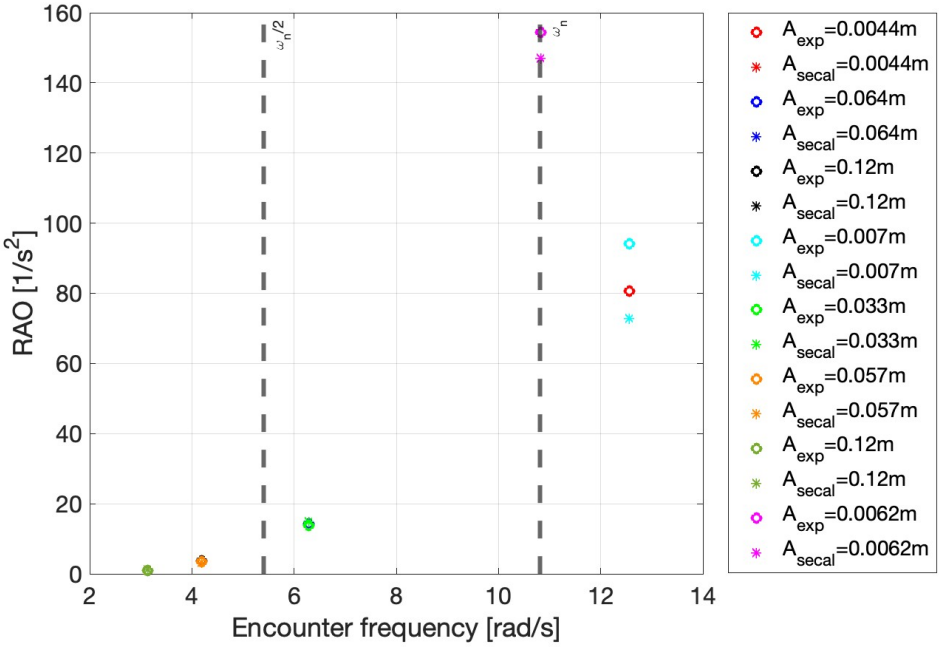


Figure 5.15: The comparison of response amplitude operator (RAO) values obtained from SEACAL and experimental data at different wave amplitudes and encounter frequencies is shown.

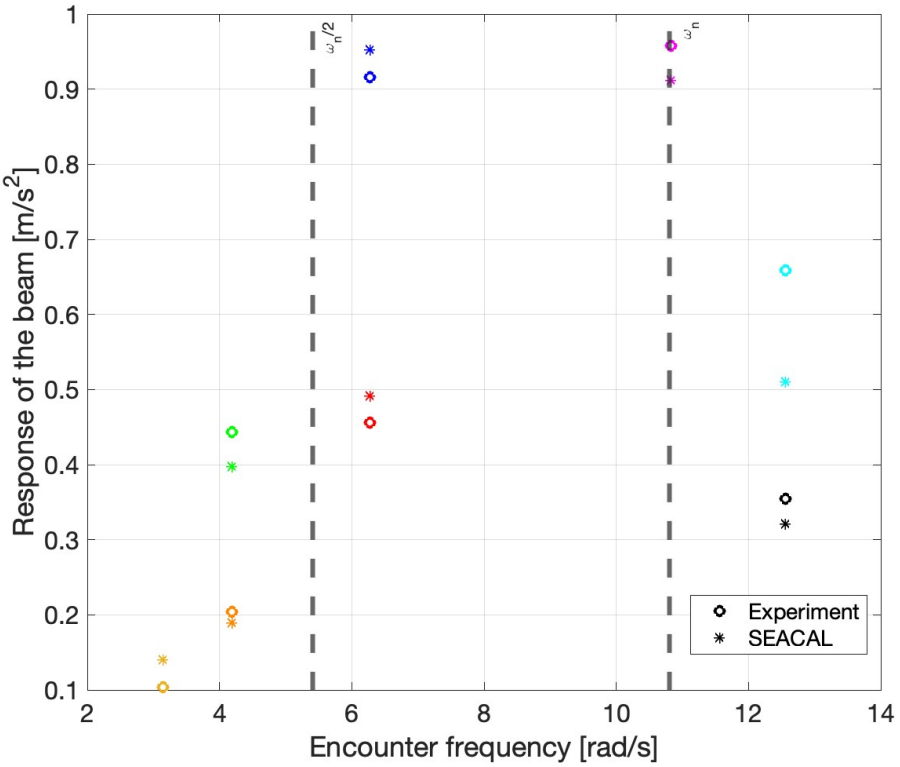


Figure 5.16: Comparison of the beam's response calculated from SEACAL with the measurements obtained from the experiments is shown. Each colour represents a specific wave amplitude corresponding to wave encounter frequency.

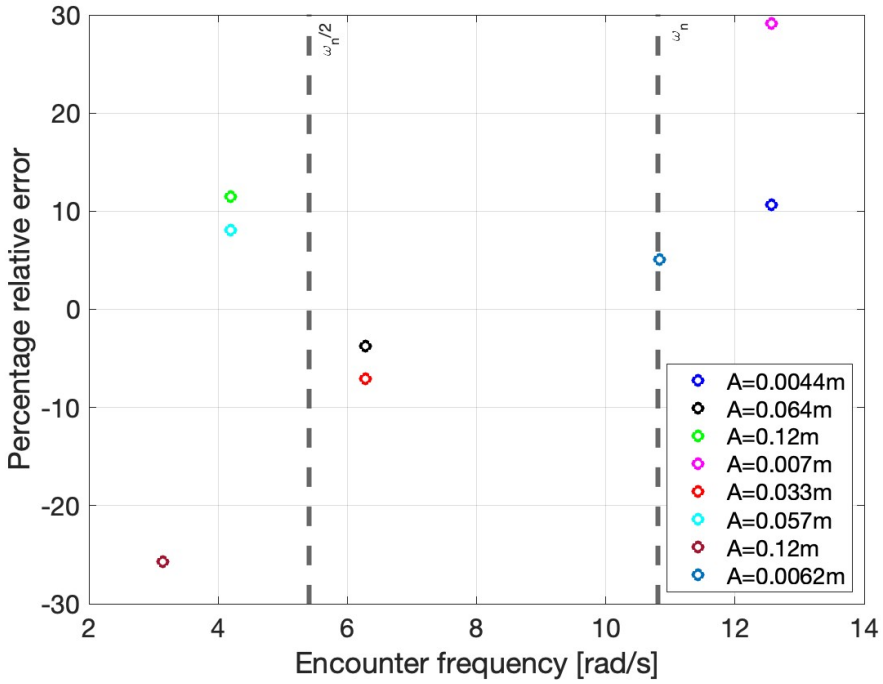


Figure 5.17: Percentage relative error of experimental values of beam's response with respect to SEACAL's results for the corresponding wave amplitudes and encounter frequencies is shown.

Fig. 5.17 shows that six of eight test cases have a percentage relative error of around 10%. Only two cases have a relative error of more than 10%, i.e. $\approx$ 29% for the wave with 0.007 m amplitude and 12.56 rad/s frequency, and $\approx$ -25% for the wave with 0.12m amplitude and frequency of 3.14rad/s. We will investigate these two cases by performing harmonic analysis. In addition to this observation, we have observed that whenever two waves of different amplitudes are tested at the same frequency, the relative error for the wave with a high amplitude is higher than the wave with a small amplitude. For example, we tested two incident waves of different amplitudes at the same encounter frequency, i.e. 6.28 rad/s and the high-amplitude wave, i.e. 0.064m, showed higher error as compared to the low-amplitude wave i.e. 0.033m.

Fig. 5.18 shows the harmonic analysis of the incident wave and beam response corresponding to the test case with the highest relative error of $\approx$ 29%. The beam's response shows one peak at the encounter frequency of the incident wave. This shows the resonance response of the beam.

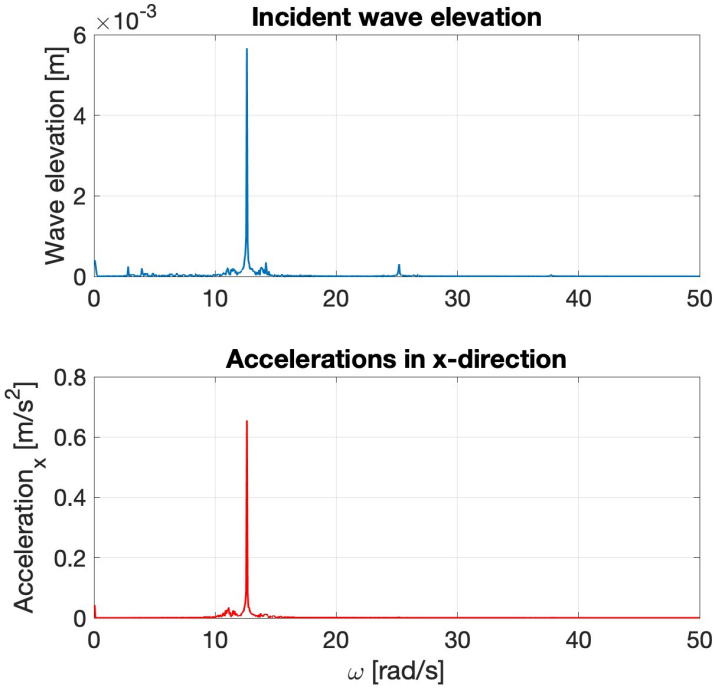


Figure 5.18: The beam's response (bottom) to the water waves (top) with the highest discrepancy, i.e. ≈ 29%, is shown in the frequency domain. This case corresponds to the wave with an amplitude of 0.007m and an encounter frequency of 12.56 rad/s. The incident wave is shown in the top plot while the beam response is shown in the bottom plot.

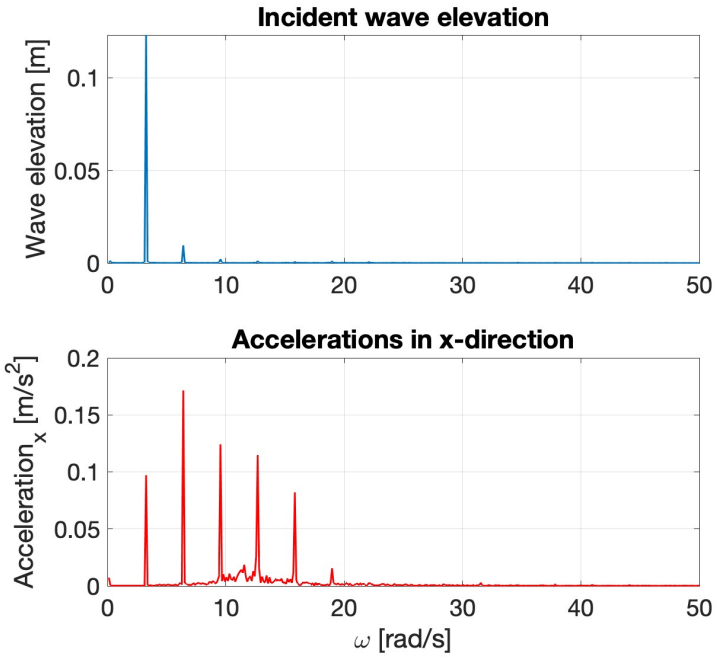


Figure 5.19: The beam's response (bottom) to the water waves (top) with the second highest discrepancy, i.e. ≈ -25%, is shown in the frequency domain. This case corresponds to the wave amplitude of 0.12m with an encounter frequency of 3.14 rad/s. The incident wave is shown in the top plot while the beam response is shown in the bottom plot.

In Fig. 5.19, results from the harmonic analysis show that the harmonic loading from the incident wave has excited the high-frequency modes whose amplitudes are comparable to the first fundamental mode. This indicates that the beam's response is nonlinear. To further elaborate, in Fig. 5.20, time domain signals of the incident wave and the resulting beam's response (accelerations in x directions) measured from all accelerometers are plotted. The beam's multiple-frequency response can be seen in the time-domain plots.

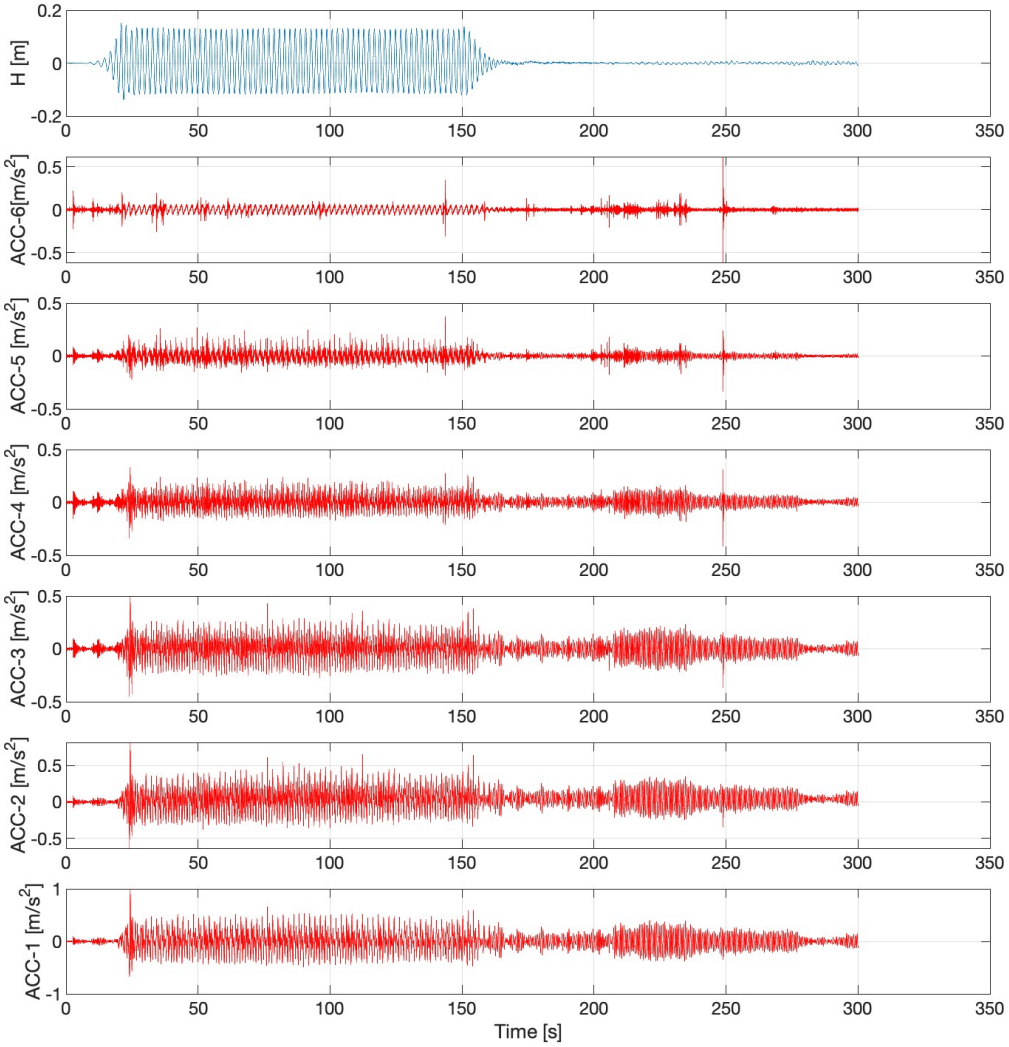


Figure 5.20: The time domain signals of the incident wave and beam's response (accelerations in x directions) measured from all accelerometers are plotted.

5.6 Conclusion

This work presents a comparison study of experimental data with SEACAL, an in-house linear FSI solver used at MARIN. The purpose is to validate SEACAL's results for a range of regular wave conditions and parameters. The presented study has two experimental subcases based on the submerged depth of the beam, i.e. 0.25m and 0.5m. It is noticed that increasing the submerged depth from 0.25m to 0.5m reduces the natural frequency of the beam, due to an increase of the added mass, thus allowing to perform comparison at resonance.

In the first subcase, when the submerged depth is 0.25m, the experimentally measured beam's resonance frequency is 14.7rad/s which is higher than the encounter frequency of all the wave conditions tested in that subcase. Therefore, it is observed from the harmonic analysis that, for most of the tests, the beam's response remained linear, i.e. the response obeyed the principle of superposition and the harmonics are integer multiples of the fundamental frequency and their amplitude decreases as the frequency increases. An example of linear response is shown in Fig. 5.5. In this subcase, the absolute error when the beam showed a linear response is less than 20%. However, when the encounter frequencies are 2.51rad/s and 3.14rad/s, the absolute error is greater than 30% for both test cases. Upon further investigation by performing harmonic analysis, it is revealed that these two cases excited the high-frequency modes of the beam and being a linear solver SEACAL could not predict the beam's response accurately.

In the second subcase, when the submerged depth is 0.5m, the experimentally measured beam's resonance frequency is 10.81rad/s, thus allowing us to study beam's response when the wave encounter frequency is close to and greater than the beam's resonance frequency. Therefore, the nonlinear resonance response of the beam could be excited by more wave conditions as compared to subcase 1. Six of the eight test cases showed a linear response and the absolute error remained around 10%, as shown in Fig. 5.17. However, when the beam's response is nonlinear, the maximum relative error is around 30% for the wave with an amplitude of 0.12m and an encounter frequency of 3.14rad/s. The cause of the large discrepancy is similar to the first subcase, i.e. excitation of high-frequency modes which has amplitudes larger than the fundamental low-frequency mode.

As SEACAL is a linear solver whose fluid solver is based on potential flow theory, therefore, the major cause for the discrepancy is due to the elimination of high-order terms and viscous

effects. However, the software is fast as the simulation time for all wave conditions is within five minutes and, according to the experts at MARIN, the discrepancy of the results, when the beam's response is linear, is in the acceptable range for the initial design testing, i.e., around 10% to 15% relative error. In the test cases when beam's nonlinear response is excited, the discrepancy in the results is over 35% which shows that a nonlinear FSI solver should be used. Therefore, in Chapter 6, we will use high-fidelity Reynold Averaged Navier Stokes equation (RANSE)-based ReFRESCO code [57] to simulate nonlinear wave-beam interactions.

Chapter 6

High-fidelity fluid structure interactions modelling of regular and irregular water waves

6.1 Introduction

In this chapter, we present numerical modelling of the fluid-structure-interaction (FSI) problem, i.e. regular and irregular water waves' interactions with a flexible beam, by coupling Reynold's Averaged Navier Stokes Equations (RANSE) based solver with a finite element model (FEM) of the flexible beam such that the beam's deformations could be large or nonlinear while the interaction between the FEM nodes is modelled as a linear spring system. This study aims to validate numerical results produced by MARIN's in-house high-fidelity FSI solver, i.e. ReFRESCO (Reliable & Fast RANS Equations Code for Ships and Constructions Offshore) [57], by comparing numerical results with the test cases from experimental case 1 and case 3. For the sake of completeness of this report, we will first briefly explain the experimental setup. The experimental set-up consists of a flexible PVC beam which is attached to the basin carriage, the beam is equipped with six accelerometers which are located equidistantly. There are two probes, indicated as probe 1 and probe 2, as shown in Fig. 6.1. Probe 1 is located in front of the beam while probe 2 is in parallel to the beam. Both probes record the total amplitude of waves which are resultant of incoming waves from the wavemaker and diffracted waves from the beam. Thus the designed experimental setup is capable of measuring the wave amplitude and

the resulting beam's response in a simultaneous fashion which makes the setup suitable for FSI study.

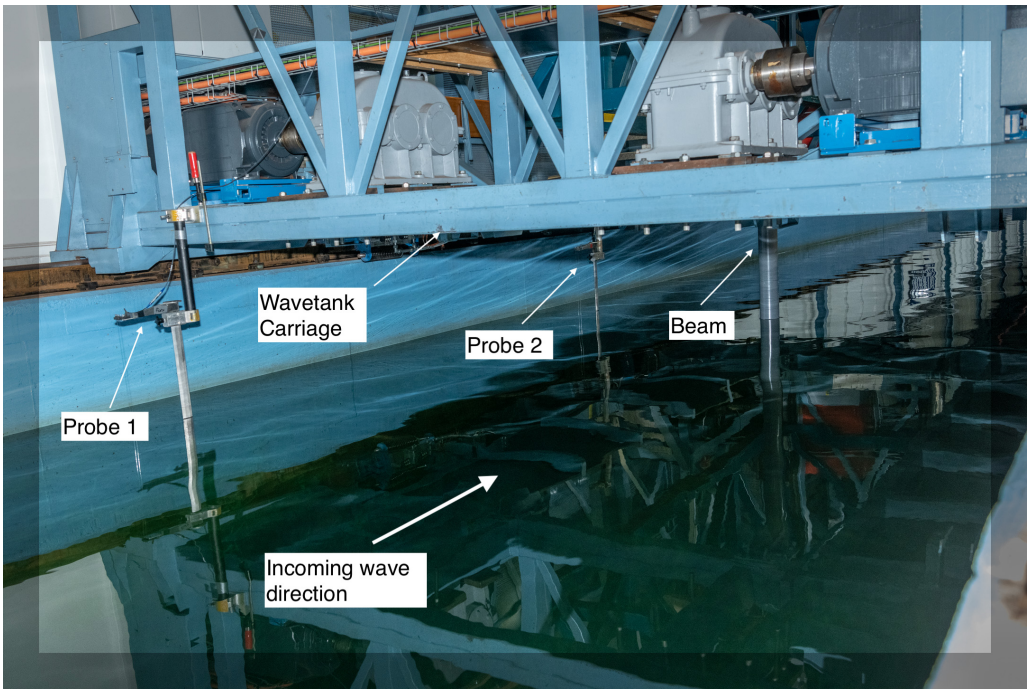


Figure 6.1: The experimental set-up for the FSI study is shown. Photo courtesy MARIN.

Before sharing the results from the comparison we would briefly explain the theoretical background of RANSE, modal analysis, and FSI.

6.1.1 RANSE-based fluid model

The basic theoretical concept of RANSE-based models is explained in §2.2.2 of this report. As you know turbulent viscosity in RANSE should be modelled to close the system of equations. However, in ReFRESCO, instead of using turbulence models we have applied a slip-wall condition at the beam's surface. The CFD software used in this project is ReFRESCO which is a RANSE-based solver. ReFRESCO is capable of handling intricate geometries and complex physics as it includes innovative meshing techniques and the latest physical models. Further in this section, we will explain how the mathematical equations which are explained in this section are discretised in space and time by ReFRESCO.

FVM-based numerical modelling of fluid

In this section, the spatial discretisation of the fluid equations is explained. The explanation is based on the theory manual of ReFRESCO [15]. ReFRESCO uses the Finite Volume Method

(FVM) along with the turbulence closure models to solve RANS equations in the integral form. To derive the integral form of the equations (2.3) and (2.4), consider an arbitrary shape control volume Γ that is enclosed by a surface S which has an outward pointing unit normal vector $\vec{n}$. This system is fixed in the absolute reference frame in time. Now, we can integrate the equations (2.3) and (2.4) for the considered control volume and apply Gauss' divergence theorem to obtain the integral form of the conservation equations, as follows:

$$\frac{\partial}{\partial t}\int_{\Gamma}\rho \mathrm{d}\Gamma + \int_{S}(\rho\vec{V})\vec{n}\mathrm{d}S = 0, \tag{6.1}$$

$$\frac{\partial}{\partial t}\int_{\Gamma}(\rho\vec{V})\mathrm{d}\Gamma + \int_{S}\rho\vec{V}\vec{V}\cdot\vec{n}\mathrm{d}S = \int_{S}(\tau_{ij}-P)\cdot\vec{n}\mathrm{d}S + \int_{\Gamma}\rho F_b \mathrm{d}\Gamma. \tag{6.2}$$

Similarly, the integral form of the transport equation is given as:

$$\frac{\partial}{\partial t}\int_{\Gamma}(\rho\phi)\mathrm{d}\Gamma + \int_{S}\rho\phi\vec{V}\cdot\vec{n}\mathrm{d}S = \int_{S}\nu\nabla\phi\cdot\vec{n}\mathrm{d}S + \int_{\Gamma}\rho Q_{\phi}\mathrm{d}\Gamma. \tag{6.3}$$

The derived integral equations can now be discretised by the finite-volume method (FVM). The physical fluid domain is discretised into cells, the integral form of the equations is integrated for each grid cell and dependent variables are defined at each cell centre. The volume integrals are approximated as follows:

$$\int_{\Gamma}\phi\mathrm{d}\Gamma \approx \phi_c\Delta\Gamma, \tag{6.4}$$

where ϕ_c is the value of integral at the cell center and $\Delta\Gamma$ is the cell volume. The surface integrals are approximated as:

$$\int_{S}\phi\mathrm{d}S \approx \sum_{i=1}^{N_f}\phi_{f_i}S_{f_i}, \tag{6.5}$$

which requires the face values ϕ_{f_i} of the integrand by interpolation on cell center data as well as the face area S_{f_i} of all N_f cell faces [15].

Time discretisation

In this section, the temporal discretisation of the spatially discredited fluid equations in ReFRESCO is explained [15]. The time derivative term for a volume control Γ takes the form

$$\frac{\partial}{\partial t}\int_{\Gamma}(\rho\phi)\mathrm{d}\Gamma, \tag{6.6}$$

where ϕ is any scalar, which is discretised by an implicit backward approximation scheme:

$$\frac{\partial}{\partial t}\int_{\Gamma}(\rho\phi)\mathrm{d}\Gamma \approx [c_1(\rho_c\phi_c\Delta\Gamma)^n + c_2(\rho_c\phi_c\Delta\Gamma)^{n-1} + c_3(\rho_c\phi_c\Delta\Gamma)^{n-2}]/\Delta t, \tag{6.7}$$

where n is the time level; Δt is the time step; ρ_c and ϕ_c are the cell centred values of the variables; and c_1, c_2, and c_3 are the coefficients whose value depend on the scheme we want to apply. For example, the values of the coefficients for the first-order backward Euler scheme are $c_1 = 1.0$, $c_2 = -1.0$, and $c_3 = 0.0$ and for the second-order backward scheme, the coefficients are: $c_1 = 1.5$, $c_2 = -2.0$, and $c_3 = 0.5$ [15].

6.1.2 Modal analysis of beam structure

The modal analysis is a type of linear dynamic analysis which is performed to determine the natural frequency and normal mode shapes of the structure by solving the equation of motion [18, 39]. Consider the following equation of motion:

$$[M]\{\ddot{x}\} + [C]\{\dot{x}\} + [K]\{x\} = \{f(t)\}, \tag{6.8}$$

where M is the mass matrix, $\ddot{x}$ is the acceleration vector, C is the linearized damping matrix, $\dot{x}$ is the velocity vector, K is the linearized stiffness matrix, x is the displacement vector, and $f(t)$ is the force vector. The linearized damping and stiffness matrices are derived from the small motion perturbations of the finite element model nodes. The natural frequencies and mode shapes are the natural characteristics of the structure and therefore are independent of time and external loads. For this reason, the equation of motion excludes the external loads and is solved in the frequency domain, this is also called free vibration analysis. The equation of motion for this case is given as:

$$[M]\{\ddot{x}\} + [K]\{x\} = 0. \tag{6.9}$$

Note that we also do not need to include the effect of damping therefore C is not taken into account. This system is referred to as a free and undamped system. (6.9) is still time-dependent as it involves x. To convert (6.9) into frequency domain we assume that each point i of the

structure is moving harmonically as follows:

$$\{x\} = \{A\}_i \sin(\omega_i t + \theta_i), \tag{6.10}$$

where A is the amplitude of the harmonic motion of each point i, ω^2 is the angular frequency which same for all points, and θ is the phase angle of each point. This means, that all points of the structure are moving harmonically such that they have the same angular frequency but can have different amplitudes, and can be out of phase with respect to each other. The double derivative of (6.10) with respect to time is

$$\{\ddot{x}\} = -\omega_i^2 \{A\}_i \sin(\omega_i t + \theta_i). \tag{6.11}$$

Substituting (6.11) and (6.10) into (6.9) we obtain:

$$\Big([K] - \omega_i^2 [M]\Big)\{A\}_i = 0, \tag{6.12}$$

where ω^2 i.e. the eigenvalue, and A i.e. the eigenvector, are the time-independent unknowns of the problem instead of time-dependent x and $\ddot{x}$. Hence, (6.12), which is a classic eigenvalue problem, can be solved in the frequency domain. The angular frequency of the system can be found by solving the non-trivial solution of (6.12), as follows:

$$\det\Big([K] - \omega_i^2 [M]\Big) = 0 \tag{6.13}$$

where, M and K are known matrices. Once the angular frequency is computed, we can substitute it into (6.12) to calculate the mode shape vector or eigenvector A. If M and K are of $N \times N$ dimensions then there are N eigenvalues; for each eigenvalue there is a corresponding eigenvector. The natural frequency is related to the angular frequency of the system by using the following expression:

$$f_i = \omega_i / 2\pi \tag{6.14}$$

Finally, the angular frequencies and amplitudes can be substituted in (6.10) and (6.11) to obtain the displacement and acceleration of the structure.

FEM-based numerical model for modal analysis

The finite element method (FEM) is a numerical method of solving differential equations. The finite element model of the beam is developed at MARIN by using ANSYS Mechanical. The beam model is created by shell elements.

6.1.3 Fluid structure interactions

Fluid-structure interaction is a multi-physics problem which deals with the interactions of fluid flow with the flexible structure in a way that the structure deforms and, as a result, the structural deformations change the initial fluid flow. Thus a FSI problem is a two-way coupled problem, due to the complexity of the underlying physics, investigation is challenging in terms of experimentation, mathematical analysis and numerical modelling. An FSI problem can be categorized either into a monolithic approach or a partitioned approach based on the way equations of fluid and structure are solved. A complete classification of a FSI problem based on the equations-solving procedure is shown in Fig. 6.2.

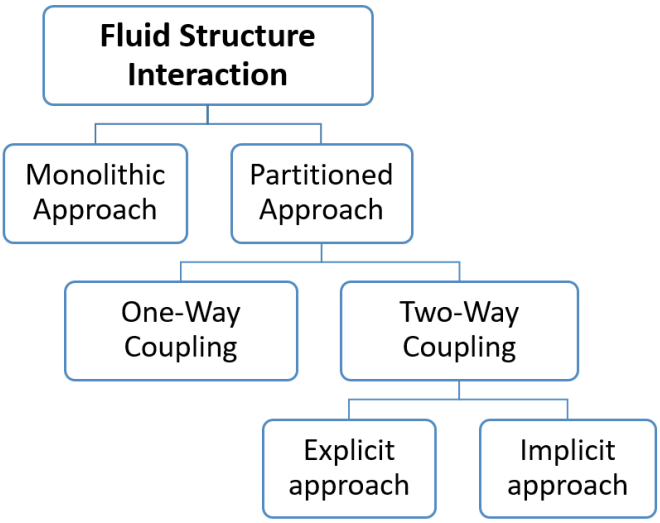


Figure 6.2: Classification of different types of FSI methods [68].

In the monolithic approach, both solid and fluid are treated as one unified system and equations of both systems are solved simultaneously. In the partitioned approach, fluid and solid are treated as two different systems coupled at the interface. The partitioned approach is preferred in engineering applications because it allows the use of independently developed and tested solvers for fluid and structural analysis. The partitioned approach is further subdivided into one-way and two-way coupling. In the one-way coupled FSI analysis, at first, the fluid analysis

is done until convergence then the loads from the fluid are imported to the FEM solver and structural analysis is done until convergence and the simulation stops [6]. Conversely, in the two-way coupled FSI analysis, the process is iterative. After the structural solver computes the geometry displacement, these displacements are imported back into the fluid solver and the fluid analysis is done for the next time step. The process is repeated until the geometry reaches equilibrium. The two-way coupled FSI analysis is further divided into explicit and implicit approaches depending on the way the information is exchanged between the fluid and the structural solvers [76].

6.2 FSI modelling in ReFRESCO

In this section, we explain the procedure by which ReFRESCO solves an FSI problem. The explanation is based on the internal reports [44] and [43]. ReFRESCO's FSI module solves the equation of motion which includes the fluid forces and the structure's deformation; however, structural damping is neglected. Therefore the resulting equation of motion is given as:

$$[M]\ddot{x} + [K]x = F(x), \tag{6.15}$$

where $F(x)$ is the fluid force vector, and $[M]$ and $[K]$ are structure's mass and stiffness matrices. The mass and stiffness matrices are computed by creating the FEM model of the beam in ANSYS and then these matrices are imported into ReFRESCO. Thus the mesh used for the fluid simulation is different from the structural mesh. Therefore, ReFRESCO uses the two-way coupled partitioned approach to solve the FSI problem. The loads computed by the fluid solver are interpolated on the structural mesh at the interface, also called as coupling interface. The coupling of the fluid and structure meshes is performed employing kinematic and dynamic boundary conditions at the coupling interface(γ), which are written as follows:

$$\vec{x}_f = \vec{x}_s \quad \text{on} \quad \gamma; \tag{6.16}$$

$$\vec{p}_f \vec{n}_f = \vec{p}_s \vec{n}_s \quad \text{on} \quad \gamma, \tag{6.17}$$

where $\vec{x}_f$ and $\vec{x}_s$ are the respective fluid and structural displacements; $\vec{p}_f$ and $\vec{p}_s$ are the respective pressure part of the stress tensors at the fluid and structure interface; and $\vec{n}_f$ and $\vec{n}_s$ are the outward normal vectors at the fluid and structure interface, respectively. These boundary

condition, i.e. kinematic and dynamic, states that either the displacement field of the fluid and structure are equal or the stresses due pressure field of the fluid and structure are in equilibrium at the coupling interface. However, if the energy is conserved at the coupling interface then a conservative coupling approach can be used, which is stated in surface integral form as follows:

$$\int_{\gamma_f} \vec{x}_f \cdot \vec{p}_f \vec{n}_f \mathrm{d}S = \int_{\gamma_s} \vec{x}_s \cdot \vec{p}_s \vec{n}_s \mathrm{d}S. \tag{6.18}$$

The FSI module of ReFRESCO monitors the total energy, energy change, and energy loss at each time step. the interpolation matrix can be created by different methods, however, in this comparison study, we have used the Radial Basis Function (RBF) interpolation method which is based on C^2 radial function with compact support.

The flow chart depicted in Fig. 6.3 explains the process of solving the FSI problem in ReFRESCO.

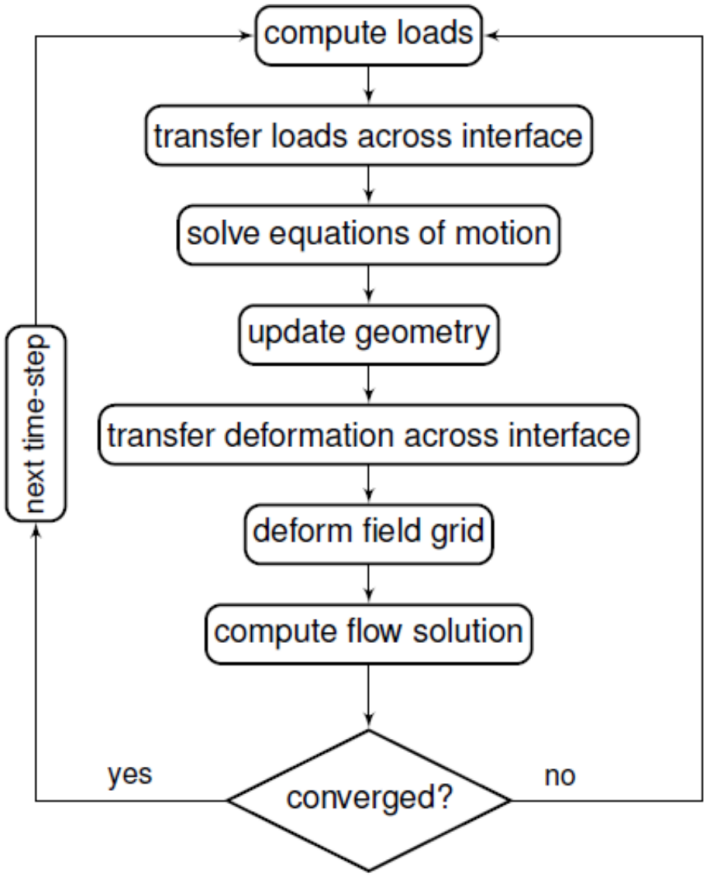


Figure 6.3: A flowchart for the FSI modelling in ReFRESCO [43].

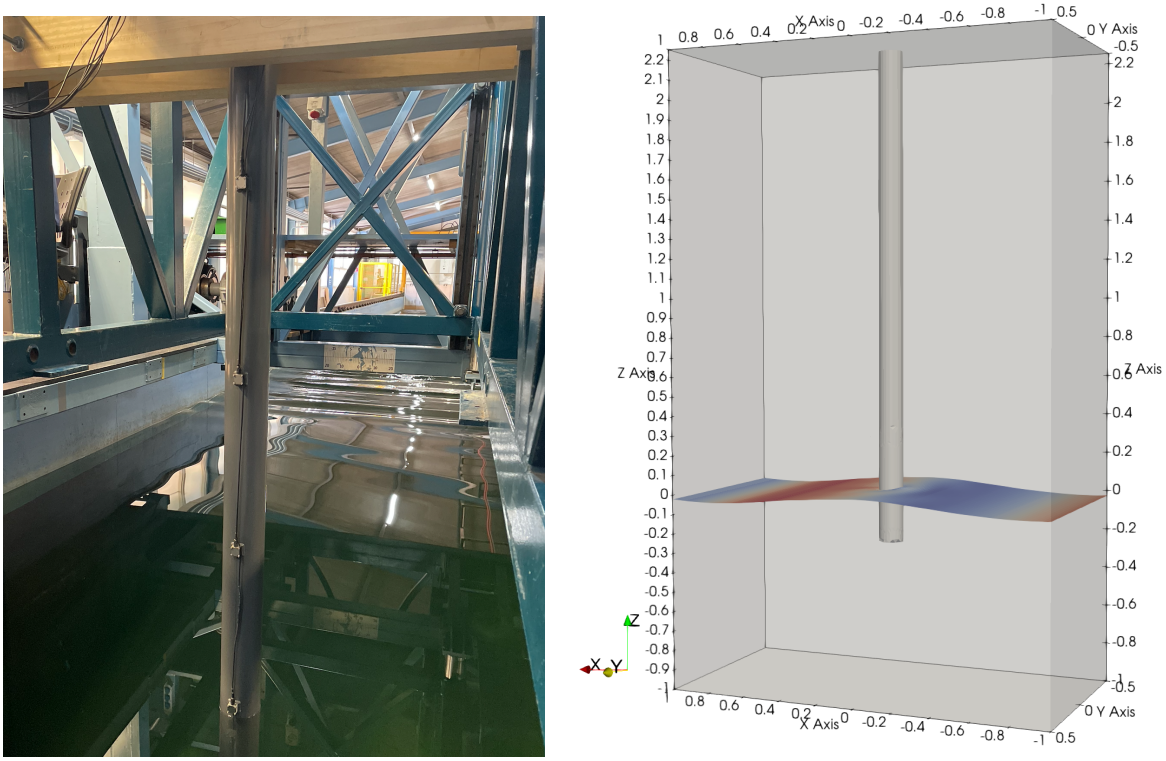
6.3 Setup for numerical modelling

Before explaining the setup of the numerical model, it is important to understand the physical fluid domain. The actual experimental set-up has a fresh water-filled wavetank which is 220m-long, 3.6m-deep, and 4.01m-wide. The wavetank has a waveflap-type wavemaker at one end to generate water waves while the other end has a beach to absorb the incoming waves and reduce the reflected waves. It would be an extremely costly and time-consuming simulation if one tries to model the complete wavetank by using CFD. To make the simulations relatively fast and affordable, we have made simplifications in the numerical model which are explained later in this section. First, we have considered a much smaller computational domain as compared to the actual wavetank. Hence, we have modelled the FSI problem in the region of interest, i.e. beam and the fluid flow in the vicinity of the beam. Second, the model does not include any wavemaker to generate water waves, instead, an inflow boundary condition is applied at one end of the computational domain. The inflow boundary condition requires the wave height and time period as input parameters for the initial condition of the wave. Third, the model does not include a beach to absorb reflected waves, instead, an outflow boundary condition (Sommerfeld) is applied with an absorption zone behind the beam in which the difference between incoming wave and ReFRESCO wave is gradually removed using body forces in momentum equation and free-surface equation. Fourth, a free-slip boundary condition is applied at the bottom of the computational domain because we did not model the actual depth of the basin.

Fig. 6.4 shows a comparison of the experimental set-up (Fig. 6.4(a)) and the numerical set-up (Fig. 6.4(b)) which are used in the study of this FSI problem. The computational domain is shown in Fig. 6.4(b).

6.3.1 Geometry and spatial discretisation of the computational domain

The computational domain consists three-dimensional rectangular domain which has a fluid part and a structure part. The structural part consists of a FEM model of a cylindrical beam whose one end is attached to the top boundary of the rectangular computational domain and the other end is submerged in the water. The water flow inserts load on the submerged end of the beam and the beam deforms in response. The three-dimensional rectangular domain is 2m-long in the x direction, 3m-high in the z direction, and 1m-thick in the y direction. The fluid part of the computational domain is discretised by using finite-volume cells of three different sizes.



(a) Experimental set-up is depicted.

(b) Numerical set-up is shown.

Figure 6.4: A comparison of the experimental and numerical set-up of the FSI problem.

The different cell sizes are used to further optimize the computational time. The computational domain is divided into three zones based on the cell size. Zone-1 has the largest cells and zone-3 has the finest cell while zone-2 has intermediary cell size as it ensures a smooth transition of cells from zone-1 to zone-3. We use fine mesh in zone-3 because it is the part of the domain where fluid interacts with the beam therefore there is a requirement for high accuracy in this region. The spatially discretised numerical model with labelled boundary conditions and cell zones is depicted in Fig. 6.5.

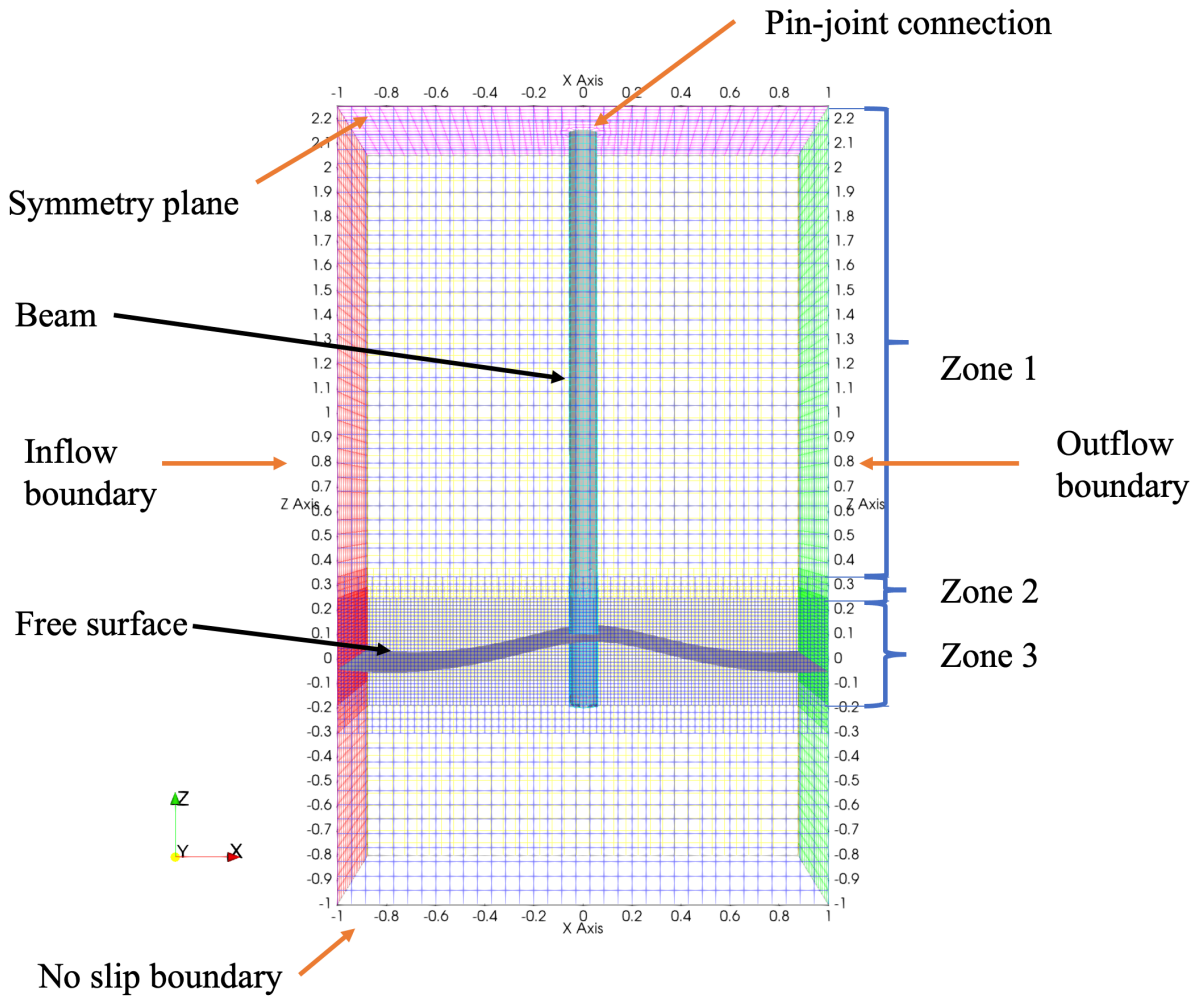


Figure 6.5: A front view of the discretised computational domain is presented. The boundary conditions at different regions of the computational domain are shown with orange-coloured arrows. The three zones based on the cell size are also shown.

In the spatially discretised fluid-free surface, shown in Fig. 6.6, a circular hole can be seen which is the region where the beam comes in contact with the free surface. The boundary of the hole is treated as an impermeable wall which ensures no fluid flow through the beam. The loads exerted by the fluid are also interpolated around the boundary of the circular hole.

The spatial discretisation of the beam is shown in Fig. 6.7. The beam has two types of meshes, i.e. structural mesh and hydro-mesh. The hydro-mesh is in the lower submerged end of the beam and it is used to compute and transfer loads due to water waves and air pressure on the beam. The beam is attached to the top boundary of the computational domain, however, the type of connection is not fixed, i.e. degrees of freedom are not equal to zero. To obtain a similar connection type as with the experimental setup, i.e. pin-joint with rotational spring, the beam allowed rotation in $x-z$ plane and restricted to have translation along $x-$, $y-$, and $z-$axis.

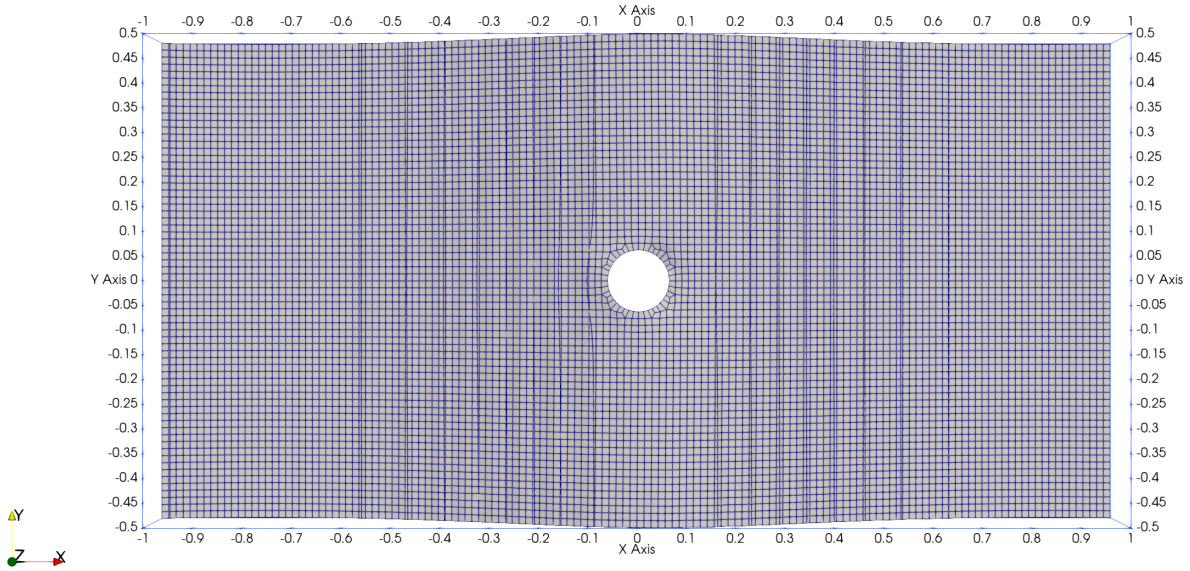


Figure 6.6: Top view of the spatially discretised fluid's free surface.

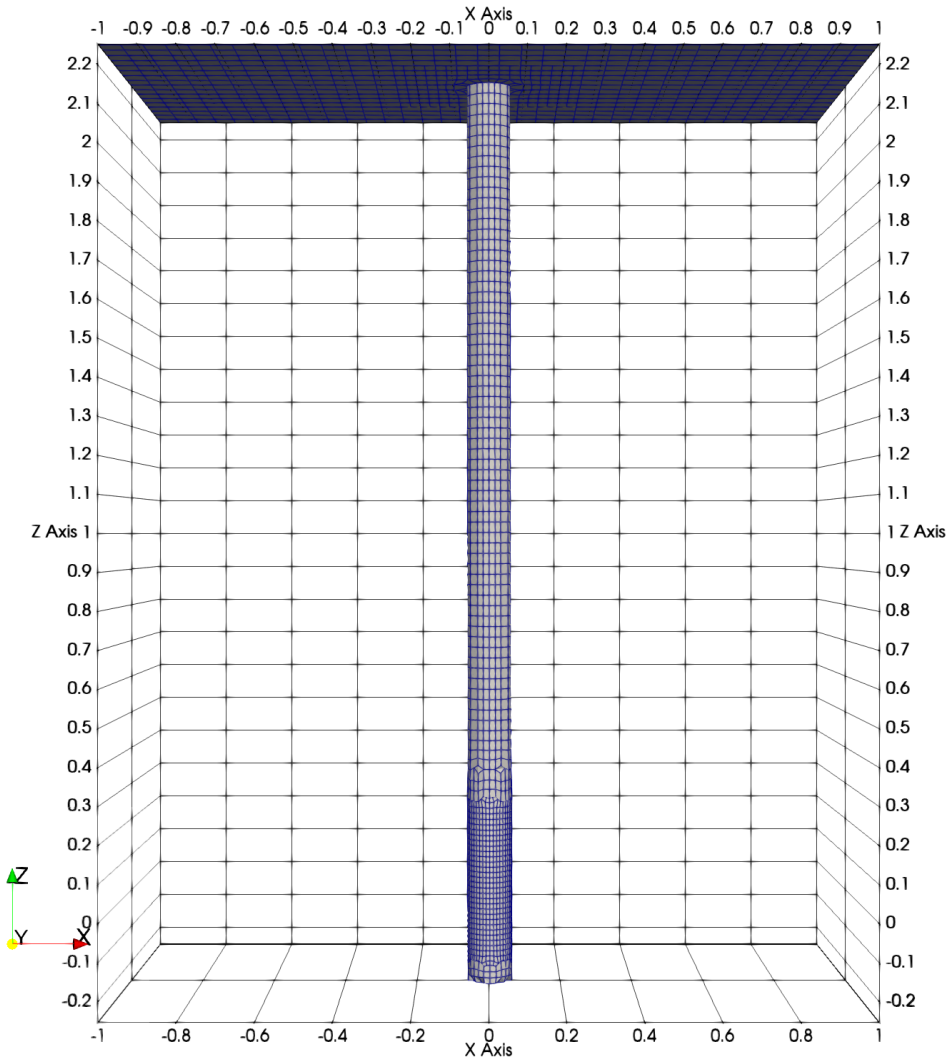


Figure 6.7: Front view of the spatially discretised beam is presented. The submerged part of the beam has hydro-mesh which is finer mesh as compared to the structural mesh.

6.3.2 Grid and time convergence study

After setting up the model, the next step is to perform a grid and time convergence study to ensure that the numerical results are not affected by grid and time-step size. In case of the beam model, the grid convergence study is performed in ANSYS. The model is constructed by using shell elements and modal analysis is performed to compute the first fundamental frequency. The modal analysis is performed by using different sizes of shell elements, this study is shown in Table 6.1 and Fig. 6.8. In addition to the first fundamental frequency value, the other criterion for the grid selection is computational time which is not shown here.

Table 6.1: Mesh convergence study for the FEM model.

	Δx	Δy	f_1
	[mm]	[mm]	[1/s]
1.	25.00	25.00	3.578
2.	16.50	18.70	3.598
3.	12.50	12.50	3.606
4.	6.15	6.25	3.612

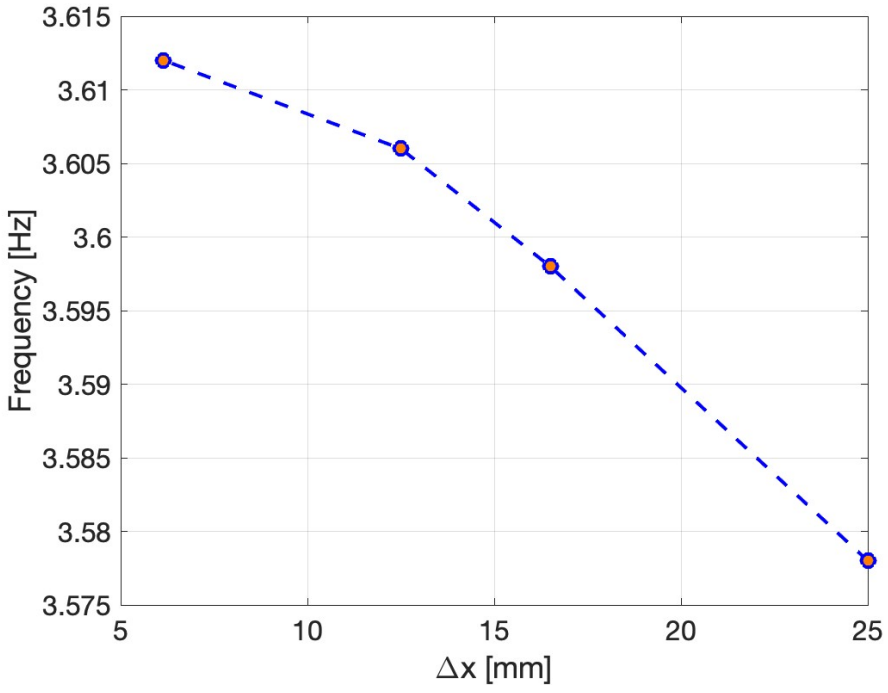


Figure 6.8: Mesh convergence study for the FEM model. The rectangular mesh elements of the beam FEM model are refined and natural frequency is computed.

Based on this study we selected a grid size of 12.5mm. This grid size is optimal for this study because the frequency has a percentage relative error of 0.16% with the experimental value of

frequency and the computational time is also acceptable.

In the case of fluid mesh, we have tested three different grid sizes for zone-1 while zone-3 has a four times smaller grid size than zone-1 grid, i.e. zone-3 has four times more cells than zone-1. The different cell numbers, along $x-$, $y-$ and $z-$axis of the domain, corresponding to each case, i.e. Grid 1, Grid 2, and Grid 3, used for the mesh convergence study are summarized in Table 6.2.

Table 6.2: Mesh convergence study for the FSI model.

	Zone-1			Zone-3		
	N_x	N_y	N_z	N_x	N_y	N_z
Grid 1	25	12	40	100	48	160
Grid 2	35	17	56	140	68	224
Grid 3	50	24	80	200	96	320

Since the FSI problem under consideration is time-dependent, hence, selection of a suitable time step that must satisfy the CFL condition is necessary for the stable simulation. We performed a temporal convergence study by using different time steps (Δt) which are: 0.01, 0.005, and 0.0025.

Table 6.3: Grid convergence study for the ReFRESCO model.

	Δt= 0.01	Δt=0.005	Δt=0.0025
	A_{wave}	A_{wave}	A_{wave}
	[m]	[m]	[m]
Grid 1	0.0162	0.0268	0.03
Grid 2	0.0246	0.0255	0.0298
Grid 3	0.0224	0.0191	0.0301

Table 6.3 shows the spatial and temporal convergence study performed for one of the test cases, i.e. incident wave with amplitude 0.031m and frequency 6.28 rad/s. The wave amplitude values obtained from the ReFRESCO simulations are compared with the experiment. A plot of the convergence study with different grid and time step sizes is shown in Fig. 6.9. Based on this study we found that Grid 2 with time step 0.0025 is most suitable in terms of accuracy and computational time. Hence, we will use Grid 2 with a time step (dt) size of 0.0025 to carry out the numerical simulations which are presented in the next section.

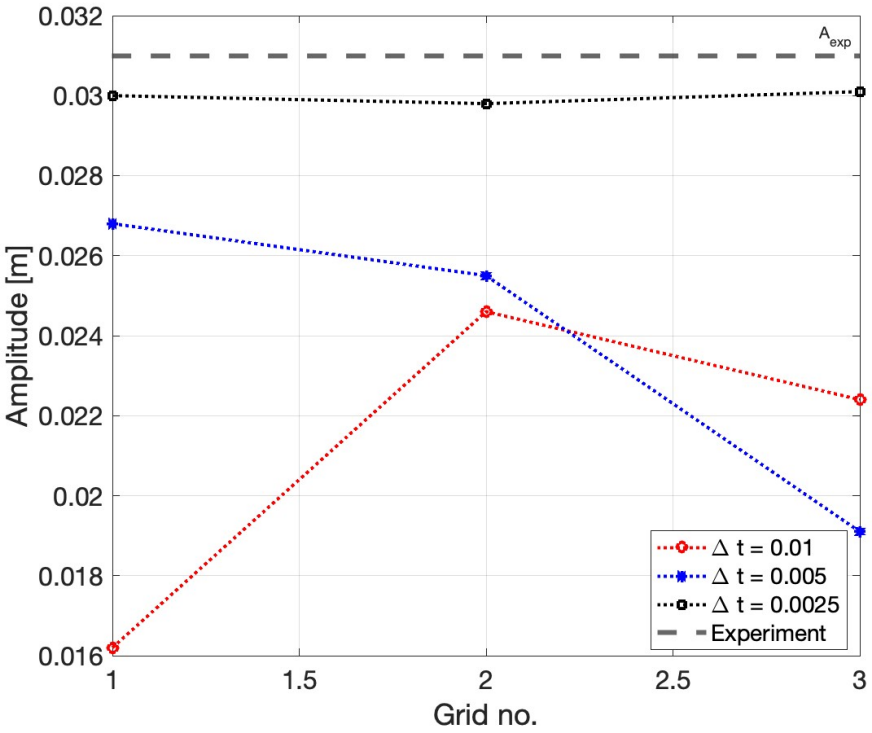


Figure 6.9: Comparison of wave amplitude value obtained from spatial and temporal convergence study with the experimental results obtained from Probe 2 (grey dashed line) is shown.

6.4 Results and Discussion

In this section, we will compare and discuss the results obtained from the numerical solver with the experimental data. Generating waves experimentally utilizing a wavemaker in the wavetank to achieve a wave of a certain wave amplitude and period is called experimental modelling of the waves. It is important to note that the actual wave generated in the wavetank is not always the same as the given input parameters and there could be a deviation. One can notice from the presented study that the wave amplitude measured from the experiments slightly varies from the input wave amplitude. Therefore, the comparison of the numerical results is done with the experimental results and not with the given inputs.

6.4.1 Regular-wave and beam interactions when the beam is submerged at 0.25m.

The actual subcase of the experimental case-1 includes eight test cases based on different wave parameters, i.e. wave amplitude and wave time period. However, this comparison study is performed for selected test cases, which are summarized in Table 6.4. In Table 6.4, the column

heading "Given initials" presents the wave parameters that we intended to achieve through experimental and numerical modelling.

Table 6.4: A summary of the wave parameters used in different test cases of the first subcase, and corresponding results obtained from experimental and numerical modelling of each test case are presented.

	Given initials		Experiments		ReFRESCO		Relative error	
	A_{wave}	ω	A_{wave}	Response	A_{wave}	Response	A_{wave}	Response
	[m]	[rad/s]	[m]	[m/s^2]	[m]	[m/s^2]	[m]	[m/s^2]
1	0.063	6.28	0.062	0.606	0.059	0.550	4.92%	10.18%
2	0.141	4.19	0.120	0.271	0.112	0.270	6.70%	0.37 %
3	0.031	6.28	0.033	0.297	0.030	0.305	8.88%	-2.62%
4	0.070	4.19	0.055	0.120	0.055	0.140	-0.91%	-14.29%
5	0.125	3.14	0.125	0.052	0.121	0.122	2.89 %	-57.20%
6	0.008	10.83	0.007	0.445	0.006	0.365	7.81%	1.37%

The column heading "Experiments" has two subcolumns which present the wave amplitude, measured by Probe 2, and beam response which is measured by an accelerometer located at the submerged end of the beam. The column heading "ReFRESCO" has two subcolumns which present the wave amplitude and beam response obtained from the numerical modelling. The accelerometer measures the beam's accelerations in all three axes, i.e. x-, y- and z-axes. However, we have only considered accelerations along the x-axis because the incoming water waves and the resulting maximum accelerations both are in the x direction. The output signals from experimental modelling and numerical simulations are in the time domain. These time domain signals need to be analysed in the frequency domain by performing a Fourier transform to determine the magnitude and phase of different harmonics. The maximum amplitude of the first harmonic is found for each signal, i.e. wave amplitude and beam's accelerations, corresponding to each test case which are then listed in the Table and plotted in the graphs shown in this section. The relative error obtained by comparing the numerical results with the experiments is presented under the column heading "Relative error". Later in this section, we present the plots of data listed in Table 6.4.

At first, we will compare the wave amplitude predicted by ReFRESCO with the actual experimental results. This comparison is shown in Fig. 6.10.

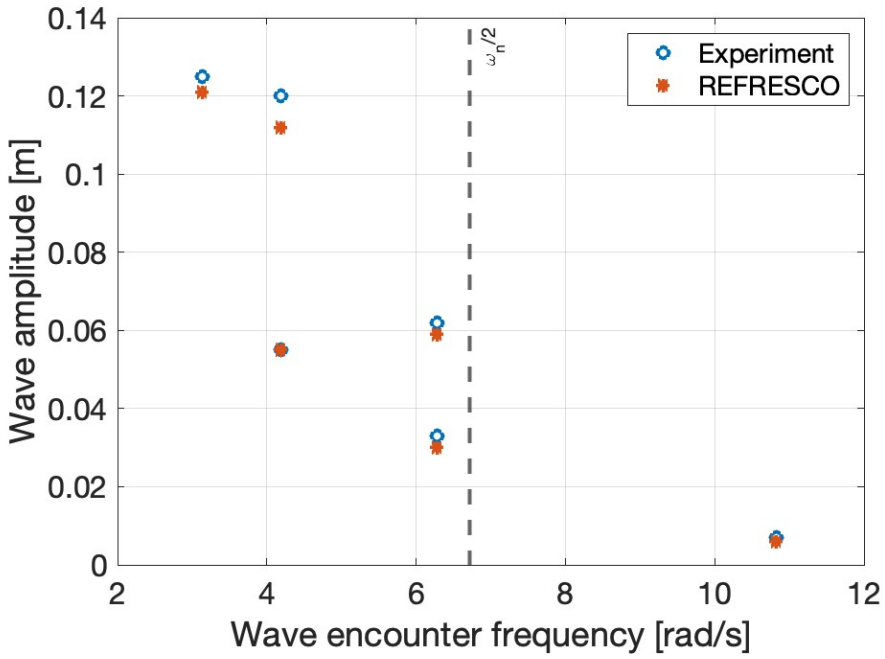


Figure 6.10: A comparison of wave amplitude values obtained from ReFRESCO and the experiments is shown. The experimental results are presented by blue marker while the numerical results are shown by red marker.

After that, we have shown the comparison of the beam response obtained from numerical modelling with the experimental results in Fig. 6.11.

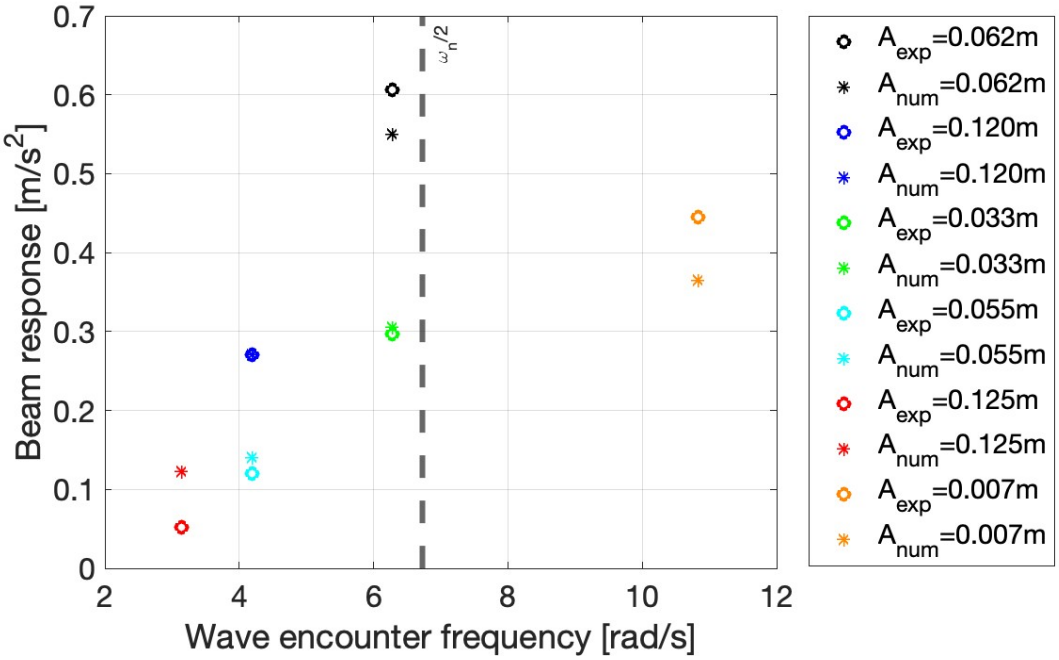
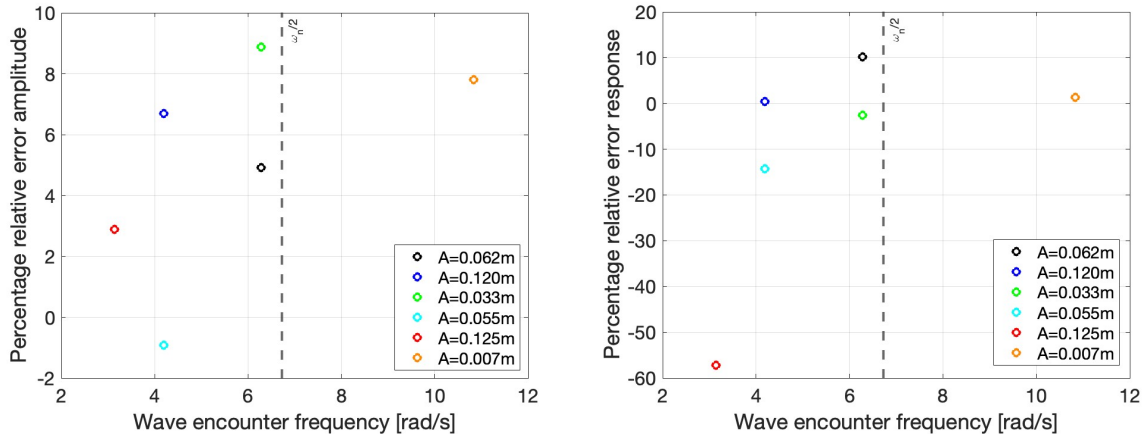


Figure 6.11: Comparison of beam response (accelerations) obtained from ReFRESCO and the experiments is shown.

Finally, the relative errors of wave amplitude and beam response obtained from the comparison of the experimental and numerical results are shown in Fig. 6.12.



(a) Percentage relative error for wave amplitude. (b) Percentage relative error for beam response.

Figure 6.12: Relative error between the experimental and numerical results for the first subcase is presented.

In Fig. 6.12 (a), we can see that the relative error of wave amplitude for all six test cases ranges from 0 to 10%. Fig 6.12(b) shows the relative error of beam response. We can note that the error for five of six test cases ranges from ≈ -15 to 10%. However, there is a case with a large deviation, i.e. relative error of ≈ 60% for the beam response while the relative error for the wave amplitude is 2.89%. To investigate the reason for deviation we perform the harmonic analysis of the numerical results for the incident wave and beam response produced in this case. The time domain signals obtained from the experiments and ReFRESCO are shown in Fig. 6.13 and Fig. 6.14, respectively. To further investigate the amplitude and frequency of the dominant signals we have performed harmonic analysis, i.e. a Fourier transform (Fast Fourier Transform, FFT) on the time-domain signals. The harmonic analysis shown in Fig. 6.15 clarifies the reason for the large deviation in the beam response. It can be seen that the wave is single-frequency whereas the beam's response has multiple frequencies. Furthermore, the high-frequency modes are dominant than the low-frequency modes, therefore, the beam's response is nonlinear. However, the FEM model used in this study is linear and thus incapable of accurately model nonlinear response.

After showing a case with the highest discrepancy in Fig. 6.15, we present a comparison case in Fig. 6.18 when the results from the numerical simulations and experiments agree well.

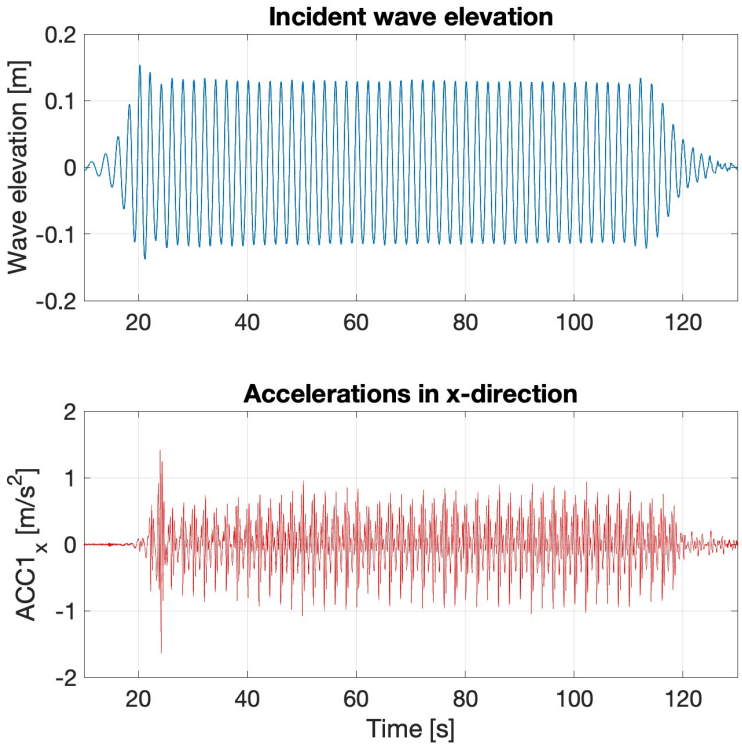


Figure 6.13: Time domain signals of the incident wave (A_{wave} = 0.125 m and ω= 3.14 rad/s) and beam response (accelerations) obtained from the experiments are shown in the frequency domain.

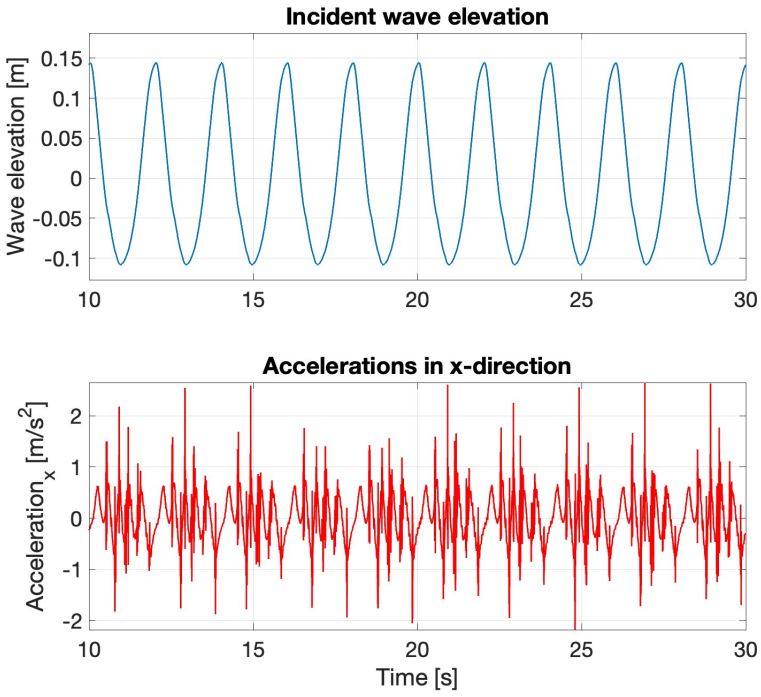


Figure 6.14: The incident wave (A_{wave} = 0.125 m and ω= 3.14 rad/s) and beam response (accelerations) obtained from ReFRESCO are shown in the time domain.

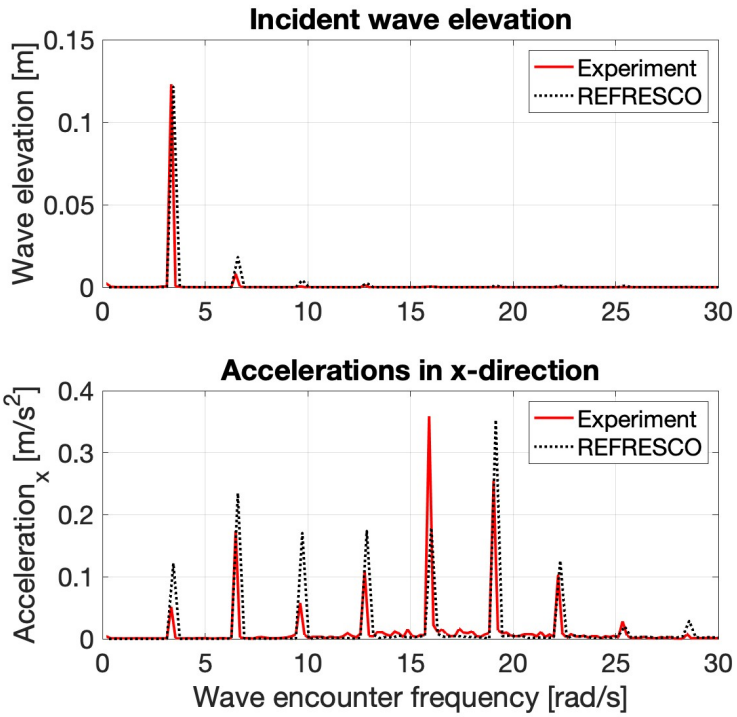


Figure 6.15: Comparison of the incident wave (A_{wave} = 0.125 m and ω= 3.14 rad/s) and beam response (accelerations) obtained from ReFRESCO and the experiments are shown in the frequency domain.

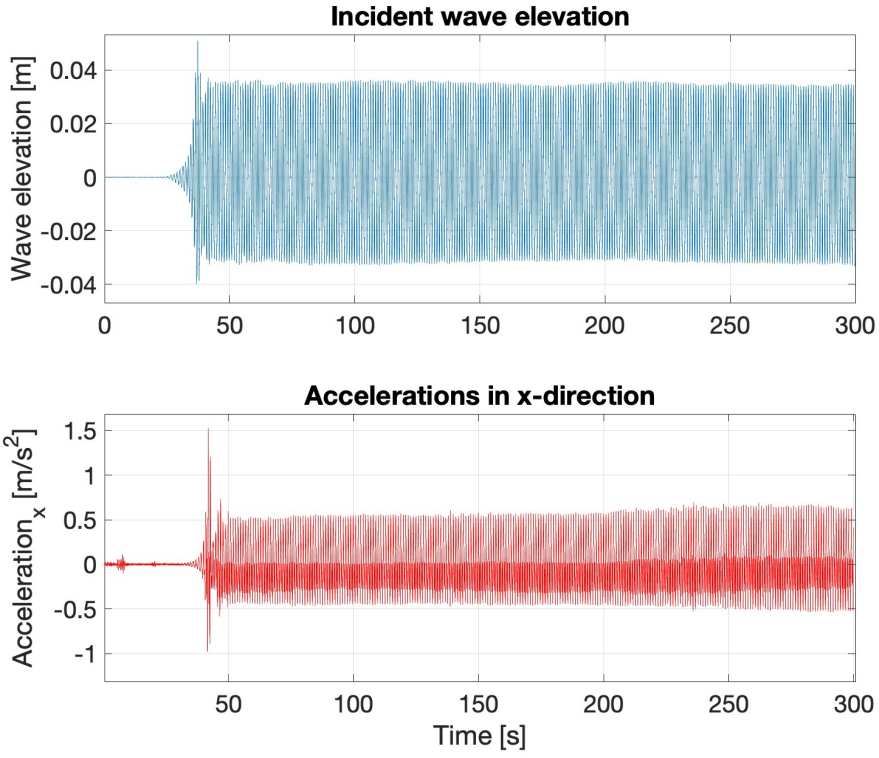


Figure 6.16: The incident wave (A_{wave} = 0.031 m and ω= 6.28 rad/s) and beam response (accelerations) obtained from the experiments are shown in the time domain.

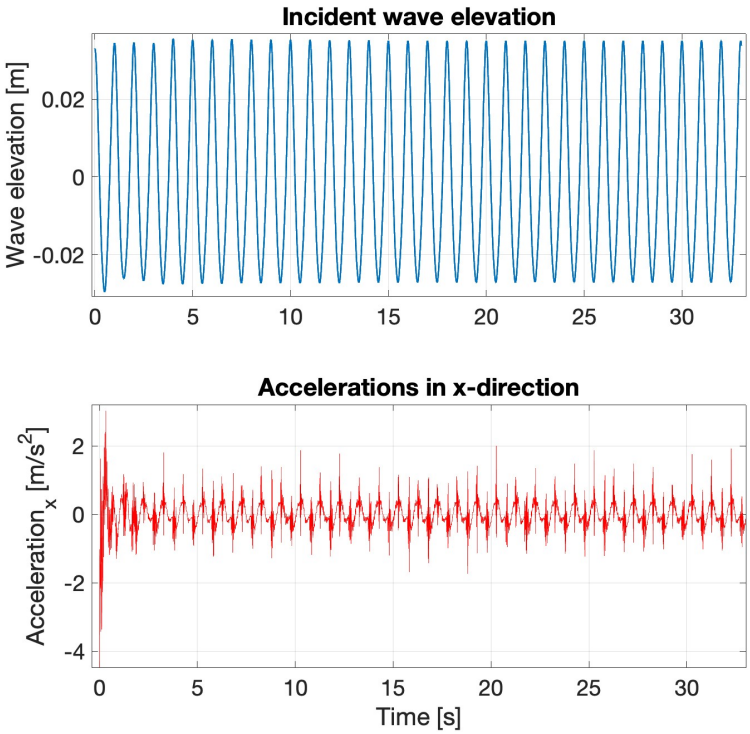


Figure 6.17: The incident wave (A_{wave} = 0.031 m and ω= 6.28 rad/s) and beam response (accelerations) obtained from ReFRESCO are shown in the time domain.

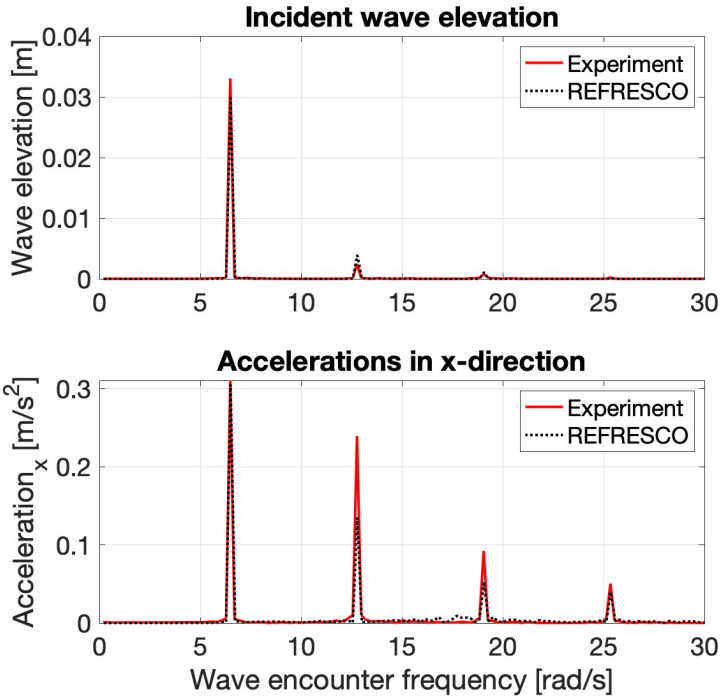


Figure 6.18: Comparison of the incident wave (A_{wave} = 0.031 m and ω= 6.28 rad/s) and beam response (accelerations) obtained from ReFRESCO and the experiments are shown in the frequency domain.

6.4.2 Regular-wave and beam interactions when the beam is submerged at 0.5m.

Similar to the first subcase, instead of simulating all the test cases in the second subcase of experimental case 1, we will simulate some cases. The cases selected for this study are summarised in Table 6.5.

Table 6.5: A summary of the wave parameters used in different test cases of the second subcase, and corresponding results obtained from experimental and numerical modelling of each test case are listed.

	Given Initials		Experiments		ReFRESCO		Relative error	
	A_{wave}	ω	A_{wave}	Response	A_{wave}	Response	A_{wave}	Response
	m	rad/s	m	m/s^2	m	m/s^2	m	m/s^2
1	0.063	6.280	0.061	0.915	0.059	0.793	2.71%	15.38%
2	0.141	4.190	0.113	0.360	0.112	0.430	0.89%	-16.28%
3	0.031	6.280	0.032	0.494	0.030	0.427	7.02%	15.69%
4	0.070	4.190	0.057	0.204	0.056	0.214	1.79%	-4.63%

Fig. 6.19 shows the comparison of wave amplitude values obtained from numerical and experimental modelling.

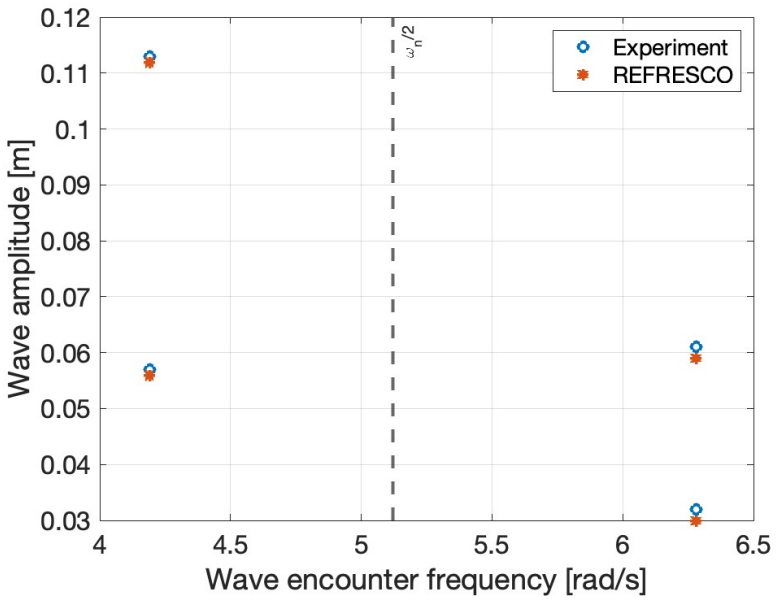


Figure 6.19: Wave amplitude values obtained from the numerical modelling are compared with the experiments. The experimental results are presented by blue marker while the numerical results are shown by red marker.

Fig. 6.20 shows the comparison of the beam response (accelerations) obtained from numerical and experimental modelling.

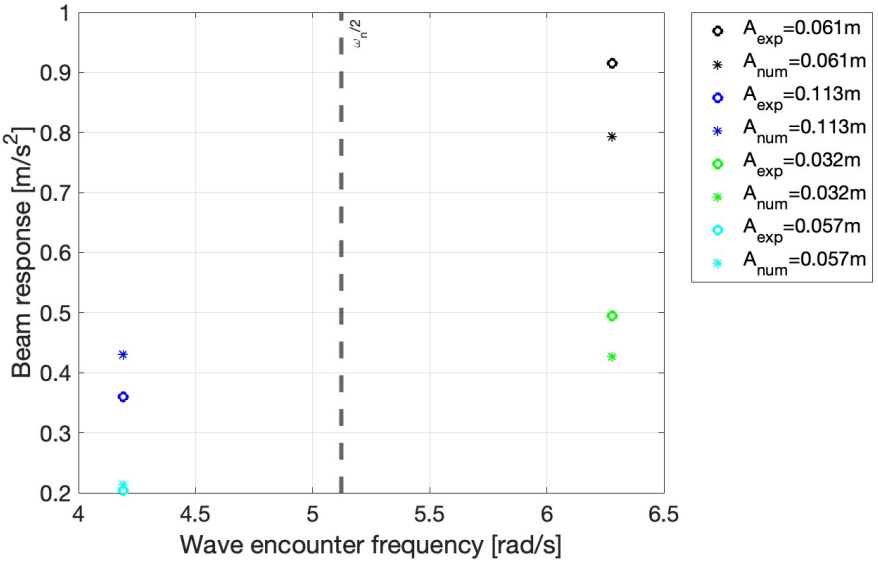
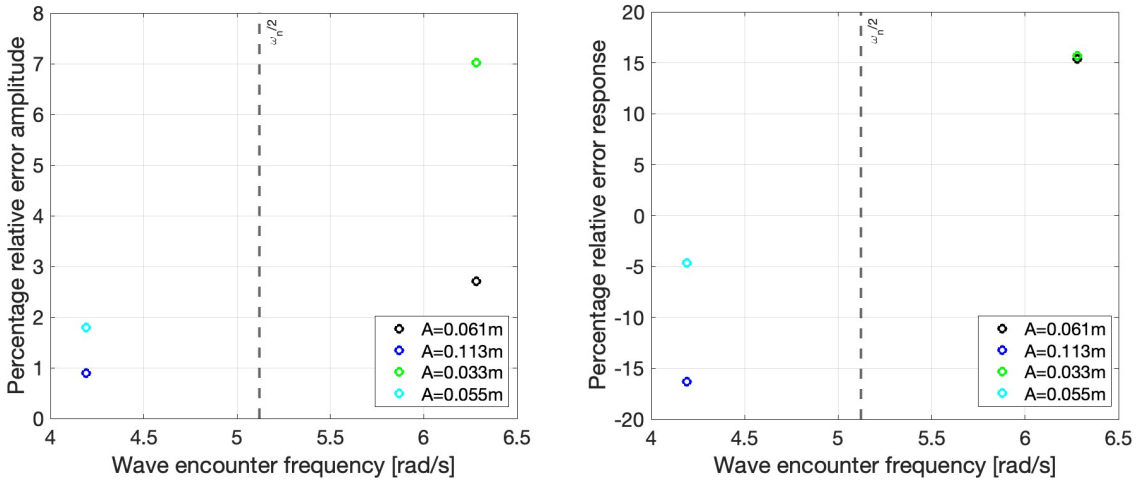


Figure 6.20: Comparison of beam response (accelerations) obtained from the numerical modelling and the experiments is shown.

Finally, the relative errors of wave amplitude and beam response obtained from the comparison of the experimental and numerical results are shown in Fig 6.21.



(a) Percentage relative error for wave amplitude. (b) Percentage relative error for beam response.

Figure 6.21: Relative error between the experimental and numerical results for the first subcase is presented.

In Fig. 6.21(a), we can see that the relative error of wave amplitude for all four test cases ranges from $\approx$ 1 to 7%. Fig. 6.21(b) shows the relative error of beam response. We can note that the error for five of six test cases ranges from $\approx$ -15 to 15%. Next, we select a test case with the highest discrepancy, i.e. A_{wave} = 0.141 m and ω= 4.19 rad/s, and compare the numerical

and experimental results in both the time and frequency domains. The time domain plots are shown in Fig. 6.22 and Fig. 6.23 while the comparison in the frequency domain is shown in Fig. 6.24.

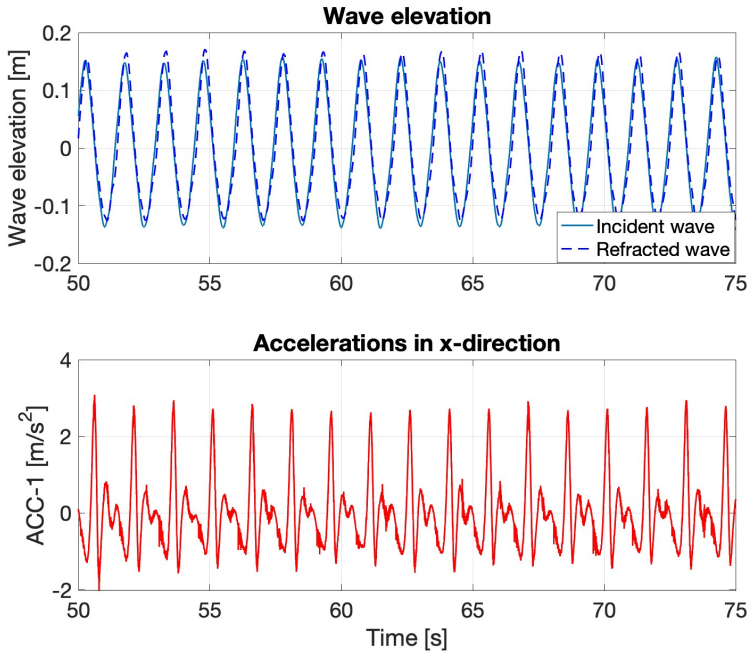


Figure 6.22: The incident wave (A_{wave} = 0.141 m and ω= 4.19 rad/s) and beam response (accelerations) obtained from the experiments are shown in the time domain.

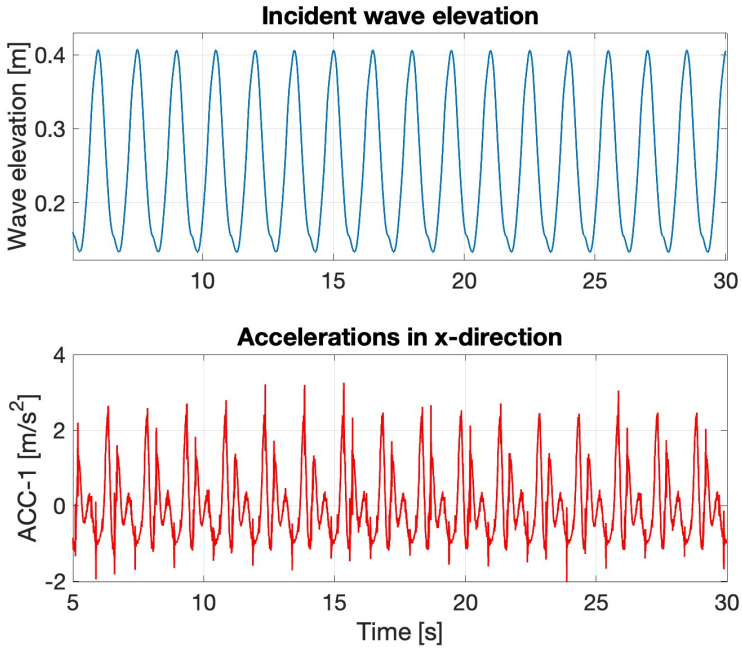


Figure 6.23: The incident wave (A_{wave} = 0.141 m and ω= 4.19 rad/s) and beam response (accelerations) obtained from ReFRESCO are shown in the time domain.

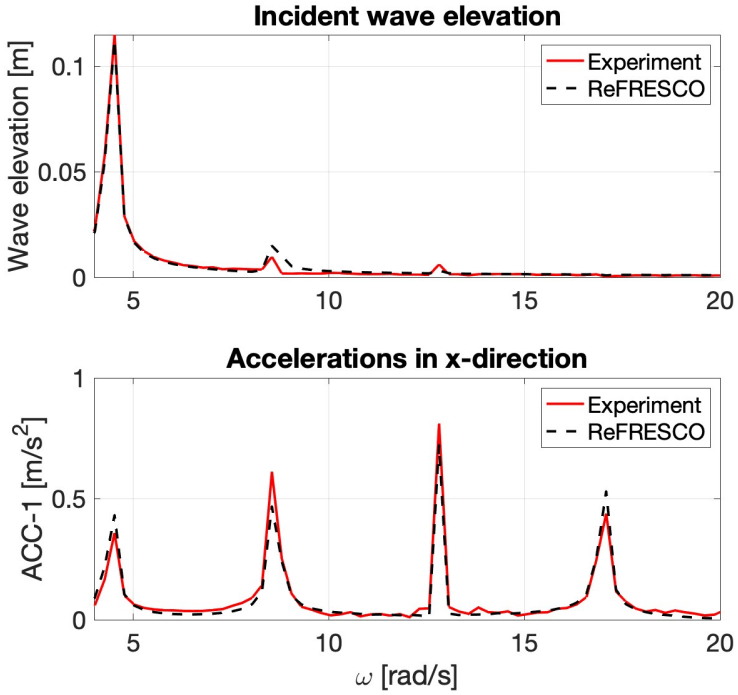


Figure 6.24: Comparison of the incident wave (A_{wave} = 0.141 m and ω= 4.19 rad/s) and beam response (accelerations) obtained from ReFRESCO and the experiments are shown in the frequency domain.

6.5 Nonlinear modelling of irregular waves

This section is dedicated to the explanation of irregular-wave modelling and the beam response to impact loads in ReFRESCO. In addition to regular waves, ReFRESCO is capable of modelling irregular waves. Researchers from MARIN have presented a detailed study in [26] with different examples of extreme waves from the maritime industry to demonstrate ReFRESCO's capabilities. Although ReFRESCO can model extreme irregular waves, due to the limitations of computational cost and time, it is impossible to simulate all events of irregular waves that happened in a 220m long wavetank for the duration of an hour. The challenging aspect of modelling a specific event is that it requires tuning the wave parameters to generate a steep wave of specific height and frequency at a precise location, i.e. at the beam location, thus the impact loads on the beam due to wave breaking can be modelled. Hence, modelling such an event is an iterative process which requires running several simulations after adjusting the parameters. Bunnik et al. [14] have elaborated the iterative process of irregular wave modelling by using ReFRESCO in their study of wave-breaking impact on a flexible fixed-bottom monopile, which is shown in Fig. 6.25.

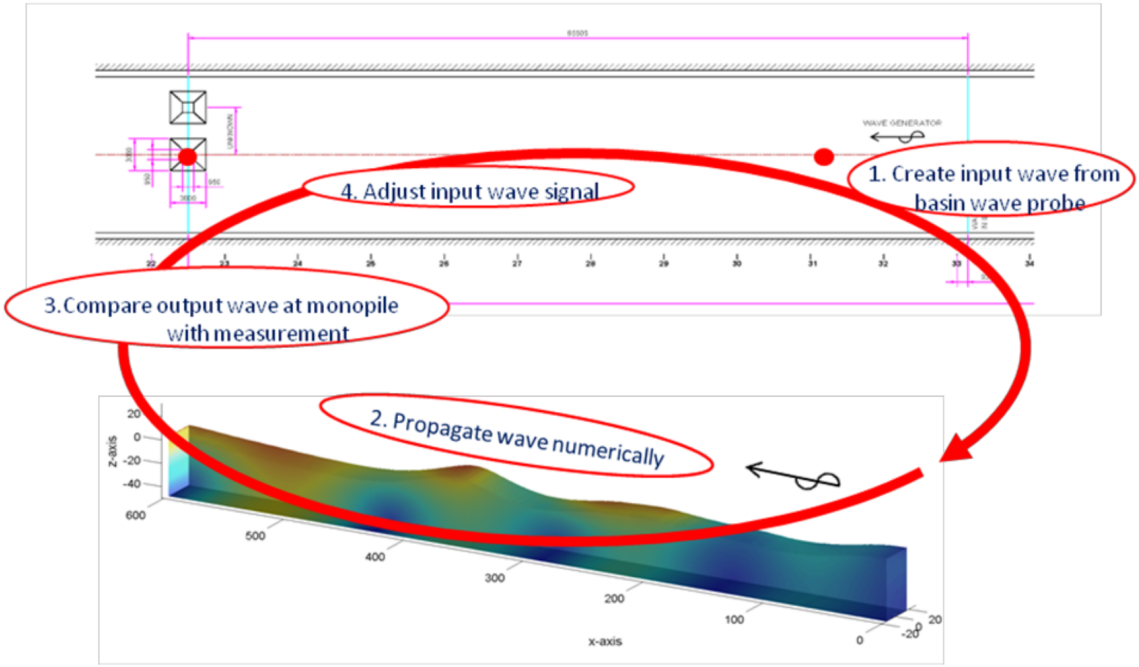


Figure 6.25: The iterative process of focused wave modelling to measure wave-breaking impact and response of flexible fixed-bottom monopile in ReFRESCO is explained [14].

The first step of modelling an irregular wave in ReFRESCO is to select a specific event that occurred during the experimental modelling of irregular waves. In this study, we select an event in which a steep wave breaks at the beam thus resulting in high beam accelerations. Fig. 6.26 shows plots of the incident wave signal and the resulting beam's response that occurred during the experiments. The top plot of Fig. 6.26 shows the wave signal measured by the probe which is located parallel to the beam (Probe 2 in Fig. 6.1)and the bottom plot shows the beam's response to the wave signal.

After the signal is selected, the next step is to run an initial simulation by using the selected wave signal from the experiments as an incoming wave condition at the left-hand side of the computational domain. The computational domain used in this irregular-wave impact on the flexible beam study is the same as the one used for the study of regular-wave impact on the flexible beam. The selection of suitable spatial and temporal sizes is based on the grid and temporal convergence study which is already performed in the regular wave case, therefore, based on our study we have selected a grid size corresponding to Grid 2 given in Table 6.3 and time step 0.0025. The results obtained from the initial run for the wave signal in the time domain are shown in Fig. 6.27.

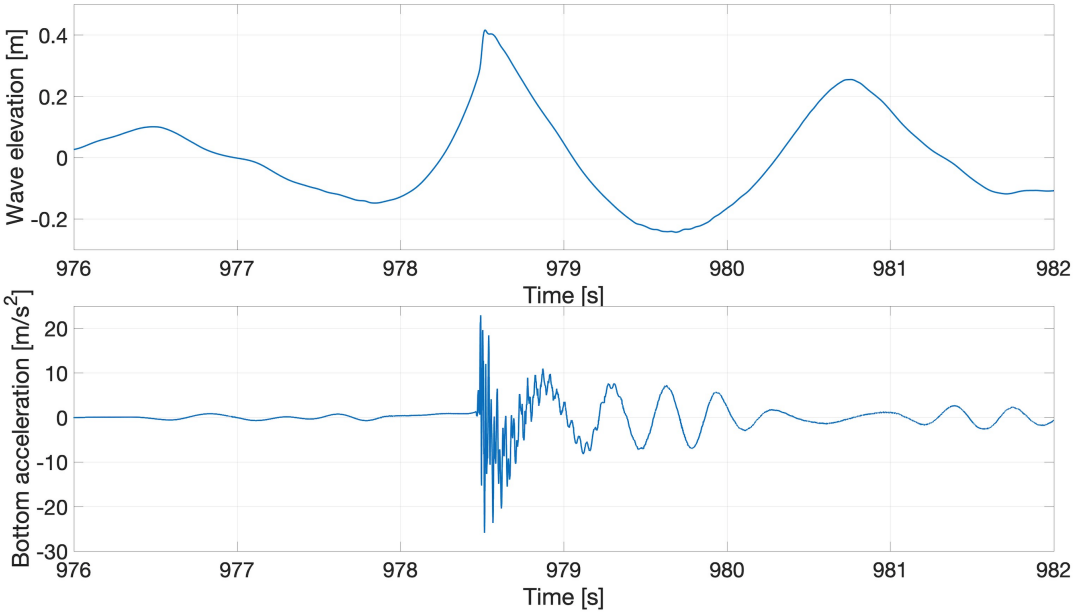


Figure 6.26: A focused wave generated at the beam's location (top plot) and the flexible beam response (bottom plot) to steep irregular waves are shown [69].

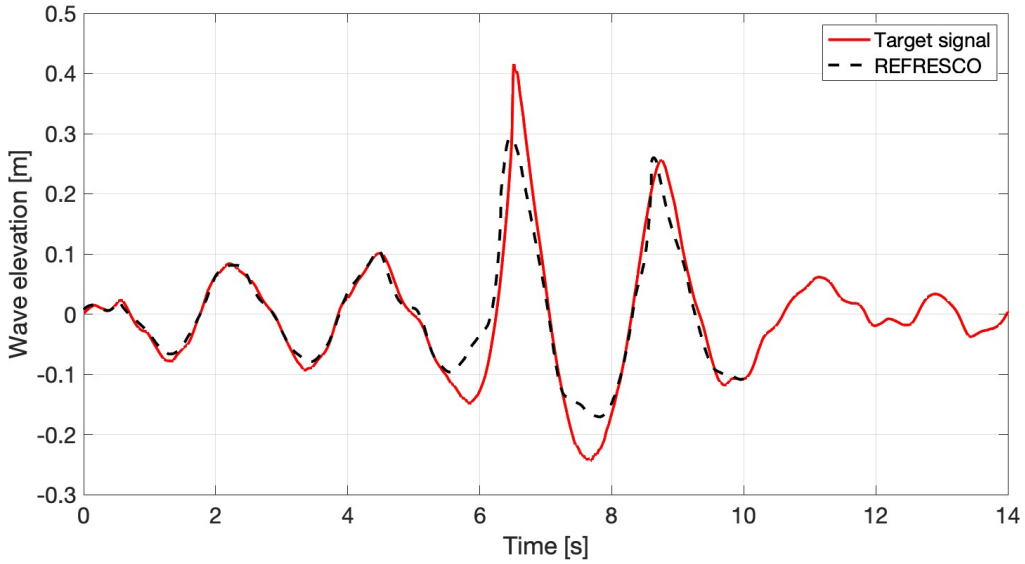


Figure 6.27: The focused wave generated in the experiments (red line) is compared with the wave modelled by ReFRESCO (black dashed line) at the beam's location.

Furthermore, a comparison between the experiments and numerical results is performed in the frequency domain, which is depicted in Fig. 6.28.

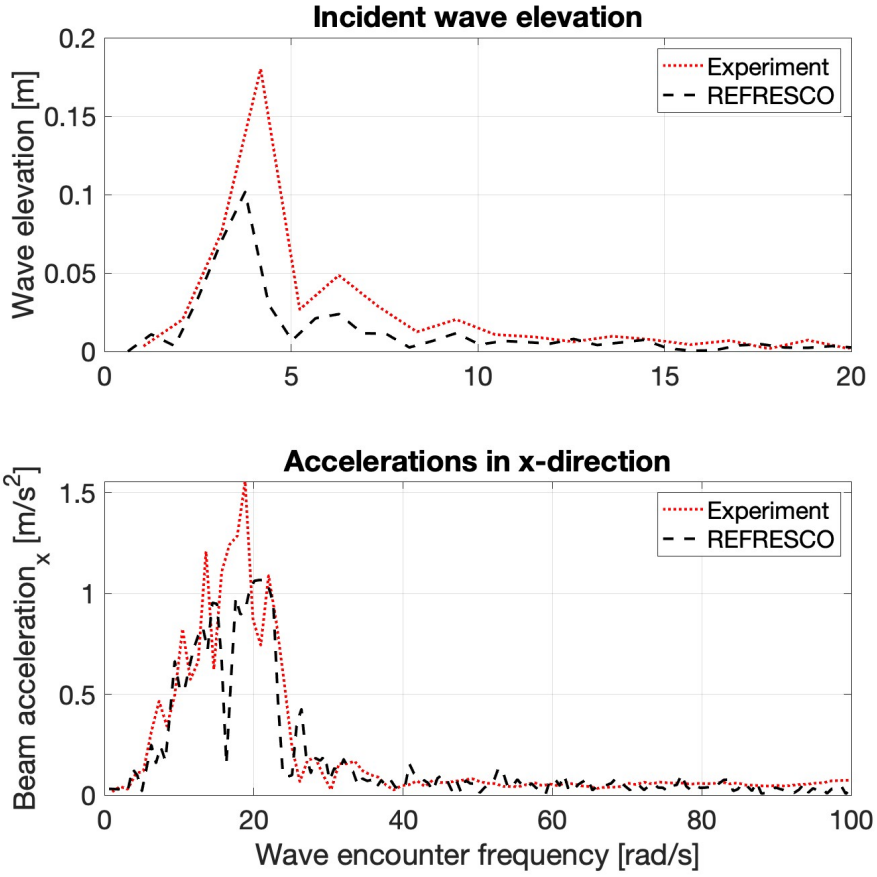


Figure 6.28: Comparison of the experimental and ReFRESCO results is plotted in the frequency domain. The top plot shows the wave signal while the bottom plot depicts the beam response.

We can conclude from the visual analysis of Figs. 6.27 and 6.28 that the wave modelled by ReFRESCO has a large discrepancy from the experimental wave, thus leading to a huge difference in impact load and dynamic response of the beam's submerged end. Therefore, another simulation should be performed with the readjusted wave parameters.

6.6 Conclusion

In this comparison study, we have simulated selected test cases from the first and second subcase of experimental case 1 which is presented in [69] by using a RANSE-based high-fidelity MARIN in-house numerical solver, i.e. ReFRESCO. This study aims to validate ReFRESCO. For both subcases, we found that the wave amplitude value predicted by ReFRESCO simulation has a relative error of $\approx 10\%$ as compared to the experimental value, whereas, in the case of beam response, the relative error was around 10 to 15%. However, we found one test case had a large

relative error because the beam response was nonlinear. Hence, we conclude that this model is limited to the cases when the structure's response is linear, however, nonlinear wave modelling is possible. ReFRESCO's capability to simulate nonlinear waves is tested by performing an initial simulation of a test case from the experimental case 3. Modelling focused waves in ReFRESCO is an iterative process; however, due to time and budget constraints, we could not complete this study and are sharing preliminary results to explain the process of irregular focused wave modelling in ReFRESCO.

Chapter 7

Code Tutorials

7.1 Introduction

This chapter is dedicated to code explanation of the numerical wavetank models that we have developed in Chapter 2 and Chapter 3 of this thesis. This tutorial aims to facilitate the user by explaining the use and possible extension of the code in Firedrake. Furthermore, the codes are also available on an open GitHub repository with public access for code sharing.
To run the codes the user must have Firedrake installed in their systems. The detailed process of Firedrake installation is available on Firedrake website. Furthermore, there is a Firedrake slack channel where users can directly ask the developers if they encounter any problems related to Firedrake. Having shared this information, we now proceed with the code tutorial.

7.2 Shallow water dynamics

This section explains the codes for the piston-driven numerical wavetank model based on the linear and nonlinear shallow water dynamics which are based on the mathematical and numerical models explained in Chapter 2 of this thesis.

7.2.1 Linear Shallow water equations

First, we will share and explain the code for the linear shallow water equations case. This code solves the problem by using the novel approach that we have developed in this thesis, i.e. time-discrete variational principle based on the first-order symplectic-Euler and second-order Störmer-Verlet (SV) time-integration schemes. As this model is the simplest of all numerical

models, therefore, using this case as the staring example for the code tutorial will assist in the understanding process of user.

Define user parameters

Solving a variational problem in Firedrake can be classified into seven steps. Before, explaining the seven-step process, we need to include the relevant libraries in our python script and define the variable. Note that we include Firedrake as a library as well by typing `"import firedrake as fd"`. In this code, the user has the choice to select a case number, as follows:

1. Case 1: Solves linear shallow water case by using time-discrete VP based on symplectic-Euler (SE) scheme.
2. Case 2: Solves linear shallow water case by using time-discrete VP based on Störmer-Verlet (SV) scheme.

After that, there are three choices for the user to select the wavemaker motion type, defined by variable `"start_wavemaker"` as follows:

1. `"start_wavemaker"` = 0. If the user assigns 0 value to `"start_wavemaker"` then the wavemaker will not move and no water waves will be generated. This setting can be used with initial conditions defined as variable `"ic"`=1 to compare the numerical results with the exact solution.
2. `"start_wavemaker"` = 1. If the user assigns 1 value to `"start_wavemaker"` then the wavemaker will keep on moving through the simulation time according to the parameters defined for the wavemaker motion and velocity.
3. `"start_wavemaker"` = 2. If the user assigns 2 to `"start_wavemaker"` then the wavemaker will move at first and then stop after a certain simulation time. This setting assists in monitoring the energy behaviour of the system. The time after which the wavemaker stops can be defined by the user in `"Parameters for wavemaker"` block of the code.

The variable `"ic"` stands for initial condition. Selecting `"ic"`=1 assign the standing wave solution to the variables ϕ and η. The user should select `"ic"`=1 when wavemaker motion is not included, i.e. `"start_wavemaker"` = 0, and aim is to compare the numerical standing-wave solution with the exact standing-wave solution. Finally, the user can compute the wave parameters, i.e wave time period (`Tp`) and angular frequency (`w`), by defining the wave number

(k). This calculation can help to select an appropriate parameters for the wavemaker motion if the user wants to produce standing waves in the numerical wavetank.

```
# Import Python libraries
import firedrake as fd
import math as m
import numpy as np
from matplotlib import animation, pyplot as plt
import os

print('########################################################################')
print('#####################  User input parameters  ##########################')
print('########################################################################')

'''
Case 1 => Solves Linear shallow water case by using time-discrete VP based on
    SE scheme.
Case 2 => Solves Linear shallow water case by using time-discrete VP based on
    SV scheme.
'''

case = 1
start_wavemaker = 2 # (start_wavemaker = 0 => Does not move at all  ,
    start_wavemaker = 1 => wavemaker keeps moving, start_wavemaker = 2 =>
    Wavemaker starts and then stops)
ic = 0                                                  #  ic = 1 to use ics
     = func, ic = 0 use ics as 0
settings = 1                                            # settings for
    wavemaker, 1 == original , 2 == yangs settings
alp = 0
dt = 0.02
print('Time step size =', dt)
save_path =  "data_Lin_SWE"
if not os.path.exists(save_path):
    os.makedirs(save_path)

H0 = 1                                                   # water depth
g = 9.8                                                  # gravitational
    acceleration
c = np.sqrt(g*H0)                                        # wave speed
```

```
31
32 #__________________  FIGURE PARAMETERS  _____________________#
33 tsize = 18 # font size of image title
34 size = 16 # font size of image axes
35 factor = 2
36 t = 0
37 tt = format(t, '.3f')
38
39 ##__________________   Parameters for wave   _____________________##
40 print("#########################################################################")
41 print('#########################   PARAMETERS  of Wave  ####################')
42 print("#########################################################################")
43
44 t = 0 # start time
45
46 m1 = 1
47
48 k1 = (2* fd.pi * m1) /Lx
49 print('Wavenumber in x direction (k1) =',k1)
50
51 w = c * np.sqrt(k1**2)
52 print('wave frequency (w)',w )
53
54 k = np.sqrt(k1**2 )
55 print('Total wavenumber (k) =',k)
56
57 Tp = (2* fd.pi ) /w
58 print('Time period of wave (Tp) =',Tp)
59
60 ##__________________ Parameters for wavemaker _____________________##
61 print("#########################################################################")
62 print('########################## Parameters for wavemaker ##################')
63 print("#########################################################################")
64 gamma = 0.002
65
66 lamb = 70                          # Wavelength
67 print('Wavelength of wavemaker=', lamb)
68
69 kp = 2*fd.pi/lamb                                          # Wave number
```

```
70 print('Wavemaker wave number (kp) =',kp)
71
72 sigma =   c * fd.sqrt(kp**2) #fd.sqrt(g*kp*fd.tanh(kp*H0))  # Wavemaker
       frequency
73 print('Wavemaker frequency (sigma) =', sigma)
74
75 Tw = 2*fd.pi/sigma                  # Wavemaker  period
76 print('Time period of wavemaker (Tw )=', Tw)
77
78 t_end =  2*Tw                                  # time of simulation in sec
79 print('End time =', t_end)
80
81 ts = int(t_end/dt)
82 print('time_steps =', ts)
83 ##_____________ Plot to spot the region of wavemaker frequency ___________##
84
85 lam = np.linspace(1, 200,200)
86 k_plot = 2*fd.pi/lam
87 w_pot =  ( np.sqrt(g* k_plot * np.tanh(k_plot*H0)))
88 w_shallow = c * np.sqrt(k_plot**2)
89 Time_period = 2*fd.pi/w_pot
90
91 plt.title(" Wave frequency ($\omega$) vs. wave number (k)",fontsize=tsize)
92 plt.plot(k_plot, w_pot,  'k--',label = '$Potential$')
93 plt.plot(kp, sigma,  'ro')
94 plt.plot(k_plot, w_shallow , 'r--', label = '$Shallow$')
95 plt.xlabel('k ',fontsize=size)
96 plt.ylabel('$\omega $ ',fontsize=size)
97 plt.xlim([0.05, 3])
98 plt.legend(loc=1)
99 plt.grid()
100
101 ##_____________  Settings to get results at different time steps
       ______________##
102
103 time = []
104 while (t <= t_end):
105         t+= dt
106         time.append(t)
```

```
107
108 x2 = int(len(time)/2)
109 t_plot = np.array([ time[0], time[x2], time[-1] ])
110 print("t_plot =", t_plot)
111 i = 0
112 if ic ==1:
113     color= np.array(['g-', 'b-', 'k-'])
114     colore= np.array(['m--', 'c--', 'r--'])
115 else:
116     color= np.array(['g-', 'b--', 'r:'])
117     colore= np.array(['k:', 'c--', 'm:'])
118
119 t_stop = t_end/2
```

Step 1: define the computational domain and mesh

The first step in the process of implementing a variational problem in Firedrake is to define a computational domain and mesh, i.e. spatially discredited computational domain. Firedrake offers built-in meshing options " *Firedrake also provides a number of built-in mesh types for a number of standard shapes. 1-dimensional intervals may be constructed with IntervalMesh(); 2-dimensional rectangles with RectangleMesh(); and 3-dimensional boxes with BoxMesh(). There are also more specific constructors (for example to build unit square meshes). See utility_meshes for full details.*" Furthermore, the user can use Gmsh, triangle, CGNS, and Exodus to create meshes for complicated geometries because the format of mesh files for the aforementioned software is compatible with Firedrake. In our case, we are using a two-dimensional rectangular domain of $L_x \times L_y$, the user can assign a value to `Lx` and `Ly` to construct a domain of their choice. Similarly, the user can define the number of nodes for spatial discretisation by assigning a value to variable `nx` and `ny`.

```
1 #_____________________ MESH  _____________________#
2
3 nx = 200 # number of nodes in x direction of mesh
4 n = nx
5 ny = 1   # number of nodes in y direction of mesh
6 dx= 1/nx
7 Lx =  140  # Length of domain in x direction
8 Ly = 40    # Length of domain in y direction
```

```
print("Lx =", Lx)
print('Ly =', Ly)
print("Nodes in x direction =", nx)

mesh = fd.RectangleMesh(nx, ny, Lx, Ly)
x,y = fd.SpatialCoordinate(mesh)
Lw =  5                                          # Point till which coordinates
    trandformation will happen
print('Lw =', Lw)

xvals = np.linspace(0, Lx-0.001   , nx)
yvals = np.linspace(0, Ly- 0.001   , ny)
yslice = Ly/2
xslice = Lx/2

wavemaker_id = 1                  # 1 => left side of the domain
```

Step 2: define the function spaces

Now, we define the function space on which we would like to solve the problem. Firedrake offers a range of function spaces that we can choose, for example continuous Galerkin (CG), Dis-continuous Galerkin (DG), and Lagrange etc. More options for function spaces can be found on Firedrake website. In our case, we are using continuous Galerkin (CG).

```
#__________________  Define function spaces  ___________________##

V = fd.FunctionSpace(mesh, "CG", 1)                          # scalar function
    space

phi = fd.Function(V, name = "phi")                           # phi^n
phi_new = fd.Function(V, name = "phi_new")                   # phi^n+1

phi_half = fd.Function(V, name = "phi_half")                          # phi^n

eta_half = fd.Function(V, name = "eta_half")                         # phi^n

eta = fd.Function(V, name =  "eta")                          # eta^n
eta_new = fd.Function(V, name =  "eta_new")                  # eta^n+1

```

```
#_____________________ Wavemaker _____________________#

R = fd.Function(V, name = "wavemaker")                          # Wavemaker motion
Rt = fd.Function(V, name = "wavemaker motion")                  # Wavemaker
    velocity
Rt_half = fd.Function(V, name = "wavemaker motion")                 # Wavemaker
    velocity

#_____________________ Exact solution _____________________#
phie= fd.Function(V, name = "phi_exact")
he = fd.Function(V, name = "h_exact")
etae = fd.Function(V, name = "eta_exact")
```

Step 3: define the test function space

A test function is a function for which we are solving the variational problem, i.e. the unknown of the problem. Therefore, the function space of the test function should be consistent with the function space of the unknown function, which are ϕ and η in this case.

```

#__________________  Define Test functions  __________________##

v = fd.TestFunction(V)
```

Step 4: define initial conditions and assign them to the corresponding function spaces

After defining the appropriate function spaces, we need to assign functions/ initial conditions to our problem. If the user choose to assign initial conditions for the standing wave solution ,i.e. `ic = 1`, then the standing wave solution at `t = 0` will be assigned to the function spaces of relevant to `phi` and `eta`. If `ic = 0`, then 0 will be assigned as initial conditions to `phi` and `eta` and changes due to the wavemaker motion will be observed in the simulation results. The wavemaker motion chosen and controllable parameters used are:

$$R(t) = \begin{cases} \gamma \cos(\sigma t) & 0 \leq t \leq T_p \\ 0 & t > T_p \end{cases}, \tag{7.1}$$

$$\sigma = \sqrt{gH_0}k = \sqrt{gH_0}2\pi/\lambda, \lambda = 70\text{m}, T_p = 2\pi/\sigma, \gamma = 0.002\text{m}.$$

Simulations are undertaken over two time periods $0 \leq t \leq 2T_p$.

```
1
2  ##____________________ Parameters for IC ___________________________##
3  if ic == 1:
4      print('#######################################################################')
5      print('################## Parameters of ICs and Exact ####################')
6      print('#######################################################################')
7
8      A0 = 0.009
9      B0 = 0.009
10
11     print('A0 =', A0)
12     print('B0 =', B0)
13
14     Uo = gamma
15
16     tic = 0
17     aic = np.exp(-1j * sigma * tic)
18     print('aic =', aic)
19
20 ##_______________________  Parameters for Exact Sol _____________________##
21
22     P = (kp * fd.sin(kp * Lx))
23     print('P = (kp * fd.sin(int(kp) * Lx)) =',P)
24
25     U_0 = Uo * 1j * sigma
26
27     a = np.exp(-1j * sigma * t_end)
28     print("Real part of exp(-1j * sigma * t_end) =",a.real)
29             ##########################################
30             #          Assign Initial Conditions       #
31             ##########################################
32
33 print('Assigning initial conditions')
34
35 if ic ==1:
36     ic1 = phi.interpolate( (U_0.real)/P * aic.real* fd.cos(kp * (x - Lx)) \
37         + (g/w) * fd.cos(k1 * x) * ( -A0*fd.sin(w * tic) + B0*fd.cos(w * tic) )
    )
```

```
38      ic2 = eta.interpolate( (((1j*sigma)/ g) * (U_0.real)/(P) * np.exp(-1j *
        sigma * tic)).real * fd.cos(kp * (x - Lx))\
39          +  fd.cos(k1 * x) * ( A0*fd.cos(w * tic) + B0*fd.sin(w * tic) ) )
40
41  else:
42      ic1 = phi.assign (0)
43      ic2 = eta.assign(0)
44
45  phi.assign(ic1)
46  phi_new.assign(ic1)
47  phi_half.assign(ic1)
48
49  eta.assign(ic2)
50  eta_new.assign(ic2)
51  eta_half.assign(ic2)
52
53  phivals = np.array([ic1.at(x, yslice) for x in xvals])
54  etavals = np.array([ic2.at(x, yslice) for x in xvals])
55
56  fig, ((ax1, ax2)) = plt.subplots(2)
57  ax1.set_title('Initial Conditions',fontsize=tsize)
58  ax1.plot(xvals, etavals, label = '$\eta$')
59  ax1.set_ylabel('$\eta (x,t)$ [m] ',fontsize=size)
60
61  ax1.grid()
62
63  ax2.plot(xvals, phivals , label = '$\phi$')
64  ax2.set_xlabel('$x$ [m] ',fontsize=size)
65  ax2.set_ylabel('$\phi (x,t)$ ',fontsize=size)
66  ax2.grid()
67
68              ###############################################
69              #                   Wavemaker                 #
70              ###############################################
71
72  print('################ Wavemaker motion calculations block ##################')
73
74  nt = 0
75  nnt = np.linspace(0, t_end, ts+1)
```

```
76
77 ##__________________  Plot of wavemaker motion  _____________________##
78 print('Plot of wavemaker motion')
79 Rt1=[]
80 Rh1 = []
81
82 if start_wavemaker == 2:
83         print('The wavemaker will stop after time step =',t_stop)
84
85 t = 0
86 for nt in range(len(nnt)):
87     if start_wavemaker  == 1:
88             R_h1 = -gamma*fd.cos(sigma*t)
89             Rt_1 = gamma*sigma*fd.sin(sigma*t)
90
91     elif start_wavemaker == 2:
92
93             R_h1 = -gamma *fd.cos(sigma*t)
94             Rt_1 = gamma *sigma*fd.sin(sigma*t)
95
96             if t >= t_stop:
97                     R_h1 = -gamma*fd.cos(sigma*t_stop)
98                     Rt_1 = 0*gamma*sigma*fd.sin(sigma*t_stop)
99
100    elif start_wavemaker == 0:
101
102            R_h1 = fd.Constant(0)
103            Rt_1 = fd.Constant(0)
104
105    t+=dt
106    Rt1.append(Rt_1)
107    Rh1.append(R_h1)
108
109 if start_wavemaker == 1:
110    Amp_wave = max(Rh1)
111    print('Maximum amplitude of wavemaker =', Amp_wave)
112    vel_wave = max(Rt1)
113    print('Maximum velocity of wavemaker =', vel_wave)
114 else:
```

```
    pass

fig, (ax1, ax2) = plt.subplots(2)

ax1.set_title('Wavemaker Position',fontsize=tsize)
ax1.plot(nnt, Rh1, 'r-', label = f'$h_e: t = {t:.3f}$ ')
ax1.set_ylabel('$R(t)[m]$ ',fontsize=size)
ax1.grid()

ax2.set_title('Wavemaker velocity',fontsize=tsize)
ax2.plot(nnt, Rt1,  'r-', label = f'$\phi_e: t = {t:.3f}$ ')
ax2.set_xlabel('$Time [s]$ ',fontsize=size)
ax2.set_ylabel('$R_{t} [m/s]$ ',fontsize=size)
ax2.grid()
##_________________  FIGURE SETTINGS ___________________________##
print('Figure settings')

plt.figure(2)
fig, (ax1, ax2) = plt.subplots(2)
ax2.set_title(r'$\phi$ value in $x$ direction',fontsize=tsize)
ax1.set_title(r'$\eta$ value in $x$ direction',fontsize=tsize)
ax1.set_ylabel(r'$\eta (x,t) \times 10^{-2} [m]$ ',fontsize=size)
ax1.grid()
ax2.set_xlabel(r'$x [m]$ ',fontsize=size)
ax2.set_ylabel(r'$\phi (x,t)$ ',fontsize=size)
ax2.grid()
```

Step 5: define the variational problem

After assigning the initial conditions we can define the variational problem in Firedrake. Note that Firedrake uses a high-level language, UFL, to describe variational problems. Therefore, the format for writing the problem should be consistent with UFL.

In Case 1, we are solving linear shallow-water equations with piston wavemaker by using time-discrete VP based on SE scheme, given as:

$$0 = \delta \int_0^L \phi^n \frac{(\eta^{n+1} - \eta^n)}{\Delta t} - \phi^{n+1} \frac{\eta^{n+1}}{\Delta t} - \frac{1}{2} H_0 |\nabla \phi^n|^2 - \frac{1}{2} g (\eta^{n+1})^2 \, \mathrm{d}x - H R_t^n \phi^n |_{x=0}, \qquad (7.2)$$

the variations of (7.2) with respect to ϕ^n and η^{n+1} yield the symplectic-Euler time-discrete weak formulations, as follows

$$\int_0^L \delta\phi^n \frac{(\eta^{n+1}-\eta^n)}{\Delta t} - H_0 \boldsymbol{\nabla}\phi^n \cdot \boldsymbol{\nabla}\delta\phi^n \,\mathrm{d}x - HR_t^n \delta\phi^n|_{x=0} = 0 \quad \text{and} \tag{7.3}$$

$$\int_0^L \Big(\frac{(\phi^{n+1}-\phi^n)}{\Delta t} + g\eta^{n+1}\Big)\delta\eta^{n+1}\,\mathrm{d}x = 0. \tag{7.4}$$

In the code, we write the time-discretised VP and automatically derive the time-discretised weak formulations by using `fd.derivative(VP, phi, v)` and `fd.derivative(VP, eta_new, v)`.

```
1
2  ########################  VARIATIONAL PRINCIPLE  ##############################
3  print("#########################################################################")
4  print('########################## Numerical Calculations   ##################')
5  print("#########################################################################")
6
7  t = 0
8
9  if case == 1:
10     print("You have selected case 1 : Linear (alpha = 0) /Nonlinear (alpha = 1)
       SWE VP solved by firedrake by using fd.derivative ")
11     print("Linear shallow water solved by using time-discrete VP based on SE
       scheme")
12
13     E1_t = []
14     E1_p = []
15     E1_k = []
16
17     VP = ( fd.inner ((eta_new - eta)/dt , phi) - fd.inner(phi_new , (eta_new/dt
       )) \
18           - (1/2 * (H0 + alp*eta_new) * fd.inner(fd.grad(phi), fd.grad(phi))) \
19           - (1/2 * g * fd.inner(eta_new,eta_new)) ) * fd.dx - (H0 + alp*eta_new
       )* Rt *phi*fd.ds(1)
20
21
22     eta_expr = fd.derivative(VP, phi, v)  # derivative of VP wrt phi^n to get
       the expression for eta^n+1 first
23     eta_expr = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(
```

```
    eta_expr, eta_new))
24
25
26      phi_expr = fd.derivative(VP, eta_new, v)  # derivative of VP wrt eta^n+1 to
        get the value of phi^n+1
27      phi_expr = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(
        phi_expr, phi_new))
28
29      ##__________ Define .PVD OUTPUT FILES _____________________##
30      if start_wavemaker ==1:
31          outfile_phi = fd.File("results_LinSWE_SE_wm1_case1/phi.pvd")
32          outfile_eta = fd.File("results_LinSWE_SE_wm1_case1/eta.pvd")
33      elif start_wavemaker == 2:
34              outfile_phi = fd.File("results_LinSWE_SE_wm2_case1/phi.pvd")
35              outfile_eta = fd.File("results_LinSWE_SE_wm2_case1/eta.pvd")
36      elif start_wavemaker == 0:
37              outfile_phi = fd.File("results_LinSWE_SE_wm0_case1/phi.pvd")
38              outfile_eta = fd.File("results_LinSWE_SE_wm0_case1/eta.pvd")
39
40      ###_________________ Define .TXT OUTPUT FILES ________________###
41      if start_wavemaker == 1:
42          filename1 = "Linear_SWE_SE_wm1.txt"
43      elif start_wavemaker == 2:
44          filename1 = "Linear_SWE_SE_wm2.txt"
45      elif start_wavemaker == 0:
46          filename1 = "Linear_SWE_SE_wm0.txt"
47
48      f = open(filename1 , 'w+')
49
50
```

Step 6: Solve the variational problem

We define a time loop in which we solve the variational problem by storing the solution at time n and updating the variable values at to find values at time $n+1$. The time step should be chosen according to the CFL condition.

```
1
2       ## _________________  TIME LOOP ________________ ##
3
```

```
    while (t <= t_end):
        tt = format(t, '.3f')
    ## ______________________  wavemaker motion  ________________________ ##
    # # wavemaker moving from t = 0 to t = t_end
        if start_wavemaker == 1:
                        R.assign(-gamma  * fd.cos(sigma*t))
                        Rt.assign( gamma * sigma * fd.sin(sigma*t))

            # # wavemaker moves at first and then stops after some time
        if start_wavemaker == 2:
                        R.assign(-gamma * fd.cos(sigma*t))
                        Rt.assign( gamma * sigma * fd.sin(sigma*t))


                        if t >= t_stop:
                            R.assign(-gamma *fd.cos(sigma*t_stop))
                            Rt.assign(0)

             # # wavemaker does not move at all
        elif start_wavemaker == 0:
                        Rt.assign(0)
                        R.assign(0)
    ## ____________________________________________________________________ ##

        eta_expr.solve()
        phi_expr.solve()

        t+= dt
        # print('velocity of wavemaker',Rt.dat.data) #Rt.dat.data

        Epp = fd.assemble(( 1/2 * g * fd.inner(eta,eta) )* fd.dx)
        Ekk = fd.assemble(0.5 * H0* (fd.grad(phi)**2 * fd.dx))
        Et = abs(Ekk) + abs(Epp)
```

Step 7: Plot and output the results

Finally, within the time loop, we compute and write the value of variables in *.txt* files and generate *.pvd* files for visualisation.

```

```

```
        f.write('%-25s %-25s %-25s %-25s %-25s %-25s %-25s %-25s\n' \
                % (str(t), str(R.dat.data[2]), str(Rt.dat.data[2]), str(phi.at(0,0)), str(eta.at(0,0)), str(Epp), str(Ekk), str(Et) ) )

        if (t in t_plot):
            print('Plotting starts')
            print('t =', t)
            i += 1


            if ic == 1:
                phi_exact = phie.interpolate( (U_0.real)/P * a.real* fd.cos(kp * (x - Lx)) \
                                    + (g/w) * fd.cos(k1 * x ) * ( -A0*fd.sin(w * t) + B0*fd.cos(w * t) ) )

                eta_exact = etae.interpolate( (((1j*sigma)/ g) * (U_0.real)/(P) * np.exp(-1j * sigma * t_end)).real * fd.cos(kp * (x - Lx))\
                                    +  fd.cos(k1 * x ) * ( A0*fd.cos(w * t) + B0*fd.sin(w * t) ) )

                phievals = np.array([phi_exact.at(x, yslice) for x in xvals])
                etaevals = np.array([eta_exact.at(x, yslice) for x in xvals])
            else:
                pass

            eta1vals = np.array([eta_new.at(x, Ly/2) for x in xvals])
            phi1vals = np.array([phi_new.at(x, Ly/2) for x in xvals])


            if start_wavemaker == 1:
                eta_file_name = 'eta_lswe_SE_wm1_'+tt+'.txt'
                phi1_file_name = 'phi_lswe_SE_wm1_'+tt+'.txt'
            elif start_wavemaker == 2:
                 eta_file_name = 'eta_lswe_SE_wm2_'+tt+'.txt'
                 phi1_file_name = 'phi_lswe_SE_wm2_'+tt+'.txt'
            elif start_wavemaker == 0:
                 eta_file_name = 'eta_lswe_SE_wm0_'+tt+'.txt'
                 phi1_file_name = 'phi_lswe_SE_wm0_'+tt+'.txt'
```

```
36
37          eta_file = open(os.path.join(save_path, eta_file_name), 'w')
38          phi1_file = open(os.path.join(save_path, phi1_file_name), 'w')
39
40          y_slice = Ly/2
41          x_coarse = np.linspace(0,Lx-0.001,200)
42          for ix in x_coarse:
43              eta_file.write('%-25s  %-25s %-25s\n' %(str(ix), str(H0 + eta.
    at(ix,y_slice)), str(eta.at(ix,y_slice))))
44              phi1_file.write('%-25s %-25s\n' %(str(ix), str(phi.at(ix,
    y_slice))))
45
46
47          ax1.plot(xvals, eta1vals * (10 ** factor), color[i-1],label = f' $\
    eta_n: t = {t:.3f}$')
48          ax2.plot(xvals,phi1vals, color[i-1], label = f' $\phi_n: t = {t:.3f
    }$')
49
50
51          if ic == 1:
52              ax1.plot(xvals, etaevals* (10 ** factor), colore[i-1], label =
    f'$h_e: t = {t:.3f}$ ')
53              ax2.plot(xvals, phievals,  colore[i-1], label = f'$\phi_e: t =
    {t:.3f}$ ')
54          else:
55              pass
56          ax1.legend(loc=4)
57          ax2.legend(loc=4)
58
59      outfile_eta.write( eta_new )
60      outfile_phi.write( phi_new )
61
62      E1_t.append(Et)
63      E1_k.append(Ekk)
64      E1_p.append(Epp)
65
66      phi.assign(phi_new)
67      eta.assign(eta_new)
68
```

```
    f.close()
    eta_file.close()
    phi1_file.close()

    fig, (ax1, ax2, ax3) = plt.subplots(3)
    fig.suptitle('Energy evolution with time',fontsize= tsize)
    ax1.plot(time, E1_k)
    ax1.set_ylabel('Kinetic energy[J] ',fontsize=size)
    ax1.grid()

    ax2.plot(time, E1_p)
    ax2.set_ylabel('Potential Energy [J]',fontsize=size)
    ax2.grid()

    ax3.plot(time, E1_t)
    ax3.set_xlabel('$Time [s]$ ',fontsize=size)
    ax3.set_ylabel('Total energy [J] ',fontsize=size)
    ax3.grid()
```

Similarly, the process is repeated from steps 5 to 7 for solving the linear shallow water case by using time-discrete VP based on the Störmer-Verlet (SV) scheme. The time-discrete variational principle (VP) based on the Stormer Verlet scheme for the linear potential-flow shallow-water dynamics reads

$$\begin{aligned} 0 =& \delta \int_0^L \Big(-2\Big(\frac{\phi^{n+1/2}-\phi^n}{\Delta t}\Big)\eta^n - 2\Big(\frac{\phi^{n+1}-\phi^{n+1/2}}{\Delta t}\Big)\eta^{n+1} \\ & - \frac{1}{2}g\Big((\eta^n)^2 + (\eta^{n+1})^2\Big) - H|\boldsymbol{\nabla}\phi^{n+1/2}|^2\Big)\,\mathrm{d}x \\ & - 2HR_t^{n+1/2}\phi^{n+1/2}\Big|_{x=0}. \end{aligned} \tag{7.5}$$

The variations of (7.5) with respect to $\delta\eta^n$, $\delta\phi^{n+1/2}$ and $\delta\eta^{n+1}$ yield

$$\int_0^L -2\Big(\frac{\phi^{n+1/2}-\phi^n}{\Delta t}\Big)\delta\eta^n - g\eta^n\delta\eta^n\,\mathrm{d}x = 0, \tag{7.6}$$

$$\int_0^L 2\Big(\frac{\eta^{n+1}-\eta^n}{\Delta t}\Big)\delta\phi^{n+1/2} - 2H\boldsymbol{\nabla}\phi^{n+1/2}\boldsymbol{\nabla}\delta\phi^{n+1/2}\,\mathrm{d}x - 2HR_t^{n+1/2}\delta\phi^{n+1/2}|_{x=0}, \tag{7.7}$$

$$\int_0^L -2\Big(\frac{\phi^{n+1}-\phi^{n+1/2}}{\Delta t}\Big)\delta\eta^{n+1} - g\eta^{n+1}\delta\eta^{n+1}\,\mathrm{d}x = 0, \tag{7.8}$$

respectively. We directly implement the time-discrete variational principle in Firedrake and

automatically derive the weak formulations.

```
1
2 elif case ==2:
3     print('Case 2: The linear SWE will be solved with time discrete VP based on
      SV scheme.')
4
5     E1_t = []
6     E1_p = []
7     E1_k = []
8
9     VP =( -fd.inner (2 * (phi_half - phi)/dt     , eta ) \
10          - fd.inner(2 * (phi_new - phi_half)/dt , eta_new ) \
11          - (1/2 * (H0) * fd.inner(fd.grad(phi_half), fd.grad(phi_half)) \
12            + 1/2 * (H0) * fd.inner(fd.grad(phi_half), fd.grad(phi_half))) \
13          - (1/2 * g * (fd.inner(eta_new,eta_new)  + fd.inner(eta,eta) )  ) )*
    fd.dx\
14          - (H0 * Rt_half *phi_half + H0 * Rt_half *phi_half ) *fd.ds(1)
15
16
17    phi_half_expr = fd.derivative(VP, eta, v)  # derivative of VP wrt eta^n+1
    to get the value of phi^n+1
18    phi_half_expr = fd.NonlinearVariationalSolver(fd.
    NonlinearVariationalProblem(phi_half_expr, phi_half))
19
20    eta_expr = fd.derivative(VP, phi_half, v)  # derivative of VP wrt phi^n to
    get the expression for eta^n+1 first
21    eta_expr = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(
    eta_expr, eta_new))
22
23
24    phi_expr = fd.derivative(VP, eta_new, v)  # derivative of VP wrt eta^n+1 to
     get the value of phi^n+1
25    phi_expr = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(
    phi_expr, phi_new))
26
27
28    ##__________ OUTPUT FILES ______________________##
29    if start_wavemaker ==1:
30        outfile_phi = fd.File("results_LinSWE_SV_wm1_case1/phi.pvd")
```

```
31          outfile_eta = fd.File("results_LinSWE_SV_wm1_case1/eta.pvd")
32      elif start_wavemaker == 2:
33              outfile_phi = fd.File("results_LinSWE_SV_wm2_case1/phi.pvd")
34              outfile_eta = fd.File("results_LinSWE_SV_wm2_case1/eta.pvd")
35      elif start_wavemaker == 0:
36              outfile_phi = fd.File("results_LinSWE_SV_wm0_case1/phi.pvd")
37              outfile_eta = fd.File("results_LinSWE_SV_wm0_case1/eta.pvd")
38
39      ###_________________  TXT FILES ______________###
40      if start_wavemaker == 1:
41          filename1 = "Linear_SWE_SV_wm1.txt"
42      elif start_wavemaker == 2:
43          filename1 = "Linear_SWE_SV_wm2.txt"
44      elif start_wavemaker == 0:
45          filename1 = "Linear_SWE_SV_wm0.txt"
46
47      exact_sol = "exact_sol.txt"
48
49      f = open(filename1 , 'w+')
50      e = open(exact_sol, 'w+')
51
52
53      ## _________________  TIME LOOP _________________ ##
54
55      while (t <= t_end):
56          tt = format(t, '.3f')
57          t_half = t + dt/2
58      ## ______________________  wavemaker motion  ________________________ ##
59      # # wavemaker moving from t = 0 to t = t_end
60          if start_wavemaker == 1:
61                          R.assign(-gamma  * fd.cos(sigma*t))
62                          Rt.assign( gamma * sigma * fd.sin(sigma*t))
63                          Rt_half.assign( gamma * sigma * fd.sin(sigma*t_half))
64
65              # # wavemaker moves at first and then stops after some time
66          if start_wavemaker == 2:
67                          R.assign(-gamma * fd.cos(sigma*t))
68                          Rt.assign( gamma * sigma * fd.sin(sigma*t))
69                          Rt_half.assign( gamma * sigma * fd.sin(sigma*t_half))
```

```
70
71                      if t >= t_stop:
72                          R.assign(-gamma *fd.cos(sigma*t_stop))
73                          Rt_half.assign( gamma * sigma * fd.sin(sigma *
    t_stop))
74                          Rt.assign(0)
75
76              # # wavemaker does not move at all
77          elif start_wavemaker == 0:
78                      Rt.assign(0)
79                      R.assign(0)
80                      Rt_half.assign(0)
81      ## ------------------------------------------------------------------- ##
82          phi_half_expr.solve()
83          eta_expr.solve()
84          phi_expr.solve()
85
86
87          t+= dt
88          # print('velocity of wavemaker',Rt.dat.data) #Rt.dat.data
89
90          Epp = fd.assemble(( 1/2 * g * fd.inner(eta,eta) )* fd.dx)
91          Ekk = fd.assemble(0.5 * H0* (fd.grad(phi)**2 * fd.dx))
92          Et = abs(Ekk) + abs(Epp)
93
94          f.write('%-25s %-25s %-25s %-25s %-25s %-25s %-25s %-25s\n' \
95                  % (str(t), str(R.dat.data[2]), str(Rt.dat.data[2]), str(phi.at
    (0,0)), str(eta.at(0,0)), str(Epp), str(Ekk), str(Et) ) )
96
97          if (t in t_plot):
98              print('Plotting starts')
99              print('t =', t)
100             i += 1
101
102             if ic == 1:
103                 phi_exact = phie.interpolate( (U_0.real)/P * a.real* fd.cos(kp
    * (x - Lx)) \
104                                     + (g/w) * fd.cos(k1 * x ) * ( -A0*fd.sin(w *
    t) + B0*fd.cos(w * t) ) )
```

```
105
106             eta_exact = etae.interpolate( (((1j*sigma)/ g) * (U_0.real)/(P)
    * np.exp(-1j * sigma * t_end)).real * fd.cos(kp * (x - Lx))\
107                                 +  fd.cos(k1 * x ) * ( A0*fd.cos(w * t) + B0*
    fd.sin(w * t) ) )
108
109             phievals = np.array([phi_exact.at(x, yslice) for x in xvals])
110             etaevals = np.array([eta_exact.at(x, yslice) for x in xvals])
111         else:
112             pass
113
114         eta1vals = np.array([eta_new.at(x, Ly/2) for x in xvals])
115         phi1vals = np.array([phi_new.at(x, Ly/2) for x in xvals])
116
117
118         if start_wavemaker == 1:
119             eta_file_name = 'eta_lswe_SV_wm1_'+tt+'.txt'
120             phi1_file_name = 'phi_lswe_SV_wm1_'+tt+'.txt'
121         elif start_wavemaker == 2:
122              eta_file_name = 'eta_lswe_SV_wm2_'+tt+'.txt'
123              phi1_file_name = 'phi_lswe_SV_wm2_'+tt+'.txt'
124         elif start_wavemaker == 0:
125              eta_file_name = 'eta_lswe_SV_wm0_'+tt+'.txt'
126              phi1_file_name = 'phi_lswe_SV_wm0_'+tt+'.txt'
127         if ic == 1:
128             etae_file_name = 'etae_lswe_SV_wm1_'+tt+'.txt'
129             phi1e_file_name = 'phie_lswe_SV_wm1_'+tt+'.txt'
130
131
132         eta_file = open(os.path.join(save_path, eta_file_name), 'w')
133         phi1_file = open(os.path.join(save_path, phi1_file_name), 'w')
134
135         y_slice = Ly/2
136         x_coarse = np.linspace(0,Lx-0.001,200)
137         for ix in x_coarse:
138             eta_file.write('%-25s  %-25s %-25s\n' %(str(ix), str(H0 + eta.
    at(ix,y_slice)), str(eta.at(ix,y_slice))))
139             phi1_file.write('%-25s %-25s\n' %(str(ix), str(phi.at(ix,
    y_slice))))
```

```
140
141
142             ax1.plot(xvals, eta1vals * (10 ** factor), color[i-1],label = f' $\
    eta_n: t = {t:.3f}$')
143             ax2.plot(xvals,phi1vals, color[i-1], label = f' $\phi_n: t = {t:.3f
    }$')
144
145             if ic == 1:
146                 ax1.plot(xvals, etaevals* (10 ** factor), colore[i-1], label =
    f'$h_e: t = {t:.3f}$ ')
147                 ax2.plot(xvals, phievals,  colore[i-1], label = f'$\phi_e: t =
    {t:.3f}$ ')
148             else:
149                 pass
150             ax1.legend(loc=4)
151             ax2.legend(loc=4)
152
153         outfile_eta.write( eta_new )
154         outfile_phi.write( phi_new )
155
156         E1_t.append(Et)
157         E1_k.append(Ekk)
158         E1_p.append(Epp)
159
160         phi.assign(phi_new)
161         eta.assign(eta_new)
162
163
164     f.close()
165     eta_file.close()
166     phi1_file.close()
167
168     fig, (ax1, ax2, ax3) = plt.subplots(3)
169     fig.suptitle('Energy evolution with time',fontsize= tsize)
170     ax1.plot(time, E1_k)
171     ax1.set_ylabel('Kinetic energy[J] ',fontsize=size)
172     ax1.grid()
173
174     ax2.plot(time, E1_p)
```

```
175     ax2.set_ylabel('Potential Energy [J]',fontsize=size)
176     ax2.grid()
177 
178     ax3.plot(time, E1_t)
179     ax3.set_xlabel('$Time [s]$ ',fontsize=size)
180     ax3.set_ylabel('Total energy [J] ',fontsize=size)
181     ax3.grid()
182 
183 
184 else:
185     print(" The selected number does not match any case")
186 
187 plt.show()
188 print('*************** PROGRAM ENDS *****************')
```

7.2.2 Nonlinear shallow water equations: comparison of two implementation approaches by using symplectic-Euler scheme

Now, we share and explain the code for the nonlinear shallow water equations case. This code solves the problem by using two approaches, i.e. the classical approach of using the time-discretised weak formulations and the novel approach of implementing the time-discretised variational principle we developed in this thesis. The aim is to compare the two approaches for the implementation of variational principle for the two approaches for the implementation of variational principles for the first-order symplectic-Euler time-integration scheme. The settings and code description is according to the explanation given for linear shallow water case except this code solves nonlinear shallow water equations instead of linear.

```
1 
2 import firedrake as fd
3 import math as m
4 import numpy as np
5 from matplotlib import animation, pyplot as plt
6 import os
7 
8 print('#########################################################################')
9 print('######################  Initial parameters  ##########################')
10 print('#########################################################################')
```

```
# case = 1 (Solves SE based Non-linear SWE with piston wavemaker by using Novel approach)
# case = 2 (Solves SE based NL-SWE by with piston wavemaker by using classical approach)
case = 1
ramp = 0
start_wavemaker = 2 # (start_wavemaker = 1 => wavemaker started to move, start_wavemaker = 2 => Wavemaker starts and then stops)
ic = 0                                                  #  ic = 1 to use ics = func, ic = 0 use ics as 0
settings = 2                                            # settings for wavemaker, 1 == original , 2 == yangs settings
alp = 1
dt = 0.02                                               # time step
print('Time step size =', dt)
save_path =  'data_SWE_SE'
if not os.path.exists(save_path):
            os.makedirs(save_path)

H0 = 1                                                   # water depth
g = 9.8                                                  # gravitational acceleration
c = np.sqrt(g*H0)                                        # wave speed
#__________________  FIGURE PARAMETERS  _____________________#
tsize = 18 # font size of image title
size = 16  # font size of image axes
factor = 2
t = 0

##__________________   Parameters for wave   _____________________##
print("#########################################################################")
print('#########################   PARAMETERS  of Wave  ####################')
print("#########################################################################")

t = 0 # start time
m1 = 1
m2 = 0

k1 = (2* fd.pi * m1) /Lx
```

```
print('Wavenumber in x direction (k1) =',k1)

k2 = 0 #(2* fd.pi * m2) /Ly
print('Wavenumber in y direction (k2)  =',k2)

w = c * np.sqrt(k1**2 + k2**2)
print('wave frequency (w)',w )

k = np.sqrt(k1**2 + k2**2)
print('Total wavenumber (k) =',k)

Tp = (2* fd.pi ) /w
print('Time period of wave (Tp) =',Tp)

##_____________________ Parameters for wavemaker _____________________##
print("#########################################################################")
print('########################## Parameters for wavemaker ##################')
print("#########################################################################")
A_max = 0.002

lamb =  70                                                  # Wavelength
print('Wavelength of wavemaker=', lamb)

kp = 2*fd.pi/lamb                                           # Wave number
print('Wavemaker wave number (kp) =',kp)

sigma =   c * fd.sqrt(kp**2) #fd.sqrt(g*kp*fd.tanh(kp*H0))  # Wavemaker
    frequency
print('Wavemaker frequency (sigma) =', sigma)

Tw = 2*fd.pi/sigma                                          # Wavemaker  period
print('Time period of wavemaker (Tw )=', Tw)

t_end =  2*Tw                                               # time of
    simulation in sec
print('End time =', t_end)

t_steps = int(t_end/dt)
print('time_steps =', t_steps)
```

```
81
82 t_stop = Tw
83 gamma = A_max
84 ##_____________ Plot to spot the region of wavemaker frequency ___________##
85
86 lam = np.linspace(1, 200,200)
87 k_plot = 2*fd.pi/lam
88 w_pot =  ( np.sqrt(g* k_plot * np.tanh(k_plot*H0)))
89 w_shallow = c * np.sqrt(k_plot**2)
90 Time_period = 2*fd.pi/w_pot
91
92 fig, ((ax1, ax2)) = plt.subplots(2)
93 ax1.set_title(" Wave frequency ($\omega$) vs. wave number (k)",fontsize=tsize)
94 ax1.plot(k_plot, w_pot,  'k--',label = '$Potential$')
95 ax1.plot(kp, sigma,  'ro')
96 ax1.plot(k_plot, w_shallow , 'r--', label = '$Shallow$')
97 ax1.set_ylabel('$\omega $ ',fontsize=size)
98 ax1.legend(loc=1)
99 ax1.grid()
100
101 ax2.plot(k_plot,w_pot,   'k--',label = '$Potential$')
102 ax2.plot(k_plot, w_shallow , 'r--', label = '$Shallow $')
103 ax2.plot(kp, sigma,  'ro')
104 ax2.set_xlabel('k ',fontsize=size)
105 ax2.set_ylabel('$\omega $ ',fontsize=size)
106 ax2.set_xlim([0.05, 3])
107 ax2.legend(loc=1)
108 ax2.grid()
109
110 ##_____________  To get results at different time steps _____________##
111
112 time = []
113 while (t <= t_end):
114         t+= dt
115         time.append(t)
116
117 x2 = int(len(time)/2)
118 t_plot = np.array([ time[0], time[x2], time[-1] ])
119 print("t_plot =", t_plot)
```

```
120
121 lim1 = t_stop
122 i = 0
123 color= np.array(['g-', 'b--', 'r:'])
124 colore= np.array(['k:', 'c--', 'm:'])
125
126 ##____________________ Parameters for IC ___________________________##
127 if ic == 1:
128     print('#####################################################################')
129     print('################## Parameters of ICs and Exact ####################')
130     print('#####################################################################')
131     if case ==1 :
132         A0 = 0.009
133         B0 =0.009
134     else:
135         A0 = 0.009
136         B0 = 0.009
137     print('A0 =', A0)
138     print('B0 =', B0)
139
140     Uo = gamma
141     tic = 0
142     aic = np.exp(-1j * sigma * tic)
143     print('aic =', aic)
144
145 ##_______________________  Parameters for Exact Sol ________________________##
146     P = (kp * fd.sin(kp * Lx))
147     print('P = (kp * fd.sin(int(kp) * Lx)) =',P)
148
149     U_0 = Uo * 1j * sigma
150
151     a = np.exp(-1j * sigma * t_end)
152     print("Real part of exp(-1j * sigma * t_end) =",a.real)
```

Step 1: define the computational domain and mesh

```
1
2 #_________________________ MESH  ________________________#
3 nx = 200
4 n = nx
```

```
ny = 1
dx= 1/nx
Lx =  140     # make it equal to wavelength
Ly = 40
print("Lx =", Lx)
print('Ly =', Ly)
print("Nodes in x direction =", nx)

mesh = fd.RectangleMesh(nx, ny, Lx, Ly)
x,y = fd.SpatialCoordinate(mesh)
Lw =  1                          # Point till which coordinates trandformation
    will happen
print('Lw =', Lw)

xvals = np.linspace(0, Lx-0.001   , nx)
yvals = np.linspace(0, Ly- 0.001   , ny)
yslice = Ly/2
xslice = Lx/2

wavemaker_id = 1                                  # 1 => left side of the domain
```

Step 2: define the function spaces

```

#_________________  Define function spaces  ___________________##

V = fd.FunctionSpace(mesh, "CG", 1)                       # scalar function
    space

phi = fd.Function(V, name = "phi")                            # phi^n
phi_new = fd.Function(V, name = "phi_new")                    # phi^n+1

h = fd.Function(V, name =  "eta")                             # h^n
h_new = fd.Function(V, name =  "eta_new")                     # h^n+1

#______________________ Exact solution _______________________#

phie= fd.Function(V, name = "phi_exact")
he = fd.Function(V, name = "h_exact")
etae = fd.Function(V, name = "eta_exact")
```

```
17
18 #_____________________ Wavemaker _____________________#
19
20 R = fd.Function(V, name = "wavemaker")                          # Wavemaker motion
21 Rt = fd.Function(V, name = "wavemaker motion")                  # Wavemaker
       velocity
22
23 Rh = fd.Function(V, name = "wavemaker")                         # Wavemaker motion
        till Lw
24 Rht = fd.Function(V, name = "wavemaker_velocity")               # Wavemaker
       velocity with Heaviside
25
26 Rh_new = fd.Function(V, name = "wavemaker")                          # Wavemaker
       motion till Lw at t+1
27
28 W_new = fd.Function(V, name = "Lw - Rh_new")
29 W = fd.Function(V, name = "Lw - Rh")
30
31 X = fd.Function(V, name = "x_coord - Lw")
```

Step 3: define the test function

```
1 trial = fd.TrialFunction(V)                                  # trail function
2
3 v = fd.TestFunction(V)
```

Step 4: assign initial conditions to the function spaces

```
1
2             ##############################################
3             #           Initial Conditions               #
4             ##############################################
5
6 print('Initial conditions')
7 if ic ==1:
8     ic1 = phi.interpolate( (U_0.real)/P * aic.real* fd.cos(kp * (x - Lx)) \
9                           + (g/w) * fd.cos(k1 * x) * ( -A0*fd.sin(w * tic) + B0
       *fd.cos(w * tic) ))
10
```

```
    ic2 =  h.interpolate( (((1j*sigma)/ g) * (U_0.real)/(P) * np.exp(-1j *
    sigma * tic)).real * fd.cos(kp * (x - Lx))\
                                +  fd.cos(k1 * x) * ( A0*fd.cos(w * tic) + B0
    *fd.sin(w * tic) ) + H0)
else:
    ic1 = phi.assign (0)
    ic2 = h.assign(1.0)

phi.assign(ic1)
phi_new.assign(ic1)

h.assign(ic2)
h_new.assign(ic2)

etavals = np.array([ic2.at(x, yslice) for x in xvals])
phivals = np.array([ic1.at(x, yslice) for x in xvals])

fig, ((ax1, ax2)) = plt.subplots(2)
ax2.plot(xvals, phivals , label = '$\phi$')
ax1.plot(xvals, etavals, label = '$h$')
ax1.set_ylabel('$h(x,t)$ [m] ',fontsize=size)
if ic == 1:
    pass
else:
    ax1.set_ylim([0.0, 1.5])

ax1.grid()
ax2.set_xlabel('$x$ [m] ',fontsize=size)
ax2.set_ylabel('$\phi (x,t)$ ',fontsize=size)
ax2.grid()

            ##############################################
            #                   Wavemaker                #
            ##############################################

print('################ Wavemaker motion calculations block #################')

nt = 0
nnt = np.linspace(0, t_end, t_steps+1)
```

```
48
49 ##__________________  Plot of wavemaker motion  _____________________##
50 print('Plot of wavemaker motion')
51 Rt1 = []
52 Rh1 = []
53 lim = t_stop          # time after which wavemaker stops
54
55 if start_wavemaker == 2:
56         print('The wavemaker will stop after time  =',lim)
57
58 t = 0
59
60 for nt in range(len(nnt)):
61     if start_wavemaker  == 1:
62         if settings == 1:
63             R_h1 = -gamma *(np.exp(-1j * sigma *t)).real
64             Rt_1 = gamma * ((1j * sigma) * np.exp(-1j * sigma *t)).real
65         else:
66             R_h1 = -gamma*fd.cos(sigma*t)
67             Rt_1 = gamma*sigma*fd.sin(sigma*t)
68
69     elif start_wavemaker  == 2:
70         if settings == 1:
71                 R_h1 = -gamma *(np.exp(-1j * sigma *t)).real
72                 Rt_1 = gamma * ((1j * sigma) * np.exp(-1j * sigma *t)).real
73
74                 if t >= t_stop:
75                     R_h1 = -gamma *(np.exp(-1j * sigma * t_stop)).real
76                     Rt_1 = 0 *  gamma * ((1j * sigma) * np.exp(-1j * sigma *
    t_stop)).real
77         elif settings == 2:
78                 R_h1 = -gamma*fd.cos(sigma*t)
79                 Rt_1 = gamma*sigma*fd.sin(sigma*t)
80
81                 if t >= t_stop:
82                     R_h1 = -gamma*fd.cos(sigma*t_stop)
83                     Rt_1 = 0*gamma*sigma*fd.sin(sigma*t_stop)
84     else:
85         R_h1 = fd.Constant(0)
```

```
86          Rt_1 = fd.Constant(0)
87
88      t+=dt
89      Rt1.append(Rt_1)
90      Rh1.append(R_h1)
91
92  if start_wavemaker == 1:
93      Amp_wave = max(Rh1)
94      print('Maximum amplitude of wavemaker =', Amp_wave)
95      vel_wave = max(Rt1)
96      print('Maximum velocity of wavemaker =', vel_wave)
97  else:
98      pass
99
100 fig, (ax1, ax2) = plt.subplots(2)
101
102 ax1.set_title('Wavemaker Position',fontsize=tsize)
103 ax1.plot(nnt, Rh1, 'r-', label = f'$h_e: t = {t:.3f}$ ')
104 ax1.set_ylabel('$R(t)[m]$ ',fontsize=size)
105 ax1.grid()
106
107 ax2.set_title('Wavemaker velocity',fontsize=tsize)
108 ax2.plot(nnt, Rt1,  'r-', label = f'$\phi_e: t = {t:.3f}$ ')
109 ax2.set_xlabel('$Time [s]$ ',fontsize=size)
110 ax2.set_ylabel('$R_{t} [m/s]$ ',fontsize=size)
111 ax2.grid()
112 ##_________________  FIGURE SETTINGS ___________________________##
113 print('Figure settings')
114
115 fig, (ax1, ax2) = plt.subplots(2)
116 ax1.set_title('Initial Conditions',fontsize=tsize)
117 ax1.set_title(r'$h $ value in $x$ direction',fontsize=tsize)
118 ax1.set_ylabel(r'$h(x,t)\times 10^{-2} [m]$ ',fontsize=size)
119 ax1.grid()
120 ax2.set_xlabel(r'$x [m]$ ',fontsize=size)
121 ax2.set_ylabel(r'$\phi (x,t)\times 10^{-2} $ ',fontsize=size)
122 ax2.grid()
```

Step 5: define the variational problem

The symplectic-Euler time-discrete version of the transformed VP is

$$
\begin{aligned}
0 =& \delta \int_0^L \Big[\phi^n \Big(X \tilde{R}_\tau^n h_\xi^{n+1} + W^n \frac{h^{n+1} - h^n}{\Delta t} \Big) - \phi^{n+1} \frac{h^{n+1}}{\Delta t} W^{n+1} \\
& - \frac{1}{2} \frac{L_w^2}{W^n} h^{n+1} (\phi_\xi^n)^2 - \frac{1}{2} W^n g (h^{n+1})^2 \\
& + W^n g H_0 h^{n+1} \Big] \mathrm{d}\xi - L_w R_\tau^n h^{n+1} \phi^n |_{\xi=0}
\end{aligned} \tag{7.9}
$$

which is directly implemented into Firedrake.

```
#######################  VARIATIONAL PRINCIPLE  ##############################
print("#########################################################################")
print('########################## Numerical Calculations   ##################')
print("#########################################################################")

t = 0

if case == 1:
    print('############################################################')
    print("You have selected case 1 : Non_Linear SWE VP with piston wavemaker
    solved by firedrake by using fd.derivative ")
    print(" Time discrete VP is based on Symplectic-Euler scheme  ")
    print('############################################################')
    E2_t = []
    E2_k = []
    E2_p = []

    pot_ener = Lx * Ly * 9.8 * 0.5
    x = fd.SpatialCoordinate(mesh)
    x_coord = fd.Function(V).interpolate(x[0])

    ################################### VP ###################################


    # VP = ( (x_coord - Lw)  * Rht * fd.inner( h_new.dx(0), phi )\
    #       +  fd.inner ((Lw - Rh) * (h_new - h)/dt , phi ) \
    #       - (Lw - Rh_new) * fd.inner(phi_new , (h_new/dt)) \
    #       - 1/2 * (Lw**2/(Lw - Rh)) * h_new * fd.inner(fd.grad(phi), fd.grad(
    phi))\
```

```
28    #       - (1/2 * g * (Lw - Rh) *fd.inner(h_new,h_new))\
29    #       + (Lw - Rh)* g * H0 * h_new    ) * fd.dx \
30    #       - (Lw * Rt * h_new*phi)*fd.ds(1)
31
32  ##_________________ VP with short hand notations _____________________##
33    # Using short hand notation
34    # X = x_coord - Lw
35    # W = (Lw - Rh)
36    # W_new = (Lw - Rh_new)
37
38    VP = ( (X)  * Rht * fd.inner( h_new.dx(0), phi )\
39          +  fd.inner (W * (h_new - h)/dt , phi ) \
40          - (W_new) * fd.inner(phi_new , (h_new/dt)) \
41          - 1/2 * (Lw**2/(W)) * h_new * fd.inner(fd.grad(phi), fd.grad(phi))\
42          - (1/2 * g * (W) *fd.inner(h_new,h_new))\
43          + (W)* g * H0 * h_new    ) * fd.dx \
44          - (Lw * Rt * h_new*phi)*fd.ds(1)
45
46
47    ##########################################################################
48
49    h_expr = fd.derivative(VP, phi, v)  # derivative of VP wrt phi^n to get the
       expression for h^n+1
50    phi_expr = fd.derivative(VP, h_new, v)  # derivative of VP wrt h^n+1 to get
       the value of phi^n+1
51
52
53    h_expr = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(
      h_expr, h_new))
54
55    phi_expr = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(
      phi_expr, phi_new))
56
57    ###_________________  OUTPUT FILES ______________###
58    if start_wavemaker ==1:
59            outfile_phi = fd.File("results_SE_NLSWE_wm1_case2/phi.pvd")
60            outfile_eta = fd.File("results_SE_NLSWE_wm1_case2/eta.pvd")
61    elif start_wavemaker == 2:
62            outfile_phi = fd.File("results_SE_NLSWE_wm2_case2/phi.pvd")
```

```
63            outfile_eta = fd.File("results_SE_NLSWE_wm2_case2/eta.pvd")
64    elif start_wavemaker == 0:
65            outfile_phi = fd.File("results_SE_NonLinSWE_wm0_case2/phi.pvd")
66            outfile_eta = fd.File("results_SE_NonLinSWE_wm0_case2/eta.pvd")
67    ###_________________  TXT FILES ______________________###
68    if start_wavemaker == 1:
69        filename1 = "NLSWE_SE_wm1.txt"
70        filename2 = "eta_NLSWE_SE_wm1.txt"
71        filename3 = "phi_NLSWE_SE_wm1.txt"
72    elif start_wavemaker == 2:
73        filename1 = "NLSWE_SE_wm2.txt"
74        filename2 = "eta_NLSWE_SE_wm2.txt"
75        filename3 = "phi_NLSWE_SE_wm2.txt"
76    elif start_wavemaker == 0:
77        filename1 = "NLSWE_SE_wm0.txt"
78        filename2 = "eta_NLSWE_SE_wm0.txt"
79        filename3 = "phi_NLSWE_SE_wm0.txt"
80
81    f = open(filename1 , 'w+')
```

Step 6: solve the variational problem

```
1     ######### TIME LOOP #############
2
3     while (t <= t_end):
4         tt = format(t, '.3f')
5         t_new = t+dt
6
7         X.interpolate( x_coord - Lw)
8         W.interpolate(Lw - Rh)
9         W_new.interpolate(Lw - Rh_new)
10    ## ______________________  wavemaker motion  ________________________ ##
11        if start_wavemaker == 1:
12                        R.assign(-gamma  * fd.cos(sigma*t))
13                        Rt.assign( gamma * sigma * fd.sin(sigma*t))
14                        Rh.interpolate(fd.conditional(fd.le(x_coord,Lw), -gamma
      * fd.cos(sigma * t), 0.0) )
15                        Rht.interpolate(fd.conditional(fd.le(x_coord,Lw),gamma
     * sigma * fd.sin(sigma*t),0.0))
16                        Rh_new.interpolate(fd.conditional(fd.le(x_coord,Lw), -
```

```
    gamma  * fd.cos(sigma * t_new), 0.0) )
17              # # wavemaker moves at first and then stops after some time
18          if start_wavemaker == 2:
19                              R.assign(-gamma * fd.cos(sigma*t))
20                              Rt.assign( gamma * sigma * fd.sin(sigma*t))
21                              Rh.interpolate(fd.conditional(fd.le(x_coord,Lw), -gamma
      * fd.cos(sigma * t), 0.0) )
22                              Rht.interpolate(fd.conditional(fd.le(x_coord,Lw),gamma
     * sigma * fd.sin(sigma*t),0.0))
23                              Rh_new.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_new), 0.0) )
24
25                              if t >= t_stop:
26                                  R.assign(-gamma *fd.cos(sigma*t_stop))
27                                  Rt.assign(0)
28                                  Rh.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_stop), 0.0) )
29                                  Rht.assign(0)
30                                  Rh_new.interpolate(fd.conditional(fd.le(x_coord,Lw)
    , -gamma  * fd.cos(sigma * (t_stop+ dt)), 0.0) )
31
32              # # wavemaker does not move at all
33          elif start_wavemaker == 0:
34                              Rt.assign(0)
35                              R.assign(0)
36                              Rh.assign(0)
37                              Rht.assign(0)
38                              Rh_new.assign(0)
39
40      ## ------------------------------------------------------------------- ##
41          h_expr.solve()
42          phi_expr.solve()
43
44          t+= dt
45
46          Epp1 = fd.assemble( ( 1/2 * g * fd.inner(h,h))* ((Lw - Rh)/Lw) * fd.dx)
47          Epp2 = fd.assemble( ( g *h * H0)* ((Lw - Rh)/Lw) * fd.dx )
48
49
```

```
50        Epp = fd.assemble( (Lw - Rh)*( g*h*(0.5*h - H0) )* fd.dx  )
51        Ekk = fd.assemble(0.5 * (Lw**2/(Lw - Rh)) * h * fd.inner(fd.grad(phi),
    fd.grad(phi)) * fd.dx )
52
53
54        Et = abs(Ekk) + abs(Epp)
```

Step 7: Plot and output the results

Finally, within the time loop, we compute and write the value of variables in *.txt* files and generate *.pvd* files for visualisation.

```
1
2         f.write('%-25s %-25s %-25s %-25s %-25s %-25s %-25s %-25s %-25s\n'
     \
3                 % (str(t), str(R.dat.data[2]), str(Rt.dat.data[2]), str(phi.at
    (0,0)),\
4                     str(h.at(0,0)), str(Epp), str(Ekk), str(Et) , str(Epp1), str
    (Epp2) ) )
5                 # % (str(t), str(R.dat.data[2]), str(Rt.dat.data[2]), str(phi.
    at(0,0)), str(h.at(0,0)), str(4410.367500000001 - Epp), str(Ekk), str(Et) )
    )
6
7         if (t in t_plot):
8             print('Plotting starts')
9             print('t =', t)
10            i += 1
11            if ic == 1:
12                phi_exact = phie.interpolate( (U_0.real)/P * a.real* fd.cos(kp
    * (x_coord - Lx)) \
13                                + (g/w) * fd.cos(k1 * x_coord) * ( -A0*fd.sin(w *
     t) + B0*fd.cos(w * t) ) )
14
15                h_exact = he.interpolate( (((1j*sigma)/ g) * (U_0.real)/(P) *
    np.exp(-1j * sigma * t_end)).real * fd.cos(kp * (x_coord - Lx))\
16                                    +  fd.cos(k1 * x_coord) * ( A0*fd.cos(w * t)
    + B0*fd.sin(w * t) ) + H0 )
17
18                phievals = np.array([phi_exact.at(x, yslice) for x in xvals])
19                etaevals = np.array([h_exact.at(x, yslice) for x in xvals])
```

```
20
21            else:
22                pass
23
24            phi1vals = np.array([phi_new.at(x, Ly/2) for x in xvals])
25            h1vals = np.array([h_new.at(x, Ly/2) for x in xvals])
26
27            if start_wavemaker == 1:
28                h_file_name = 'h_SE_nlswe_wm1_'+tt+'.txt'
29                phi_file_name = 'phi_SE_nlswe_wm1_'+tt+'.txt'
30            elif start_wavemaker == 2:
31                 h_file_name = 'h_SE_nlswe_wm2_'+tt+'.txt'
32                 phi_file_name = 'phi_SE_nlswe_wm2_'+tt+'.txt'
33            elif start_wavemaker == 0:
34                 h_file_name = 'h_SE_nlswe_wm0_'+tt+'.txt'
35                 phi_file_name = 'phi_SE_nlswe_wm0_'+tt+'.txt'
36
37            h_file = open(os.path.join(save_path, h_file_name), 'w')
38            phi_file = open(os.path.join(save_path, phi_file_name), 'w')
39
40            y_slice = Ly/2
41            x_coarse = np.linspace(0,Lx-0.001,200)
42            for ix in x_coarse:
43                h_file.write('%-25s  %-25s\n' %(str(ix), str(h.at(ix,y_slice)))
    )
44                phi_file.write('%-25s  %-25s\n' %(str(ix), str(phi.at(ix,
    y_slice))))
45
46            ax1.plot(xvals, h1vals * (10 ** factor), color[i-1],label = f' $\
    eta_n: t = {t:.3f}$')
47            ax2.plot(xvals,phi1vals, color[i-1], label = f' $\phi_n: t = {t:.3f
    }$')
48
49
50            if ic == 1:
51                ax1.plot(xvals, etaevals* (10 ** factor), colore[i-1], label =
    f'$h_e: t = {t:.3f}$ ')
52                ax2.plot(xvals, phievals,  colore[i-1], label = f'$\phi_e: t =
    {t:.3f}$ ')
```

```
            else:
                pass
            ax1.legend(loc=4)
            ax2.legend(loc=4)


        outfile_eta.write( h_new )
        outfile_phi.write( phi_new )

        # set-up next time-step
        E2_t.append(abs(Et))
        E2_p.append(abs(Epp))
        E2_k.append(abs(Ekk))

        phi.assign(phi_new)
        h.assign(h_new)

    f.close()
    h_file.close()
    phi_file.close()

    fig, (ax1, ax2, ax3) = plt.subplots(3)
    fig.suptitle('Energy evolution with time',fontsize= tsize)
    ax1.plot(time, E2_k)
    ax1.set_ylabel('Kinetic energy[J] ',fontsize=size)
    ax1.grid()

    ax2.plot(time, E2_p)
    ax2.set_ylabel('Potential Energy [J]',fontsize=size)
    ax2.grid()

    ax3.plot(time, E2_t)
    ax3.set_xlabel('$Time [s]$ ',fontsize=size)
    ax3.set_ylabel('Total energy [J] ',fontsize=size)
    ax3.grid()
```

The complete code is publicly available as **Lin_SWE_piston.py** on the GitHub repository.

Case 2: solve the nonlinear SWE VP with piston wavemaker by using time discrete weak formulations.

The time-discrete weak formulations based on the symplectic-Euler scheme are as follows:

$$\int_0^L W^n \frac{h^{n+1}-h^n}{\Delta t}\delta\phi^n \,\mathrm{d}\xi = \int_0^L -X\tilde{R}_\tau^n h_\xi^{n+1}\delta\phi^n + \frac{L_w^2}{W^n}h^{n+1}\phi_\xi^n\,\partial_\xi(\delta\phi^n)\,\mathrm{d}\xi + L_w R_\tau^n h^{n+1}\delta\phi^n\Big|_{\xi=0}; \tag{7.10}$$

$$\int_0^L \frac{W^{n+1}\phi^{n+1}-W^n\phi^n}{\Delta t}\delta h^{n+1}\,\mathrm{d}\xi = \int_0^L X\tilde{R}_\tau^n\phi^n\partial_\xi(\delta h^{n+1}) - \frac{1}{2}\frac{L_w^2}{W^n}(\phi_\xi^n)^2\delta h^{n+1} - gW^n(h^{n+1}-H_0)\delta h^{n+1}\,\mathrm{d}\xi - L_w R_\tau^n\phi^n\delta h^{n+1}\Big|_{\xi=0}. \tag{7.11}$$

```
elif case == 2:

    print('############################################################')
    print("You have selected case 2 : Non_Linear SWE VP with piston wavemaker
    solved by using time discrete weak formulations. ")
    print("Time discrete weak formulations are based on Symplectic-Euler scheme
    .  ")
    print('############################################################')
    E2_t = []
    E2_k = []
    E2_p = []

    pot_ener = Lx * Ly * 9.8 * 0.5
    DBC = fd.DirichletBC( V, Rht , wavemaker_id)
    x = fd.SpatialCoordinate(mesh)
    y = 0
    x_coord = fd.Function(V).interpolate(x[0])


    ##################################### VP ###################################


    h_expr = ( fd.inner ( (Lw - Rh)* (h_new - h)/dt , v) \
              + (x_coord - Lw)* Rht * h_new.dx(0) * v \
              - ((Lw**2)/(Lw - Rh)) * h_new * phi.dx(0) * v.dx(0) )*fd.dx\
```

```
            - (Lw * Rt * h_new *v )* fd.ds(1)


 phi_expr = (fd.inner ( ( ((Lw - Rh_new)* phi_new) - ((Lw - Rh)*phi) )/dt, v ) \
             - (x_coord - Lw) * Rht * phi *  v.dx(0)  \
             + 0.5 * ((Lw**2)/(Lw - Rh)) *  fd.inner(fd.grad(phi), fd.grad(phi)) * v \
             + g * (Lw - Rh) * (h_new - H0) * v)  *fd.dx \
             + (Lw * Rt * phi * v) * fd.ds(1)


 h_expr  = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(h_expr, h_new))

 phi_expr = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(phi_expr, phi_new))


 ###_________________  OUTPUT FILES ______________###
 if start_wavemaker ==1:
         outfile_phi = fd.File("results_SE2_NLSWE_wm1/phi.pvd")
         outfile_eta = fd.File("results_SE2_NLSWE_wm1/eta.pvd")
 elif start_wavemaker == 2:
         outfile_phi = fd.File("results_SE2_NLSWE_wm2_case2/phi.pvd")
         outfile_eta = fd.File("results_SE2_NLSWE_wm2_case2/eta.pvd")
 elif start_wavemaker == 0:
         outfile_phi = fd.File("results_SE2_NLSWE_wm0/phi.pvd")
         outfile_eta = fd.File("results_SE2_NLSWE_wm0/eta.pvd")
 ###_________________  TXT FILES ______________###
 if start_wavemaker == 1:
     filename1 = "NLSWE_SE2_wm1.txt"
     filename2 = "eta_NLSWE_SE2_wm1.txt"
     filename3 = "phi_NLSWE_SE2_wm1.txt"
 elif start_wavemaker == 2:
     filename1 = "NLSWE_SE2_wm2.txt"
     filename2 = "eta_NLSWE_SE2_wm2.txt"
     filename3 = "phi_NLSWE_SE2_wm2.txt"
 elif start_wavemaker == 0:
```

```
59          filename1 = "NLSWE_SE2_wm0.txt"
60          filename2 = "eta_NLSWE_SE2_wm0.txt"
61          filename3 = "phi_NLSWE_SE2_wm0.txt"
62      # filename = "NL_SWE.txt"
63      f = open(filename1 , 'w+')
64      ######### TIME LOOP ############
65
66      while (t <= t_end):
67          tt = format(t, '.3f')
68          t_new = t+dt
69      ## ----------------------  wavemaker motion  ------------------------- ##
70          if start_wavemaker == 1:
71                          R.assign(-gamma  * fd.cos(sigma*t))
72                          Rt.assign( gamma * sigma * fd.sin(sigma*t))
73                          Rh.interpolate(fd.conditional(fd.le(x_coord,Lw), -gamma
      * fd.cos(sigma * t), 0.0) )
74                          Rht.interpolate(fd.conditional(fd.le(x_coord,Lw),gamma
     * sigma * fd.sin(sigma*t),0.0))
75                          Rh_new.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_new), 0.0) )
76              # # wavemaker moves at first and then stops after some time
77          if start_wavemaker == 2:
78                          R.assign(-gamma * fd.cos(sigma*t))
79                          Rt.assign( gamma * sigma * fd.sin(sigma*t))
80                          Rh.interpolate(fd.conditional(fd.le(x_coord,Lw), -gamma
      * fd.cos(sigma * t), 0.0) )
81                          Rht.interpolate(fd.conditional(fd.le(x_coord,Lw),gamma
     * sigma * fd.sin(sigma*t),0.0))
82                          Rh_new.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_new), 0.0) )
83
84                          if t >= t_stop:
85                              R.assign(-gamma *fd.cos(sigma*t_stop))
86                              Rt.assign(0)
87                              Rh.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_stop), 0.0) )
88                              Rht.assign(0)
89                              Rh_new.interpolate(fd.conditional(fd.le(x_coord,Lw)
    , -gamma  * fd.cos(sigma * (t_stop+ dt)), 0.0) )
```

```
90
91              # # wavemaker does not move at all
92          elif start_wavemaker == 0:
93                          Rt.assign(0)
94                          R.assign(0)
95                          Rh.assign(0)
96                          Rht.assign(0)
97                          Rh_new.assign(0)
98      ## ------------------------------------------------------------------- ##
99
100         h_expr.solve()
101         phi_expr.solve()
102
103         t+= dt
104
105         Epp1 = fd.assemble( ( 1/2 * g * fd.inner(h,h))* ((Lw - Rh)/Lw) * fd.dx)
106         Epp2 = fd.assemble( ( g *h * H0)* ((Lw - Rh)/Lw) * fd.dx )
107
108         Epp = fd.assemble( (Lw - Rh)*( g*h*(0.5*h - H0) )* fd.dx  )
109         Ekk = fd.assemble(0.5 * (Lw**2/(Lw - Rh)) * h * fd.inner(fd.grad(phi),
    fd.grad(phi)) * fd.dx )
110
111         Et = abs(Ekk) + abs(Epp)
112
113         f.write('%-25s %-25s %-25s %-25s %-25s %-25s %-25s %-25s %-25s\n'
     \
114                 % (str(t), str(R.dat.data[2]), str(Rt.dat.data[2]), str(phi.at
    (0,0)),\
115                    str(h.at(0,0)), str(Epp), str(Ekk), str(Et) , str(Epp1), str
    (Epp2) ) )
116                 # % (str(t), str(R.dat.data[2]), str(Rt.dat.data[2]), str(phi.
    at(0,0)), str(h.at(0,0)), str(4410.367500000001 - Epp), str(Ekk), str(Et) )
    )
117
118         if (t in t_plot):
119             i+=1
120             print('Plotting starts')
121             print('t =', t)
122             if ic == 1:
```

```
123             phi_exact = phie.interpolate( (U_0.real)/P * a.real* fd.cos(kp
    * (x_coord - Lx)) \
124                             + (g/w) * fd.cos(k1 * x_coord) * ( -A0*fd.sin(w *
     t) + B0*fd.cos(w * t) ) )
125 
126             h_exact = he.interpolate( (((1j*sigma)/ g) * (U_0.real)/(P) *
    np.exp(-1j * sigma * t_end)).real * fd.cos(kp * (x_coord - Lx))\
127                                 +  fd.cos(k1 * x_coord) * ( A0*fd.cos(w * t)
    + B0*fd.sin(w * t) ) + H0 )
128 
129             phievals = np.array([phi_exact.at(x, yslice) for x in xvals])
130             etaevals = np.array([h_exact.at(x, yslice) for x in xvals])
131 
132         else:
133             pass
134 
135         phi1vals = np.array([phi_new.at(x, Ly/2) for x in xvals])
136         h1vals = np.array([h_new.at(x, Ly/2) for x in xvals])
137 
138         if start_wavemaker == 1:
139             h_file_name = 'h_SE2_nlswe_wm1_'+tt+'.txt'
140             phi_file_name = 'phi_SE2_nlswe_wm1_'+tt+'.txt'
141         elif start_wavemaker == 2:
142              h_file_name = 'h_SE2_nlswe_wm2_'+tt+'.txt'
143              phi_file_name = 'phi_SE2_nlswe_wm2_'+tt+'.txt'
144         elif start_wavemaker == 0:
145              h_file_name = 'h_SE2_nlswe_wm0_'+tt+'.txt'
146              phi_file_name = 'phi_SE2_nlswe_wm0_'+tt+'.txt'
147 
148         h_file = open(os.path.join(save_path, h_file_name), 'w')
149         phi_file = open(os.path.join(save_path, phi_file_name), 'w')
150 
151         y_slice = Ly/2
152         x_coarse = np.linspace(0,Lx-0.001,200)
153         for ix in x_coarse:
154             h_file.write('%-25s  %-25s\n' %(str(ix), str(h.at(ix,y_slice)))
    )
155             phi_file.write('%-25s  %-25s\n' %(str(ix), str(phi.at(ix,
    y_slice))))
```

```
156
157             ax1.plot(xvals, h1vals * (10 ** factor), color[i-1],label = f' $\
    eta_n: t = {t:.3f}$')
158             ax2.plot(xvals,phi1vals, color[i-1], label = f' $\phi_n: t = {t:.3f
    }$')
159
160
161             if ic == 1:
162                 ax1.plot(xvals, etaevals* (10 ** factor), colore[i-1], label =
    f'$h_e: t = {t:.3f}$ ')
163                 ax2.plot(xvals, phievals,  colore[i-1], label = f'$\phi_e: t =
    {t:.3f}$ ')
164             else:
165                 pass
166             ax1.legend(loc=4)
167             ax2.legend(loc=4)
168
169
170         outfile_eta.write( h_new )
171         outfile_phi.write( phi_new )
172
173         # set-up next time-step
174         E2_t.append(abs(Et))
175         E2_p.append(abs(Epp))
176         E2_k.append(abs(Ekk))
177
178         phi.assign(phi_new)
179         h.assign(h_new)
180
181     f.close()
182     h_file.close()
183     phi_file.close()
184
185     fig, (ax1, ax2, ax3) = plt.subplots(3)
186     fig.suptitle('Energy evolution with time',fontsize= tsize)
187     ax1.plot(time, E2_k)
188     ax1.set_ylabel('Kinetic energy[J] ',fontsize=size)
189     ax1.grid()
190
```

```
    ax2.plot(time, E2_p)
    ax2.set_ylabel('Potential Energy [J]',fontsize=size)
    ax2.grid()

    ax3.plot(time, E2_t)
    ax3.set_xlabel('$Time [s]$ ',fontsize=size)
    ax3.set_ylabel('Total energy [J] ',fontsize=size)
    ax3.grid()

else:
    print(" The selected number does not match any case")

plt.show()
print('*************** PROGRAM ENDS ******************')
```

The complete code is publicly available as **NL_SWE_SE.py** on the GitHub repository.

7.2.3 Nonlinear shallow water equations: comparison of two implementation approaches by using Störmer-Verlet scheme

Now, we share and explain the code for the nonlinear shallow water equations case. This code solves the problem by using two approaches, i.e. the classical approach of using the time-discretised weak formulations and the novel approach of implementing the time-discretised variational principle we developed in this thesis. The aim is to compare the two approaches for the implementation of variational principle for the second-order Störmer-Verlet (SV) time-integration scheme.

Step 0: define the user inputs

```
import firedrake as fd
import math as m
import numpy as np
from matplotlib import animation, pyplot as plt
import os

'''
Case 1 => Solves NL_SWE case by using time-discrete VP based on SV scheme.
Case 2 => Solves NL_SWE case by using time-discrete  weak formulations VP based
    on SV scheme.
```

```
10
11 '''
12 print('#########################################################################')
13 print('#######################  Initial parameters  ############################')
14 print('#########################################################################')
15
16
17 case = 2 #  Case = 1 (Non-linear SWE by SV) & case = 3 (NLSWE by SV2)
18 start_wavemaker = 2 # (start_wavemaker = 1 => wavemaker started to move,
       start_wavemaker = 2 => Wavemaker starts and then stops)
19 ic = 0                                                    #  ic = 1 to use ics
        = func, ic = 0 use ics as 0
20 settings = 2                                             # settings for
       wavemaker, 1 == original , 2 == yangs settings
21 alp = 1
22 dt = 0.02                              # time step
23 print('Time step size =', dt)
24 save_path =  'data_SWE_SV'
25 if not os.path.exists(save_path):
26             os.makedirs(save_path)
27
28
29 H0 = 1                                                     # water depth
30 g = 9.8                                                   # gravitational
       acceleration
31 c = np.sqrt(g*H0)                                         # wave speed
32
33 #__________________  FIGURE PARAMETERS  _____________________#
34
35 tsize = 18 # font size of image title
36 size = 16  # font size of image axes
37 factor = 2
38 t = 0
39
40
41 ##__________________   Parameters for wave   _____________________##
42 print("#########################################################################")
43 print('########################   PARAMETERS  of Wave  ####################')
44 print("#########################################################################")
```

```
45
46 t = 0 # start time
47 m1 = 1
48 m2 = 0
49
50 k1 = (2* fd.pi * m1) /Lx
51 print('Wavenumber in x direction (k1) =',k1)
52
53 k2 = 0 #(2* fd.pi * m2) /Ly
54 print('Wavenumber in y direction (k2)  =',k2)
55
56 w = c * np.sqrt(k1**2 + k2**2)
57 print('wave frequency (w)',w )
58
59 k = np.sqrt(k1**2 + k2**2)
60 print('Total wavenumber (k) =',k)
61
62 Tp = (2* fd.pi ) /w
63 print('Time period of wave (Tp) =',Tp)
64
65 ##------------------ Parameters for wavemaker ----------------------##
66 print("#########################################################################")
67 print('########################## Parameters for wavemaker ##################')
68 print("#########################################################################")
69 A_max = 0.002
70
71 lamb =  70                                                  # Wavelength
72 print('Wavelength of wavemaker=', lamb)
73
74 kp = 2*fd.pi/lamb                                           # Wave number
75 print('Wavemaker wave number (kp) =',kp)
76
77 sigma =   c * fd.sqrt(kp**2)   # Wavemaker frequency
78 print('Wavemaker frequency (sigma) =', sigma)
79
80 Tw = 2*fd.pi/sigma                 # Wavemaker  period
81 print('Time period of wavemaker (Tw )=', Tw)
82
```

```
t_end =  2*Tw                                                    # time of
    simulation in sec
print('End time =', t_end)

t_steps = int(t_end/dt)
print('time_steps =', t_steps)

t_stop = Tw

gamma = A_max

##_____________ Plot to spot the region of wavemaker frequency ___________##

lam = np.linspace(1, 200,200)
k_plot = 2*fd.pi/lam
w_pot =  ( np.sqrt(g* k_plot * np.tanh(k_plot*H0)))
w_shallow = c * np.sqrt(k_plot**2)
Time_period = 2*fd.pi/w_pot

fig, ((ax1, ax2)) = plt.subplots(2)
ax1.set_title(" Wave frequency ($\omega$) vs. wave number (k)",fontsize=tsize)
ax1.plot(k_plot, w_pot,  'k--',label = '$Potential$')
ax1.plot(kp, sigma,  'ro')
ax1.plot(k_plot, w_shallow , 'r--', label = '$Shallow$')
ax1.set_ylabel('$\omega $ ',fontsize=size)
ax1.legend(loc=1)
ax1.grid()

ax2.plot(k_plot,w_pot,   'k--',label = '$Potential$')
ax2.plot(k_plot, w_shallow , 'r--', label = '$Shallow $')
ax2.plot(kp, sigma,  'ro')
ax2.set_xlabel('k ',fontsize=size)
ax2.set_ylabel('$\omega $ ',fontsize=size)
ax2.set_xlim([0.05, 3])
ax2.legend(loc=1)
ax2.grid()

##_____________  To get results at different time steps _____________##

```

```
time = []
while (t <= t_end):
        t+= dt
        time.append(t)

x2 = int(len(time)/2)
t_plot = np.array([ time[0], time[x2], time[-1] ])
print("t_plot =", t_plot)

lim1 = t_stop
i = 0
color= np.array(['g-', 'b--', 'r:'])
colore= np.array(['k:', 'c--', 'm:'])

##------------------- Parameters for IC -------------------------##
if ic == 1:
    print('#########################################################################')
    print('################# Parameters of ICs and Exact ###################')
    print('#########################################################################')
    if case ==1 :
        A0 = 0.009
        B0 =0.009
    else:
        A0 = 0.009
        B0 = 0.009
    print('A0 =', A0)
    print('B0 =', B0)

    Uo = gamma
    tic = 0
    aic = np.exp(-1j * sigma * tic)
    print('aic =', aic)

##----------------------  Parameters for Exact Sol ------------------------##

    P = (kp * fd.sin(kp * Lx))
    print('P = (kp * fd.sin(int(kp) * Lx)) =',P)

    U_0 = Uo * 1j * sigma
```

```
160
161     a = np.exp(-1j * sigma * t_end)
162     print("Real part of exp(-1j * sigma * t_end) =",a.real)
```

Step 1: define the computational domain and mesh

```
1  #________________________ MESH  ________________________#
2
3  nx = 200
4  n = nx
5  ny = 1
6
7  dx= 1/nx
8
9  Lx =  140  # make it equal to wavelength
10 Ly = 40
11 print("Lx =", Lx)
12 print('Ly =', Ly)
13 print("Nodes in x direction =", nx)
14
15 mesh = fd.RectangleMesh(nx, ny, Lx, Ly)
16 x,y = fd.SpatialCoordinate(mesh)
17 Lw =  1                                    # Point till which coordinates
       trandformation will happen
18 print('Lw =', Lw)
19
20 xvals = np.linspace(0, Lx-0.001   , nx)
21 yvals = np.linspace(0, Ly- 0.001   , ny)
22 yslice = Ly/2
23 xslice = Lx/2
24
25 wavemaker_id = 1                           # 1 => left side of the domain
```

Step 2: define the function spaces

```
1  #__________________  Define function spaces  __________________##
2
3  V = fd.FunctionSpace(mesh, "CG", 1)                  # scalar function space
4
5
```

```
6
7  phi = fd.Function(V, name = "phi")                              # phi^n
8  phi_half = fd.Function(V, name = "phi")                         # phi^n+1/2
9  phi_new = fd.Function(V, name = "phi_new")                      # phi^n+1
10
11
12 h = fd.Function(V, name =  "eta")                               # h^n
13 h_new = fd.Function(V, name =  "eta_new")                       # h^n+1
14
15 #_____________________ Exact solution ________________________#
16
17 phie= fd.Function(V, name = "phi_exact")
18 he = fd.Function(V, name = "h_exact")
19 etae = fd.Function(V, name = "eta_exact")
20
21 #_____________________ Wavemaker ________________________#
22
23 R         = fd.Function(V, name = "wavemaker")          # Wavemaker motion
24 Rt        = fd.Function(V, name = "wavemaker motion")   # Wavemaker velocity
25
26
27 Rt_half       = fd.Function(V, name = "Rt_half")   # Wavemaker velocity
28
29 Rh        = fd.Function(V, name = "wavemaker")      # Wavemaker motion till Lw
30 Rh_new    = fd.Function(V, name = "wavemaker")      # Wavemaker motion till Lw
       at t+1
31 Rh_half   = fd.Function(V, name = "Rh_half ")
32 Rht_half = fd.Function(V, name = "Rht_half")
33
34 W         = fd.Function(V, name = "Lw - Rh")
35 W_half    = fd.Function(V, name = "Lw - Rh")
36 W_new     = fd.Function(V, name = "Lw - Rh_new")
37
38 X = fd.Function(V, name = "x_coord - Lw")
```

Step 3: define the test function

```
1 trial = fd.TrialFunction(V)                                      # trail function
2
3 v = fd.TestFunction(V)
```

Step 4: assign initial conditions to the function spaces

```
            ##############################################
            #           Initial Conditions               #
            ##############################################

print('Initial conditions')

if ic ==1:
    ic1 = phi.interpolate( (U_0.real)/P * aic.real* fd.cos(kp * (x - Lx)) \
                          + (g/w) * fd.cos(k1 * x) * ( -A0*fd.sin(w * tic) + B0
    *fd.cos(w * tic) ))

    ic2 =  h.interpolate( (((1j*sigma)/ g) * (U_0.real)/(P) * np.exp(-1j *
    sigma * tic)).real * fd.cos(kp * (x - Lx))\
                                  +  fd.cos(k1 * x) * ( A0*fd.cos(w * tic) + B0
    *fd.sin(w * tic) ) + H0)
else:
    ic1 = phi.assign (0)
    ic2 = h.assign(1.0)

phi.assign(ic1)
phi_new.assign(ic1)

h.assign(ic2)
h_new.assign(ic2)

etavals = np.array([ic2.at(x, yslice) for x in xvals])
phivals = np.array([ic1.at(x, yslice) for x in xvals])

fig, ((ax1, ax2)) = plt.subplots(2)
ax2.plot(xvals, phivals , label = '$\phi$')

ax1.plot(xvals, etavals, label = '$h$')
ax1.set_ylabel('$h(x,t)$ [m] ',fontsize=size)
if ic == 1:
    pass
else:
    ax1.set_ylim([0.0, 1.5])

```

```
ax1.grid()
ax2.set_xlabel('$x$ [m] ',fontsize=size)
ax2.set_ylabel('$\phi (x,t)$ ',fontsize=size)
ax2.grid()

            ##########################################
            #                   Wavemaker              #
            ##########################################

print('################ Wavemaker motion calculations block #################')

nt = 0
nnt = np.linspace(0, t_end, t_steps+1)

##__________________  Plot of wavemaker motion  ____________________##
print('Plot of wavemaker motion')
Rt1 = []
Rh1 = []
lim = t_stop          # time after which wavemaker stops

if start_wavemaker == 2:
        print('The wavemaker will stop after time  =',lim)

t = 0

for nt in range(len(nnt)):
    if start_wavemaker  == 1:
        if settings == 1:
            R_h1 = -gamma *(np.exp(-1j * sigma *t)).real
            Rt_1 = gamma * ((1j * sigma) * np.exp(-1j * sigma *t)).real
        else:
            R_h1 = -gamma*fd.cos(sigma*t)
            Rt_1 = gamma*sigma*fd.sin(sigma*t)

    elif start_wavemaker  == 2:
        if settings == 1:
                R_h1 = -gamma *(np.exp(-1j * sigma *t)).real
                Rt_1 = gamma * ((1j * sigma) * np.exp(-1j * sigma *t)).real

```

```
                if t >= t_stop:
                    R_h1 = -gamma *(np.exp(-1j * sigma * t_stop)).real
                    Rt_1 = 0 *  gamma * ((1j * sigma) * np.exp(-1j * sigma *
    t_stop)).real
        elif settings == 2:
                R_h1 = -gamma*fd.cos(sigma*t)
                Rt_1 = gamma*sigma*fd.sin(sigma*t)

                if t >= t_stop:
                    R_h1 = -gamma*fd.cos(sigma*t_stop)
                    Rt_1 = 0*gamma*sigma*fd.sin(sigma*t_stop)
    else:
        R_h1 = fd.Constant(0)
        Rt_1 = fd.Constant(0)

    t+=dt
    Rt1.append(Rt_1)
    Rh1.append(R_h1)


if start_wavemaker == 1:
    Amp_wave = max(Rh1)
    print('Maximum amplitude of wavemaker =', Amp_wave)
    vel_wave = max(Rt1)
    print('Maximum velocity of wavemaker =', vel_wave)
else:
    pass

fig, (ax1, ax2) = plt.subplots(2)

ax1.set_title('Wavemaker Position',fontsize=tsize)
ax1.plot(nnt, Rh1, 'r-', label = f'$h_e: t = {t:.3f}$ ')
ax1.set_ylabel('$R(t)[m]$ ',fontsize=size)
ax1.grid()

ax2.set_title('Wavemaker velocity',fontsize=tsize)
ax2.plot(nnt, Rt1,  'r-', label = f'$\phi_e: t = {t:.3f}$ ')
ax2.set_xlabel('$Time [s]$ ',fontsize=size)
ax2.set_ylabel('$R_{t} [m/s]$ ',fontsize=size)
```

```
ax2.grid()
##_________________  FIGURE SETTINGS ___________________________##
print('Figure settings')

fig, (ax1, ax2) = plt.subplots(2)
ax1.set_title('Initial Conditions',fontsize=tsize)
ax1.set_title(r'$h $ value in $x$ direction',fontsize=tsize)
ax1.set_ylabel(r'$h(x,t)\times 10^{-2} [m]$ ',fontsize=size)
ax1.grid()
ax2.set_xlabel(r'$x [m]$ ',fontsize=size)
ax2.set_ylabel(r'$\phi (x,t)\times 10^{-2} $ ',fontsize=size)
ax2.grid()
```

Step 5: define the variational problem

The Störmer-Verlet based time-discrete version of the transformed variational principle is given as:

$$
\begin{aligned}
0 =& \delta \int_0^L \Big[-2h^n \Big(\frac{W^{n+1/2}\phi^{n+1/2} - W^n\phi^n}{\Delta t} \Big) - 2h^{n+1} \Big(\frac{W^{n+1}\phi^{n+1} - W^{n+1/2}\phi^{n+1/2}}{\Delta t} \Big) \\
&+ X\tilde{R}_\tau^{n+1/2}(h_\xi^{n+1} + h_\xi^n)\phi^{n+1/2} - \frac{1}{2}\frac{L_w^2}{W^{n+1/2}}(\phi_\xi^{n+1/2})^2(h^{n+1} + h^n) \\
&- gW^{n+1/2}\Big(\frac{1}{2}((h^{n+1})^2 + (h^n)^2) + H_0(h^{n+1} + h^n) \Big) \Big] \, \mathrm{d}\xi \\
&- L_w R_\tau^{n+1/2}\phi^{n+1/2}(h^{n+1} + h^n)\Big|_{\xi=0},
\end{aligned}
\tag{7.12}
$$

where $X = \xi - L_w$, and $W = L_w - R(\tau)\Theta(L_w - \xi)$. In the code, `W` is W^n, `W_half` is $W^{n+1/2}$, `W_new` is W^{n+1}, `h` is h^n, `h_new` is h^{n+1}, `phi` is ϕ^n, `phi_new` is ϕ^{n+1}, `phi_half` is $\phi^{n+1/2}$, `Rt_half` is $R_\tau^{n+1/2}$, and `h_new.dx(0)` is h_ξ^{n+1}.

```
########################  VARIATIONAL PRINCIPLE  ##############################
print("#########################################################################")
print('########################## Numerical Calculations  ##################')
print("#########################################################################")

t = 0

if case == 1:
    print('#############################################################')
```

```
10    print("You have selected case 1 : Non_Linear SWE VP with piston wavemaker
      solved by firedrake by using fd.derivative ")
11    print(" Time discrete VP is based on Symplectic-Euler scheme  ")
12    print('############################################################')
13    E2_t = []
14    E2_k = []
15    E2_p = []
16
17    pot_ener = Lx * Ly * 9.8 * 0.5
18    x = fd.SpatialCoordinate(mesh)
19    x_coord = fd.Function(V).interpolate(x[0])
20
21
22    ##################################### VP ##################################
23
24
25    VP =( - 2  * h     * ( (W_half*phi_half) - (W*phi) )/dt \
26          - 2  * h_new * ( (W_new*phi_new) - (W_half*phi_half) )/dt \
27          + X * Rht_half * ( h_new.dx(0) + h.dx(0)) * phi_half \
28          - (1/2 * (Lw**2/ W_half) * fd.inner(fd.grad(phi_half), fd.grad(
      phi_half)) * h_new\
29            + 1/2 * (Lw**2/ W_half) * fd.inner(fd.grad(phi_half), fd.grad(
      phi_half)) * h) \
30          - (1/2 * g  * (Lw - Rh_half) * (fd.inner(h_new,h_new)  + fd.inner(h,h
      ) )  ) \
31              + g  * H0 * (Lw - Rh_half) * (h_new + h) )* fd.dx \
32          - (Lw * Rt_half * phi_half* (h_new + h) ) *fd.ds(1)
33
34
35    ###########################################################################
36    phi_half_expr = fd.derivative(VP, h ,v)   # derivative of VP wrt h^n to get
       the expression for phi^n+1/2
37    h_expr = fd.derivative(VP, phi_half , v)  # derivative of VP wrt phi^n+1/2
      to get the expression for h^n+1
38    phi_expr = fd.derivative(VP, h_new , v)  # derivative of VP wrt h^n+1 to
      get the value of phi^n+1
39
40
```

```
41    phi_half_expr  = fd.NonlinearVariationalSolver(fd.
      NonlinearVariationalProblem( phi_half_expr, phi_half))
42    h_expr  = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(
      h_expr, h_new))
43    phi_expr = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(
      phi_expr, phi_new))
44
45
46    ###_________________  OUTPUT FILES ______________###
47    if start_wavemaker ==1:
48            outfile_phi = fd.File("results_SV_NLSWE_wm1_case2/phi.pvd")
49            outfile_eta = fd.File("results_SV_NLSWE_wm1_case2/eta.pvd")
50    elif start_wavemaker == 2:
51            outfile_phi = fd.File("results_SV_NLSWE_wm2_case2/phi.pvd")
52            outfile_eta = fd.File("results_SV_NLSWE_wm2_case2/eta.pvd")
53    elif start_wavemaker == 0:
54            outfile_phi = fd.File("results_SV_NonLinSWE_wm0_case2/phi.pvd")
55            outfile_eta = fd.File("results_SV_NonLinSWE_wm0_case2/eta.pvd")
56    ###_________________  TXT FILES ______________###
57    if start_wavemaker == 1:
58        filename1 = "NLSWE_SV_wm1.txt"
59        filename2 = "eta_NLSWE_SV_wm1.txt"
60        filename3 = "phi_NLSWE_SV_wm1.txt"
61    elif start_wavemaker == 2:
62        filename1 = "NLSWE_SV_wm2.txt"
63        filename2 = "eta_NLSWE_SV_wm2.txt"
64        filename3 = "phi_NLSWE_SV_wm2.txt"
65    elif start_wavemaker == 0:
66        filename1 = "NLSWE_SV_wm0.txt"
67        filename2 = "eta_NLSWE_SV_wm0.txt"
68        filename3 = "phi_NLSWE_SV_wm0.txt"
69
70    f = open(filename1 , 'w+')
```

Step 6: solve the variational problem

```
1     ######### TIME LOOP #############
2
3     while (t <= t_end):
4         tt = format(t, '.3f')
```

```
        print('t =', t)

        t_new = t + dt
        t_half = t + dt/2

        X.interpolate( x_coord - Lw)
        W.interpolate(Lw - Rh)
        W_new.interpolate(Lw - Rh_new)
        W_half.interpolate(Lw - Rh_half)
    ## ______________________  wavemaker motion  ________________________ ##
        if start_wavemaker == 1:
                        R.assign(-gamma  * fd.cos(sigma*t))
                        Rt.assign( gamma * sigma * fd.sin(sigma*t))

                        Rt_half.assign( gamma * sigma * fd.sin(sigma*t_half))

                        Rh.interpolate(fd.conditional(fd.le(x_coord,Lw), -gamma
      * fd.cos(sigma * t), 0.0) )
                        Rh_half.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_half), 0.0) )
                        Rh_new.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_new), 0.0) )

                        Rht_half.interpolate(fd.conditional(fd.le(x_coord,Lw),
    gamma  * sigma * fd.sin(sigma*t_half),0.0))

             # # wavemaker moves at first and then stops after some time
        if start_wavemaker == 2:
                        R.assign(-gamma  * fd.cos(sigma*t))
                        Rt.assign( gamma * sigma * fd.sin(sigma*t))

                        Rt_half.assign( gamma * sigma * fd.sin(sigma*t_half))

                        Rh.interpolate(fd.conditional(fd.le(x_coord,Lw), -gamma
      * fd.cos(sigma * t), 0.0) )
                        Rh_half.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_half), 0.0) )
                        Rh_new.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_new), 0.0) )
```

```
37
38                      Rht_half.interpolate(fd.conditional(fd.le(x_coord,Lw),
    gamma  * sigma * fd.sin(sigma* t_half),0.0))
39
40                      if t >= t_stop:
41                          R.assign(-gamma  * fd.cos(sigma*t_stop))
42                          Rt.assign(0)
43
44                          Rt_half.assign(0)
45
46                          Rh.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_stop), 0.0) )
47                          Rh_half.interpolate(fd.conditional(fd.le(x_coord,Lw
    ), -gamma  * fd.cos(sigma * (t_stop + dt/2)), 0.0) )
48                          Rh_new.interpolate(fd.conditional(fd.le(x_coord,Lw)
    , -gamma  * fd.cos(sigma * (t_stop + dt)), 0.0) )
49                          Rht_half.assign(0)
50
51              # # wavemaker does not move at all
52          elif start_wavemaker == 0:
53                      R.assign(0)
54                      Rt.assign(0)
55
56                      Rt_half.assign(0)
57                      Rh.assign(0)
58                      Rh_half.assign(0)
59                      Rh_new.assign(0)
60                      Rht_half.assign(0)
61
62      ## ------------------------------------------------------------------ ##
63
64          phi_half_expr.solve()
65          h_expr.solve()
66          phi_expr.solve()
67
68          t+= dt
69
70          Epp1 = fd.assemble( ( 1/2 * g * fd.inner(h,h))* ((Lw - Rh)/Lw) * fd.dx)
71          Epp2 = fd.assemble( ( g *h * H0)* ((Lw - Rh)/Lw) * fd.dx )
```

```
72
73        Epp = fd.assemble( (Lw - Rh)*( g*h*(0.5*h - H0) )* fd.dx  )
74        Ekk = fd.assemble(0.5 * (Lw**2/(Lw - Rh)) * h * fd.inner(fd.grad(phi),
    fd.grad(phi)) * fd.dx )
75
76
77        Et = abs(Ekk) + abs(Epp)
```

Step 7: Plot and output the results

Finally, within the time loop, we compute and write the value of variables in *.txt* files and generate *.pvd* files for visualisation.

```
1
2         f.write('%-25s %-25s %-25s %-25s %-25s %-25s %-25s %-25s %-25s %-25s\n'
     \
3                 % (str(t), str(R.dat.data[2]), str(Rt.dat.data[2]), str(phi.at
    (0,0)),\
4                     str(h.at(0,0)), str(Epp), str(Ekk), str(Et) , str(Epp1),
    str(Epp2) ) )
5                 # % (str(t), str(R.dat.data[2]), str(Rt.dat.data[2]), str(phi.
    at(0,0)), str(h.at(0,0)), str(4410.367500000001 - Epp), str(Ekk), str(Et) )
    )
6
7         if (t in t_plot):
8             print('Plotting starts')
9             print('t =', t)
10            i += 1
11            if ic == 1:
12                phi_exact = phie.interpolate( (U_0.real)/P * a.real* fd.cos(kp
    * (x_coord - Lx)) \
13                                + (g/w) * fd.cos(k1 * x_coord) * ( -A0*fd.sin(w *
     t) + B0*fd.cos(w * t) ) )
14
15                h_exact = he.interpolate( (((1j*sigma)/ g) * (U_0.real)/(P) *
    np.exp(-1j * sigma * t_end)).real * fd.cos(kp * (x_coord - Lx))\
16                                    +  fd.cos(k1 * x_coord) * ( A0*fd.cos(w * t)
    + B0*fd.sin(w * t) ) + H0 )
17
18                phievals = np.array([phi_exact.at(x, yslice) for x in xvals])
```

```
                etaevals = np.array([h_exact.at(x, yslice) for x in xvals])

            else:
                pass

            phi1vals = np.array([phi_new.at(x, Ly/2) for x in xvals])
            h1vals = np.array([h_new.at(x, Ly/2) for x in xvals])

            if start_wavemaker == 1:
                h_file_name = 'h_SV_nlswe_wm1_'+tt+'.txt'
                phi_file_name = 'phi_SV_nlswe_wm1_'+tt+'.txt'
            elif start_wavemaker == 2:
                 h_file_name = 'h_SV_nlswe_wm2_'+tt+'.txt'
                 phi_file_name = 'phi_SV_nlswe_wm2_'+tt+'.txt'
            elif start_wavemaker == 0:
                 h_file_name = 'h_SV_nlswe_wm0_'+tt+'.txt'
                 phi_file_name = 'phi_SV_nlswe_wm0_'+tt+'.txt'

            h_file = open(os.path.join(save_path, h_file_name), 'w')
            phi_file = open(os.path.join(save_path, phi_file_name), 'w')

            y_slice = Ly/2
            x_coarse = np.linspace(0,Lx-0.001,200)
            for ix in x_coarse:
                h_file.write('%-25s  %-25s\n' %(str(ix), str(h.at(ix,y_slice)))
)
                phi_file.write('%-25s  %-25s\n' %(str(ix), str(phi.at(ix,
y_slice))))

            ax1.plot(xvals, h1vals * (10 ** factor), color[i-1],label = f' $\
eta_n: t = {t:.3f}$')
            ax2.plot(xvals,phi1vals, color[i-1], label = f' $\phi_n: t = {t:.3f
}$')


            if ic == 1:
                ax1.plot(xvals, etaevals* (10 ** factor), colore[i-1], label =
f'$h_e: t = {t:.3f}$ ')
```

```
52                ax2.plot(xvals, phievals,  colore[i-1], label = f'$\phi_e: t =
    {t:.3f}$ ')
53            else:
54                pass
55            ax1.legend(loc=4)
56            ax2.legend(loc=4)
57
58
59        outfile_eta.write( h_new )
60        outfile_phi.write( phi_new )
61
62        # set-up next time-step
63        E2_t.append(abs(Et))
64        E2_p.append(abs(Epp))
65        E2_k.append(abs(Ekk))
66
67        phi.assign(phi_new)
68        h.assign(h_new)
69
70    f.close()
71    h_file.close()
72    phi_file.close()
73
74    fig, (ax1, ax2, ax3) = plt.subplots(3)
75    fig.suptitle('Energy evolution with time',fontsize= tsize)
76    ax1.plot(time, E2_k)
77    ax1.set_ylabel('Kinetic energy[J] ',fontsize=size)
78    ax1.grid()
79
80    ax2.plot(time, E2_p)
81    ax2.set_ylabel('Potential Energy [J]',fontsize=size)
82    ax2.grid()
83
84    ax3.plot(time, E2_t)
85    ax3.set_xlabel('$Time [s]$ ',fontsize=size)
86    ax3.set_ylabel('Total energy [J] ',fontsize=size)
87    ax3.grid()
```

Case 2: Non-Linear SWE VP with piston wavemaker solved by firedrake by using time-discrete weak formulations.

The time-discrete weak formulations based on the second order Störmer-Verlet time stepping scheme are as follows

$$\begin{aligned}\int_0^L 2\Big(\frac{W^{n+1/2}\phi^{n+1/2}-W^n\phi^n}{\Delta t}\Big)\delta h^n\,\mathrm{d}\xi = \int_0^L & X\tilde{R}_\tau^{n+1/2}\phi^{n+1/2}\partial_\xi(\delta h^n)\\ & -\frac{1}{2}\frac{L_w^2}{W^{n+1/2}}(\phi_\xi^{n+1/2})^2\delta h^n\\ & -gW^{n+1/2}(h^n-H_0)\delta h^n\,\mathrm{d}\xi\\ & -L_wR_\tau^{n+1/2}\phi^{n+1/2}\delta h^n\Big|_{\xi=0};\end{aligned}\tag{7.13}$$

$$\begin{aligned}\int_0^L 2W^{n+1/2}\Big(\frac{h^{n+1}-h^n}{\Delta t}\Big)\delta\phi^{n+1/2}\,\mathrm{d}\xi = \int_0^L & -X\tilde{R}_\tau^{n+1/2}(h_\xi^{n+1}+h_\xi^n)\delta\phi^{n+1/2}\\ & +\frac{L_w^2}{W^{n+1/2}}(h^{n+1}+h^n)\phi_\xi^{n+1/2}\,\partial_\xi(\delta\phi^{n+1/2})\,\mathrm{d}\xi\\ & +L_wR_\tau^{n+1/2}(h^{n+1}+h^n)\delta\phi^{n+1/2}\Big|_{\xi=0};\end{aligned}\tag{7.14}$$

$$\begin{aligned}\int_0^L 2\Big(\frac{W^{n+1}\phi^{n+1}-W^{n+1/2}\phi^{n+1/2}}{\Delta t}\Big)\delta h^{n+1}\,\mathrm{d}\xi = \int_0^L & X\tilde{R}_\tau^{n+1/2}\phi^{n+1/2}\partial_\xi(\delta h^{n+1})\\ & -\frac{1}{2}\frac{L_w^2}{W^{n+1/2}}(\phi_\xi^{n+1/2})^2\delta h^{n+1}\\ & -gW^{n+1/2}(h^{n+1}-H_0)\delta h^{n+1}\,\mathrm{d}\xi\\ & -L_wR_\tau^{n+1/2}\phi^{n+1/2}\delta h^{n+1}\Big|_{\xi=0}.\end{aligned}\tag{7.15}$$

```
1
2  elif case == 2:
3
4      print('############################################################')
5      print("You have selected case 2 : Non_Linear SWE VP with piston wavemaker
       solved by firedrake by using time-discrete weak formulations. ")
6      print("Time discrete weak formulations based on SV scheme.  ")
7      print('############################################################')
8      E2_t = []
9      E2_k = []
10     E2_p = []
11
12     pot_ener = Lx * Ly * 9.8 * 0.5
13     DBC = fd.DirichletBC( V, Rht_half , wavemaker_id)
```

```
14    x = fd.SpatialCoordinate(mesh)
15    y = 0
16    x_coord = fd.Function(V).interpolate(x[0])
17
18
19    ################################### VP ################################
20
21    VP =( - 2  * h     * ( (W_half*phi_half) - (W*phi) )/dt \
22          - 2  * h_new * ( (W_new*phi_new) - (W_half*phi_half) )/dt \
23          + X * Rht_half * ( h_new.dx(0) + h.dx(0)) * phi_half \
24          - (1/2 * (Lw**2/ W_half) * fd.inner(fd.grad(phi_half), fd.grad(
    phi_half)) * h_new\
25            + 1/2 * (Lw**2/ W_half) * fd.inner(fd.grad(phi_half), fd.grad(
    phi_half)) * h) \
26          - (1/2 * g  * (Lw - Rh_half) * (fd.inner(h_new,h_new)  + fd.inner(h,h
    ) )  ) \
27              + g  * H0 * (Lw - Rh_half) * (h_new + h) )* fd.dx \
28          - (Lw * Rt_half * phi_half* (h_new + h) ) *fd.ds(1)
29
30    phi_half_expr = ( -2 * fd.inner( (W_half*phi_half - W*phi )/dt, v )\
31                      + ( X * Rht_half * phi_half * v.dx(0) ) \
32                      - 0.5* (Lw**2/W_half) * (phi_half.dx(0))**2 * v  \
33                      - g * W_half * (h - H0)* v  )* fd.dx \
34                      - ( Lw * Rt_half * phi_half * v )*fd.ds(1)
35
36
37    h_expr = (+2* fd.inner( W_half* (h_new - h)/dt, v) \
38              + ( X * Rht_half * fd.inner( h_new.dx(0) + h.dx(0), v)) \
39              - (Lw**2/W_half) * phi_half.dx(0) * v.dx(0) * (h_new + h) )*fd.dx
    \
40              - (Lw * Rt_half * (h_new + h) * v ) * fd.ds(1)
41
42    phi_new_expr = ( - 2 * fd.inner( ( (W_new*phi_new) - (W_half*phi_half) )/dt
    , v ) \
43                   + ( X * Rht_half * phi_half * v.dx(0) )\
44                     - 0.5 * (Lw**2/(W_half)) * (phi_half.dx(0))**2 * v \
45                     - g*(W_half)*(h_new - H0)* v)*fd.dx\
46                     - (Lw * Rt_half * phi_half * v) * fd.ds(1)
47
```

```
48    phi_half_expr  = fd.NonlinearVariationalSolver(fd.
      NonlinearVariationalProblem( phi_half_expr, phi_half))
49    h_expr  = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(
      h_expr, h_new))
50    phi_new_expr = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem
      (phi_new_expr, phi_new))
51
52    ###_________________  OUTPUT FILES ______________###
53    if start_wavemaker ==1:
54            outfile_phi = fd.File("results_SV2_NLSWE_wm1/phi.pvd")
55            outfile_eta = fd.File("results_SV2_NLSWE_wm1/eta.pvd")
56    elif start_wavemaker == 2:
57            outfile_phi = fd.File("results_SV2_NLSWE_wm2_case2/phi.pvd")
58            outfile_eta = fd.File("results_SV2_NLSWE_wm2_case2/eta.pvd")
59    elif start_wavemaker == 0:
60            outfile_phi = fd.File("results_SV2_NLSWE_wm0/phi.pvd")
61            outfile_eta = fd.File("results_SV2_NLSWE_wm0/eta.pvd")
62    ###_________________  TXT FILES ______________###
63    if start_wavemaker == 1:
64        filename1 = "NLSWE_SV2_wm1.txt"
65        filename2 = "eta_NLSWE_SV2_wm1.txt"
66        filename3 = "phi_NLSWE_SV2_wm1.txt"
67    elif start_wavemaker == 2:
68        filename1 = "NLSWE_SV2_wm2.txt"
69        filename2 = "eta_NLSWE_SV2_wm2.txt"
70        filename3 = "phi_NLSWE_SV2_wm2.txt"
71    elif start_wavemaker == 0:
72        filename1 = "NLSWE_SV2_wm0.txt"
73        filename2 = "eta_NLSWE_SV2_wm0.txt"
74        filename3 = "phi_NLSWE_SV2_wm0.txt"
75
76    f = open(filename1 , 'w+')
77    ######### TIME LOOP #############
78
79    while (t <= t_end):
80        tt = format(t, '.3f')
81        t_new = t + dt
82        t_half = t + dt/2
83
```

```
84        X.interpolate( x_coord - Lw)
85        W.interpolate(Lw - Rh)
86        W_new.interpolate(Lw - Rh_new)
87        W_half.interpolate(Lw - Rh_half)
88    ## ----------------------  wavemaker motion  ------------------------- ##
89
90        if start_wavemaker == 1:
91                        R.assign(-gamma  * fd.cos(sigma*t))
92                        Rt.assign( gamma * sigma * fd.sin(sigma*t))
93
94                        Rt_half.assign( gamma * sigma * fd.sin(sigma*t_half))
95
96                        Rh.interpolate(fd.conditional(fd.le(x_coord,Lw), -gamma
      * fd.cos(sigma * t), 0.0) )
97                        Rh_half.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_half), 0.0) )
98                        Rh_new.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_new), 0.0) )
99
100                       Rht_half.interpolate(fd.conditional(fd.le(x_coord,Lw),
    gamma  * sigma * fd.sin(sigma*t_half),0.0))
101
102         # # wavemaker moves at first and then stops after some time
103       if start_wavemaker == 2:
104                       R.assign(-gamma  * fd.cos(sigma*t))
105                       Rt.assign( gamma * sigma * fd.sin(sigma*t))
106
107                       Rt_half.assign( gamma * sigma * fd.sin(sigma*t_half))
108
109                       Rh.interpolate(fd.conditional(fd.le(x_coord,Lw), -gamma
      * fd.cos(sigma * t), 0.0) )
110                       Rh_half.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_half), 0.0) )
111                       Rh_new.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_new), 0.0) )
112
113                       Rht_half.interpolate(fd.conditional(fd.le(x_coord,Lw),
    gamma  * sigma * fd.sin(sigma* t_half),0.0))
114
```

```
115                         if t >= t_stop:
116                             R.assign(-gamma  * fd.cos(sigma*t_stop))
117                             Rt.assign(0)
118
119                             Rt_half.assign(0)
120
121                             Rh.interpolate(fd.conditional(fd.le(x_coord,Lw), -
    gamma  * fd.cos(sigma * t_stop), 0.0) )
122                             Rh_half.interpolate(fd.conditional(fd.le(x_coord,Lw
    ), -gamma  * fd.cos(sigma * (t_stop + dt/2)), 0.0) )
123                             Rh_new.interpolate(fd.conditional(fd.le(x_coord,Lw)
    , -gamma  * fd.cos(sigma * (t_stop + dt)), 0.0) )
124
125                             Rht_half.assign(0)
126
127             # # wavemaker does not move at all
128         elif start_wavemaker == 0:
129                         R.assign(0)
130                         Rt.assign(0)
131                         Rt_half.assign(0)
132                         Rh.assign(0)
133                         Rh_half.assign(0)
134                         Rh_new.assign(0)
135                         Rht_half.assign(0)
136
137     ## ------------------------------------------------------------------ ##
138         phi_half_expr.solve()
139         h_expr.solve()
140         phi_new_expr.solve()
141
142         t+= dt
143
144         Epp1 = fd.assemble( ( 1/2 * g * fd.inner(h,h))* ((Lw - Rh)/Lw) * fd.dx)
145         Epp2 = fd.assemble( ( g *h * H0)* ((Lw - Rh)/Lw) * fd.dx )
146
147         Epp = fd.assemble( (Lw - Rh)*( g*h*(0.5*h - H0) )* fd.dx  )
148         Ekk = fd.assemble(0.5 * (Lw**2/(Lw - Rh)) * h * fd.inner(fd.grad(phi),
    fd.grad(phi)) * fd.dx )
149
```

```
150         Et = abs(Ekk) + abs(Epp)
151
152         f.write('%-25s %-25s %-25s %-25s %-25s %-25s %-25s %-25s %-25s\n'
     \
153                 % (str(t), str(R.dat.data[2]), str(Rt.dat.data[2]), str(phi.at
    (0,0)),\
154                    str(h.at(0,0)), str(Epp), str(Ekk), str(Et) , str(Epp1), str
    (Epp2) ) )
155
156         if (t in t_plot):
157             i+=1
158             print('Plotting starts')
159             print('t =', t)
160             if ic == 1:
161                 phi_exact = phie.interpolate( (U_0.real)/P * a.real* fd.cos(kp
    * (x_coord - Lx)) \
162                                 + (g/w) * fd.cos(k1 * x_coord) * ( -A0*fd.sin(w *
     t) + B0*fd.cos(w * t) ) )
163
164                 h_exact = he.interpolate( (((1j*sigma)/ g) * (U_0.real)/(P) *
    np.exp(-1j * sigma * t_end)).real * fd.cos(kp * (x_coord - Lx))\
165                                     +  fd.cos(k1 * x_coord) * ( A0*fd.cos(w * t)
    + B0*fd.sin(w * t) ) + H0 )
166
167                 phievals = np.array([phi_exact.at(x, yslice) for x in xvals])
168                 etaevals = np.array([h_exact.at(x, yslice) for x in xvals])
169
170             else:
171                 pass
172
173             phi1vals = np.array([phi_new.at(x, Ly/2) for x in xvals])
174             h1vals = np.array([h_new.at(x, Ly/2) for x in xvals])
175
176             if start_wavemaker == 1:
177                 h_file_name = 'h_SV2_nlswe_wm1_'+tt+'.txt'
178                 phi_file_name = 'phi_SV2_nlswe_wm1_'+tt+'.txt'
179             elif start_wavemaker == 2:
180                  h_file_name = 'h_SV2_nlswe_wm2_'+tt+'.txt'
181                  phi_file_name = 'phi_SV2_nlswe_wm2_'+tt+'.txt'
```

```
182             elif start_wavemaker == 0:
183                  h_file_name = 'h_SV2_nlswe_wm0_'+tt+'.txt'
184                  phi_file_name = 'phi_SV2_nlswe_wm0_'+tt+'.txt'
185 
186             h_file = open(os.path.join(save_path, h_file_name), 'w')
187             phi_file = open(os.path.join(save_path, phi_file_name), 'w')
188 
189             y_slice = Ly/2
190             x_coarse = np.linspace(0,Lx-0.001,200)
191             for ix in x_coarse:
192                 h_file.write('%-25s  %-25s\n' %(str(ix), str(h.at(ix,y_slice)))
    )
193                 phi_file.write('%-25s  %-25s\n' %(str(ix), str(phi.at(ix,
    y_slice))))
194 
195             ax1.plot(xvals, h1vals * (10 ** factor), color[i-1],label = f' $\
    eta_n: t = {t:.3f}$')
196             ax2.plot(xvals,phi1vals, color[i-1], label = f' $\phi_n: t = {t:.3f
    }$')
197 
198             if ic == 1:
199                 ax1.plot(xvals, etaevals* (10 ** factor), colore[i-1], label =
    f'$h_e: t = {t:.3f}$ ')
200                 ax2.plot(xvals, phievals,  colore[i-1], label = f'$\phi_e: t =
    {t:.3f}$ ')
201             else:
202                 pass
203             ax1.legend(loc=4)
204             ax2.legend(loc=4)
205 
206         outfile_eta.write( h_new )
207         outfile_phi.write( phi_new )
208 
209         # set-up next time-step
210         E2_t.append(abs(Et))
211         E2_p.append(abs(Epp))
212         E2_k.append(abs(Ekk))
213 
214         phi.assign(phi_new)
```

```
215         h.assign(h_new)
216
217     f.close()
218     h_file.close()
219     phi_file.close()
220
221     fig, (ax1, ax2, ax3) = plt.subplots(3)
222     fig.suptitle('Energy evolution with time',fontsize= tsize)
223     ax1.plot(time, E2_k)
224     ax1.set_ylabel('Kinetic energy[J] ',fontsize=size)
225     ax1.grid()
226
227     ax2.plot(time, E2_p)
228     ax2.set_ylabel('Potential Energy [J]',fontsize=size)
229     ax2.grid()
230
231     ax3.plot(time, E2_t)
232     ax3.set_xlabel('$Time [s]$ ',fontsize=size)
233     ax3.set_ylabel('Total energy [J] ',fontsize=size)
234     ax3.grid()
235
236 else:
237     print(" The selected number does not match any case")
238
239 plt.show()
240 print('*************** PROGRAM ENDS ******************')
```

The complete code is publicly available as **NL_SWE_SV.py** on the GitHub repository.

7.3 Piston-driven numerical wavetank based on potential flow equations

In this section, we share and explain the code developed and implemented for simulating the nonlinear potential flow equations with and without the piston wavemaker. This code solves the problem by using the novel approach that we have developed in Chapter 3 of this thesis, i.e. time-discrete variational principle based on the first-order symplectic-Euler and second-order Störmer-Verlet (SV) time-integration schemes. As we have already explained the seven-step

process of solving the variational problem in Firedrake and the current code is based on the extension of the numerical wavetank model based on shallow-water equations, hence, we will only discuss the steps which are different from shallow water equations case and share a weblink of the repository at the end of this section where the complete code for all the test cases is available.

User defined parameters

First, we will describe the user-defined parameters. The user can choose the dimensions of the two-dimensional wavetank by changing the value of variables `Lx` and `Lz` which denote the wavetank length in x and z directions. Furthermore, the user can change the spatial discretisation of the domain by changing the number of nodes in x and z directions, denoted by `nx` and `nz` respectively. Lastly, the user has the choice to select one of the four test cases by assigning an appropriate value to the variable ‘‘`nvpcase"`, as follows:

1. `"nvpcase"` = 1. If the user assigns 1 value to `"nvpcase"` then the equations for the linear potential-flow case without wavemaker will be solved by using the time-discrete variational principle.

2. `"nvpcase"` = 2. If the user assigns 2 to `"nvpcase"` then the equations for the non-linear potential-flow case without wavemaker will be solved by using the time-discrete variational principle based on the first-order symplectic-Euler scheme.

3. `"nvpcase"` = 21. If the user assigns 21 to `"nvpcase"` then the equations for the non-linear potential-flow case with piston wavemaker will be solved by using the time-discrete variational principle. The time-discretisation is based on the first-order symplectic-Euler scheme.

4. `"nvpcase"` = 22. If the user assigns 22 to `"nvpcase"` then the equations for the non-linear potential-flow case with piston wavemaker will be solved by using the time-discrete variational principle. The time-discretisation is based on the second-order Störmer-Verlet (SV) scheme.

The user also finds a variable called `"top_id"`, we suggest not to change this variable because it denotes the top-edge or free-surface of the rectangular numerical wavetank. We use this variable later to define the Dirichlet boundary condition to solve the Laplace equation. The code block

for the user parameters is shown as follows:

```
# parameters in SI units
g = 9.81  # gravitational acceleration [m/s^2]

# water
Lx = 140 # length of the tank [m] in x-direction
Lz = 1
nx = 120 # no. of nodes in x-direction
nz = 6   # no. of nodes in z-direction
nCG = 2  # function space order horizontal
nCGvert = 6 # function space order vertical
H0 = Lz     # rest water depth [m]

# control parameters
output_data_every_x_time_steps = 20  # to avoid saving data every time step

top_id = 'top'

nvpcase = 1 # case 1 (SE linear), 2 (SE nonlinear), 21 (SE nonlinear piston
    wavemaker), 22 (SV nonlinear piston wavemaker)
```

Step 1: define computational domain and mesh

Now, we will define the creation of the computational domain and mesh to solve the variation problem. In this case, we use a special mesh known as extruded mesh. This mesh type allows us to use different function spaces in different regions of the computational domain. Therefore, a two-dimensional computational domain is created by extruding the horizontal line (x direction) along the fluid depth, i.e. z direction.

```
#________________________ MESH  ________________________#
# Extruded mesh; see example:
# https://www.firedrakeproject.org/demos/extruded_continuity.py.html
# https://www.firedrakeproject.org/extruded-meshes.html
# CG x R for surface eta and phi
# CG x CG for interior phi or varphi


mesh1d = fd.IntervalMesh(nx, Lx)
```

```
mesh = fd.ExtrudedMesh(mesh1d, nz, layer_height=Lz/nz, extrusion_type='uniform')
```

Step 2: define function spaces

The function space to numerically solve the equations is the first-order piece-wise-linear continuous Galerkin (CG); and it is created in a way that the function space of the free-surface velocity potential ($\psi(x, H_0, t)$) and the velocity potential of the inner domain ($\varphi(x, z, t)$) can be differentiated. For this purpose, Firedrake allows the creation of a mixed-function space (`mixed_V = V_R * V_W`) that can extract the $\phi_f(x, H_0, t)$ and $\varphi(x, z, t)$ from the total velocity potential $\phi(x, z, t)$ by using split command on the mixed function space. The function space for the functions at the free-surface, i.e. velocity potential at free-surface and free-surface elevation, is created by expression `V_R = fd.FunctionSpace(mesh, 'CG', nCG, vfamily='R', vdegree=0)`, this expression creates a matrix equal for the complete domain size but assigns zero values to the nodes corresponding to the inner domain.

```
#__________________  Define function spaces  __________________#

# interior potential varphi; can have mix degrees in horizontal and vertical dimension
V_W = fd.FunctionSpace(mesh, 'CG', nCG, vfamily='CG', vdegree=nCGvert)

# free surface eta and surface potential phi extended uniformly in vertical: vdegree=0
V_R = fd.FunctionSpace(mesh, 'CG', nCG, vfamily='R', vdegree=0)

# Variables at the inner-domain
varphi = fd.Function(V_W, name="varphi")

# Variables at the free-surface
phi_f = fd.Function(V_R, name="phi_f")
phiii = fd.Function(V_R, name="phi")
phif_new = fd.Function(V_R, name="phi_f")

eta = fd.Function(V_R, name="eta")
eta_new = fd.Function(V_R, name="eta_new")

heta = fd.Function(V_R, name="heta")
```

```
h_new = fd.Function(V_R, name="h_new")

# Variables for Stormer-Verlet waves
mixed_V = V_R * V_W
result_mixed = fd.Function(mixed_V)
phii, varphii = fd.split(result_mixed)
```

Step 3: define test functions

The test functions corresponding to the unknown functions at the free surface and inner domain are `v_R = fd.TestFunction(V_R)`, and `v_W = fd.TestFunction(V_W)` respectively. Moreover, we also need a test function corresponding to the mixed function space which is defined as `vvp = fd.TestFunction(mixed_V)`.

```
#__________________  Define test functions  __________________#

# Test functions
v_W = fd.TestFunction(V_W)
v_R = fd.TestFunction(V_R)

# Stormer-Verlet waves
vvp = fd.TestFunction(mixed_V)
vvp0, vvp1 = fd.split(vvp)  # These represent "blocks".
```

Step 4: define initial conditions

The fourth step is to assign values to the function spaces. The user has a choice to assign one of two values to the variable `"nic"` and solve one of the following two cases:

1. `nic = 0` computes the exact standing wave solution at the initial time, i.e. `t=0`, by using the analytical expression. Then the computed values are assigned as initial conditions to the function spaces of the problem. This choice should be made when the user wants to compare the numerical results with the exact solution.

2. `nic = 1` computes the wavemaker displacement and velocity by using the predefined or user-defined functions. In this case, the value 0 is assigned to the functions spaces and wavemaker motion and velocity will dictate the functions according to the time-discrete weak formulations.

```
if nic == 0:
    print('########################   Parameters of standing-wave exact sol
    ######################')
    n_mode = 2
    kx = 2 * np.pi * n_mode / Lx
    omega = np.sqrt(gg * kx * np.tanh(kx * Lz))
    A = 0.2*4
    D = -gg*A/(omega*np.cosh(kx*H0))
    Tperiod = 2*np.pi/omega
    print('Period: ', Tperiod)
    phi_exact_expr = D * fd.cos(kx * x[0]) * fd.cosh(kx * x[1]) * np.sin(omega
    * t0) # D cos(kx*x) cosh(kx*z) cos(omega t)
    phi_exact_exprH0 = D * fd.cos(kx * x[0]) * fd.cosh(kx * H0) * np.sin(omega
    * t0) # D cos(kx*x) cosh(kx*z) cos(omega t)
    eta_exact_expr = A * fd.cos(kx * x[0]) * np.cos(omega * t0)
    t_end = Tperiod  # time of simulation [s]
    dtt = np.minimum(Lx/nx,Lz/nz)/(np.pi*np.sqrt(gg*H0)) # i.e. dx/max(c0) with
     c0 =sqrt(g*H0)
    Nt = 500
    CFL = 0.5
    dt = CFL*Tperiod/Nt   # time step [s]
    print('dtt=',dtt, t_end/dtt,dt,2/omega)

    ##_____________  To get results at different time steps _____________##
    while (t <= t_end+dt):
        time.append(t)
        t+= dt
    nplot = 4

elif nic == 1:

    print('########################   PARAMETERS  of Wavemaker
    ######################')

    lambd = 70
    n_mode = Lx/lambd #
    print('n_mode',n_mode)
    kx = 2 * np.pi * n_mode / Lx
    omega = np.sqrt(gg * kx * np.tanh(kx * Lz))
```

```
35    Tperiod = 2.0*np.pi/omega
36
37    nTfac = 35
38    tstop = (nTfac-7)*Tperiod
39    t_end = nTfac*Tperiod # time of simulation [s]
40    Tstartmeas = 30*Tperiod
41
42    dtt = np.minimum(Lx/nx,Lz/nz)/(np.pi*np.sqrt(gg*H0)) # i.e. dx/max(c0) with
      c0 =sqrt(g*H0)
43    Nt = 500
44    CFL = 0.5
45    dt = CFL*Tperiod/Nt     # time step
46    print('dtt=',dtt, t_end/dtt,dt,2/omega)
47    D = 0.0
48    phi_exact_expr = D * x[0] * x[1]
49    phi_exact_exprH0 = D * x[0]
50    eta_exact_expr = D * x[0]
```

Step 5: define the variational problem

The code can solve four cases of the variational problem based on potential flow equations ranging from linear to nonlinear, and with and without piston wavemaker, see §7.3. Solving the variational problem based on potential-flow equations is similar to the case of the shallow-water equation. However, the two cases differ in the application of boundary conditions and the use of mixed-function spaces. To explain the use of boundary conditions and mixed function spaces, we are using the simplest case, i.e. the time-discrete variational principle of linear potential-flow equations based on the symplectic-Euler scheme without a piston wavemaker, given as:

$$
\begin{aligned}
0 = \delta \int_0^{L_x} \phi_f^{n+1} \frac{(\eta^{n+1} - \eta^n)}{\Delta t} + \phi_f^n \frac{\eta^n}{\Delta t} - \frac{1}{2} g (\eta^n)^2 \, \mathrm{d}x \\
- \int_0^{L_x} \int_0^{H_0} \frac{1}{2} |\boldsymbol{\nabla} \varphi^{n+1}|^2 + \boldsymbol{\nabla} \varphi^{n+1} \cdot \boldsymbol{\nabla} \phi_f^{n+1} + \frac{1}{2} |\boldsymbol{\nabla} \phi_f^{n+1}|^2 \, \mathrm{d}x \, \mathrm{d}z.
\end{aligned} \tag{7.16}
$$

The expression denoted by `VP11` in the code block represents the VP in (7.16). By using (7.16), we aim to complete three steps at each time step, explained as follows:

1. Step 1: compute the value of free-surface velocity potential (ϕ_f^{n+1}), where the suffix $n+1$ denotes the next time step, by using the initial conditions of free-surface velocity potential

(ϕ_f^n) and free-surface elevation (η^n), where the suffix n denotes the current time step. The time-discrete weak formulation is given as:

$$\int_0^{L_x} \Big(\frac{\phi_f^{n+1} - \phi_f^n}{\Delta t} + g\eta^n \Big) \delta\eta^n \, \mathrm{d}x = 0. \tag{7.17}$$

To obtain (7.17), we take variations of (7.16) with respect to η^n, which is accomplished by `phif_expr1 = fd.derivative(VP11, eta, du=vvp0)`, where `vvp0` represents the block for ϕ_f in the mixed test-function space.

2. Step 2: update the value of the velocity potential for the whole domain (ϕ) by using the updated value of free-surface velocity potential (ϕ_f^{n+1}) as a Dirichlet boundary condition.

$$\int_0^{L_x} \int_0^{H_0} \boldsymbol{\nabla}(\phi_f^{n+1} + \varphi^{n+1}) \cdot \boldsymbol{\nabla}\delta\phi^{n+1} \, \mathrm{d}x \, \mathrm{d}z = 0 \tag{7.18}$$

Note that the dimensions of matrices for the free-surface velocity potential (ϕ_f^{n+1}) and velocity potential for the inner domain (φ) would be different if we use standard Function spaces. Therefore, to make the dimensions of matrices consistent we use mixed-function space which contains the value of free-surface and inner domain functions simultaneously and we can use fd.split() command to extract the values later in the time loop. To accomplish this step in Firedrake, first, we take variations of (7.16) with respect to inner-domain velocity potential (φ) which is done by the expression `phi_expr1 = fd.derivative(VP11, varphii, du=vvp1)` in the code block, where `vvp1` represents the block for φ in the mixed test-function space. Then we add the `phi_expr1` to (7.17) to have an expression for the velocity potential of the whole domain, given by the expression `phi_combo` in the code. In this way, we find the updated value of free-surface velocity potential (ϕ_f^{n+1}) and update the value of inner-domain velocity potential in one step when we solve the variational problem.

3. Step 3: compute the value of free surface elevation η^{n+1} by using the value of ϕ_f^{n+1}, as follows:

$$\int_0^{L_x} \frac{(\eta^{n+1} - \eta^n)}{\Delta t} \delta\phi_f^{n+1} \, \mathrm{d}x = 0. \tag{7.19}$$

To obtain (7.19), we take variations of (7.16) with respect to ϕ_f^n, which is accomplished

by `eta_expr2 = fd.derivative(VP11, phii, du=v_R)`, where `v_R` represents the test function for ϕ_f in the function space `V_R` .

```
##_________________  Boundary Conditions ____________________________##

BC_varphi_mixed = fd.DirichletBC(mixed_V.sub(1), 0, top_id)

if nvpcase==1:
    print('#####################################################')
    print("You have selected case 1 : Linear PF VP without piston wavemaker
    solved by firedrake by using fd.derivative. ")
    print("Time discrete VP is based on Symplectic-Euler scheme.  ")
    print('#####################################################')

    ##________ Time-discrete Variational Principle
      ________##


    VP11 = ( fd.inner(phii, (eta_new - eta)/dt) + fd.inner(phi_f, eta/dt) -
    (1/2 * gg * fd.inner(eta, eta)) )* fd.ds_t \
        - ( 1/2 * fd.inner(fd.grad(phii+varphii), fd.grad(phii+varphii))  ) *
    fd.dx

    ##________  Automatic derivation of time-discrete weak formulations
    ________##

    # Step-1 and 2 must be solved in tandem: f-derivative VP wrt eta to find
    update of phi at free surface
    # int -phi/dt + phif/dt - gg*et) delta eta ds a=0 -> (phi-phif)/dt = -gg *
    eta
    phif_expr1 = fd.derivative(VP11, eta, du=vvp0)  # du represents
    perturbation

    # Step-2: f-derivative VP wrt varphi to get interior phi given surface
    update phi
    # int nabla (phi+varphi) cdot nabla delta varphi dx = 0
    phi_expr1 = fd.derivative(VP11, varphii, du=vvp1)
    Fexpr = phif_expr1+phi_expr1
    phi_combo = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(
    Fexpr, result_mixed, bcs = BC_varphi_mixed))
```

```
29
30    # Step-3: f-derivative wrt phi but restrict to free surface to find updater
       eta_new; only solve for eta_new by using exclude
31    eta_expr2 = fd.derivative(VP11, phii, du=v_R)
32    eta_expr = fd.NonlinearVariationalSolver(fd.NonlinearVariationalProblem(
      eta_expr2,eta_new))
```

Step 6: solve the variational problem

Finally, we initiate the time loop and solve the variational problem at each time step. The code for `nvpcase==1` is given as follows:

```
1  while t <= t_end + dt: #
2
3      tt = format(t, '.3f')
4
5      # Solves the variational problem
6      if nvpcase == 1: # VP linear steps 1 and 2 combined # solve of phi
      everywhere steps 1 and 2 combined
7          phi_combo.solve() #
8          phii, varphii = result_mixed.split()
9          eta_expr.solve()
10
11     # Update the values of Functions
12     if nvpcase == 1:  # VP linear steps 1 and 2 combined
13         phi_f.assign(phii)
14         eta.assign(eta_new)
15
16     # Compute energy
17     if nvpcase == 1: # VP linear steps 1 and 2 combined
18         EKin = fd.assemble( 0.5*fd.inner(fd.grad(phii+varphii),fd.grad(phii+
      varphii))*fd.dx )
19         EPot = fd.assemble( 0.5*gg*fd.inner(eta,eta)*fd.ds_t )
20
21     E = EKin+EPot
22
23     t+= dt
```

The complete code is publicly available as **PF_piston_wavetank.py** on the GitHub repository. Following this tutorial, the user can adopt the code to their problem and simulate the wave

waves in a piston-driven numerical wavetank.

7.4 Conclusion

This chapter shares the codes for the simulation of numerical wavetank models based on linear/non-linear shallow water equations and linear/non-linear potential flow equations. For the sake of brevity, the equations are not rewritten, however, the corresponding chapter numbers are mentioned in each section. All the the codes produce '.txt' files which include the evolution of free-surface elevation and velocity potential at different time steps. The user will need MATLAB or Python codes to read the files and plot the data. Furthermore, separate folders are created with '.pvd' files that can be visualized in Paraview.

Chapter 8

Conclusion

8.1 Summary

Undeniably, the maritime industry has a significant role in global trading and the energy sector. Hence, the research and development sector of the maritime industry is actively developing innovative designs for better shipping and energy solutions which require extensive experimental testing. However, the time and budget constraints associated with the experimental campaigns limit the number of tests which compels the researchers to develop affordable numerical tools to assist industry in the design process. To address the industrial demands, we have derived and implemented numerical wavetank models by utilizing a novel approach that can drastically reduce the time and effort for model implementation. Furthermore, the developed numerical wavetank models can be coupled with hyperelastic structure models to perform structure interaction (FSI) analysis.

8.1.1 Overview of objectives and accomplishments

The project aims to develop a practical and effective numerical tool to simulate water-wave interactions with hyperelastic structures, also known as fluid-structure interaction (FSI) analysis. This process has been accomplished in two major steps, as follows:

1. In the first step we developed wavemaker-driven numerical wavetank models that can simulate a wide range of water dynamics, i.e. shallow to deep. These models will be the numerical representation of the physical wavetanks that are employed in maritime facilities. Furthermore, the developed numerical wavetank models are capable of extending

their functionality to include fluid-structure interaction (FSI) analysis of water waves and the hyperelastic structures. Therefore, we have developed a hyperelastic structure model whose formulation is compatible with the numerical wavetank models. A novel feature of our numerical models is that we have developed and tested the numerical implementations of the dynamics via consistent (i.e. stable and conservative) geometric time-discrete variational principles that automate the derivation of discrete weak formulations within *Firedrake* and drastically reduce the time and effort for model implementation. The details of the derivation and implementation of the numerical models developed in this step are discussed in Chapter 2 and 3, and the actual codes are shared in Chapter 6.

2. The numerical results require experimental validation and hence, we performed a series of FSI experiments under a wide range of wave conditions ranging from regular-to-irregular and moderate-to-steep to record the dynamic response of a flexible beam at MARIN's concept basin. The experimental setup is designed such that it is capable of measuring incident waves, beam accelerations, and waves reflected from the beam simultaneously, making it suitable for studying FSI problems experimentally. The data acquired from the experiments are made publicly available with open access via a GitHub repository. Lastly, we used the experimental data for the validation of MARIN in-house FSI solvers which confirmed that the data are suitable for the validation of a variety of FSI numerical solvers, i.e. linear to high-fidelity. The details of the experimental campaign are discussed in Chapter 4 while the validation studies of the FSI solvers are discussed in Chapter 5 and 6.

The numerical models can be further improved to enhance their applicability in industry. Such extensions and improvements will be discussed next.

8.2 Discussion on concomitant extensions

8.2.1 Inclusion of wave absorbing feature

The numerical wavetank models developed in this research project do not include any wave-absorbing feature in this instance. A quick solution is to change the solid-wall boundary condition at the end of the wavetank to a static wave-absorbing boundary condition as explained in Chapter 2. Nevertheless, our objective is to create a numerical model that accurately reflects

the experimental wavetank, which necessitates the inclusion of a numerical beach. Furthermore, the numerical beaches should be adjustable to accommodate the various shapes and types of beaches installed in the experimental facilities. However, there is a challenge associated with the implementation of slopes as the water dynamics transit from deep to shallow when the waves move along the slope or beach which makes a numerical model, solely based on potential flow equations, inadequate to predict water dynamics at beaches. A solution is to couple potential flow equations with shallow water equations such that the bottom-varying part of the numerical wavetank is solved by shallow water equations while potential flow equations are used in other regions. Gidel [34] has presented the derivation and implementation of such beaches by using shallow water equations

8.2.2 Numerical wavetank for FSI analysis

The maritime industry frequently uses wavetank facilities to study the response of fixed and floating offshore structures in water waves. Hence, a numerical wavetank that can implicitly perform two-way coupled FSI analysis is an industrial necessity. To cater to this need, we have developed a hyperelastic structure model which is compatible with the numerical wavetank models based on our novel approach. Nevertheless, there is a challenge associated with the coupling of two governing equations as the potential flow equations are in the Eulerian reference frame while the hyperelastic structure equations are in the Lagrangian frame of reference, thus making the nodes of computational meshes discontinuous at the FSI interface which causes numerical instabilities. Therefore, this is a topic under research and requires further investigation. A temporary solution for this problem is to couple the two systems explicitly, i.e. solve the fluid equations first and interpolate the fluid loads on the structure's mesh. After that, solve the hyperelastic structure equations and after convergence interpolate the structure's deformation back onto the fluid mesh and solve the fluid equations.

8.2.3 Extension and experimental validation of the waveflap driven numerical wavetank

A piston-driven numerical wavetank is capable of simulating shallow- and intermediate-depth water dynamics, as the motion of water particles generated by piston wavemaker resembles shallow-water dynamics. The maritime industry uses piston wavemakers in shallow-water basins while for deep-water basins waveflap wavemakers are deployed. Therefore, a waveflap-driven

numerical wavetank model is an industrial requirement to study ship manoeuvring and the response of offshore floating structures in deep waters. The novel approach we have developed in Chapter 3 can be extended to include waveflap wavemaker. See [69] for the preliminary results obtained from the implementation of the time-discrete variational principle for the waveflap-driven wavemaker. A schematic of the wavetank is demonstrated in Fig. 2.6.

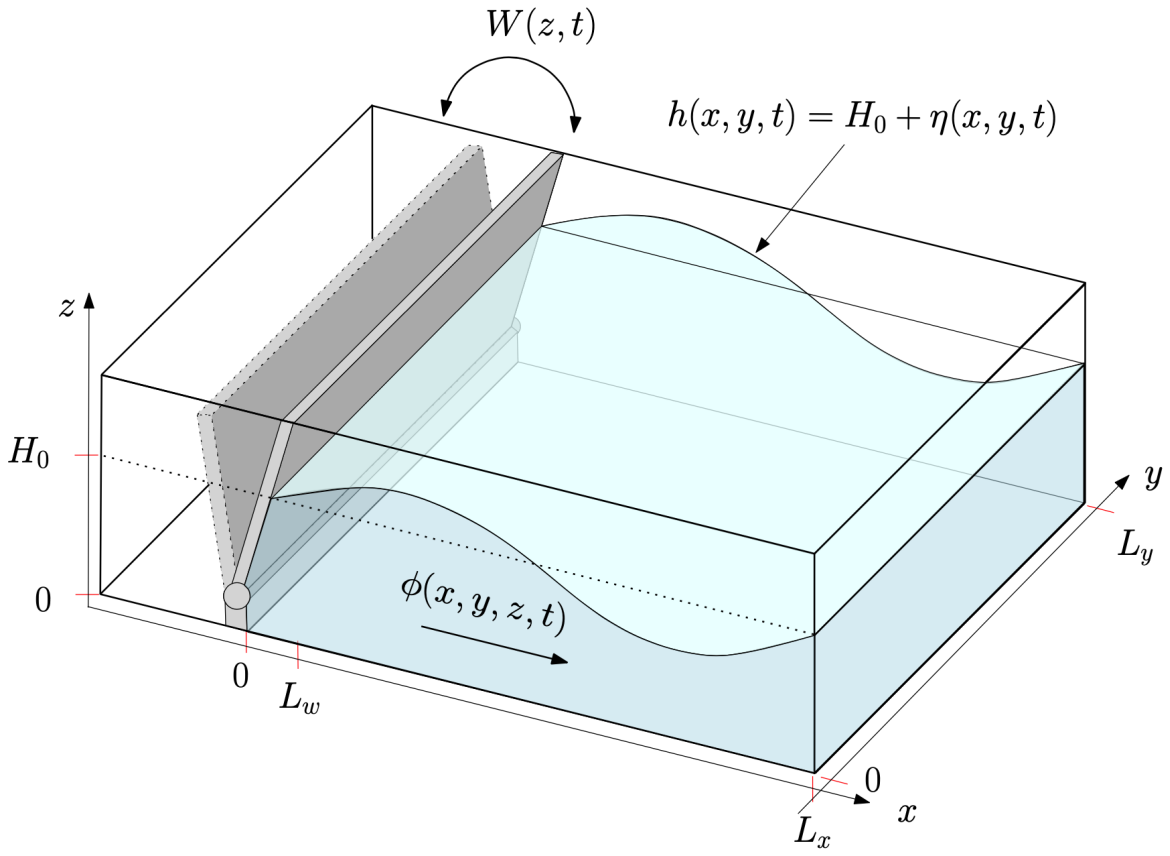


Figure 8.1: Schematic wavetank with waveflap wavemaker, at its left-hand end, having position $x = W(z,t)$; it is this relationship that binds two spatial coordinates.

As we know, experimental validation is an essential step before utilizing a numerical tool in industry. Therefore, to serve the purpose of validation, the experimental study presented in Chapter 4 has been conducted in waveflap-driven wavetank. We have recorded the waveflap displacement and velocity for different wave conditions. This data can be found in '.txt' files shared through GitHub repository. To numerically reproduce the same wavemaker motion as in the experimental study, the measured data is linearly interpolated which then can be assigned to a numerical function at each time step and used as a seed to run the numerical wavetank simulations. The numerical time step is often smaller than the experimental time step due to the

CFL condition; therefore, we interpolate the measured data by using the first-order polynomial. The measured and interpolated wavemaker motion is plotted in Fig. 8.2.

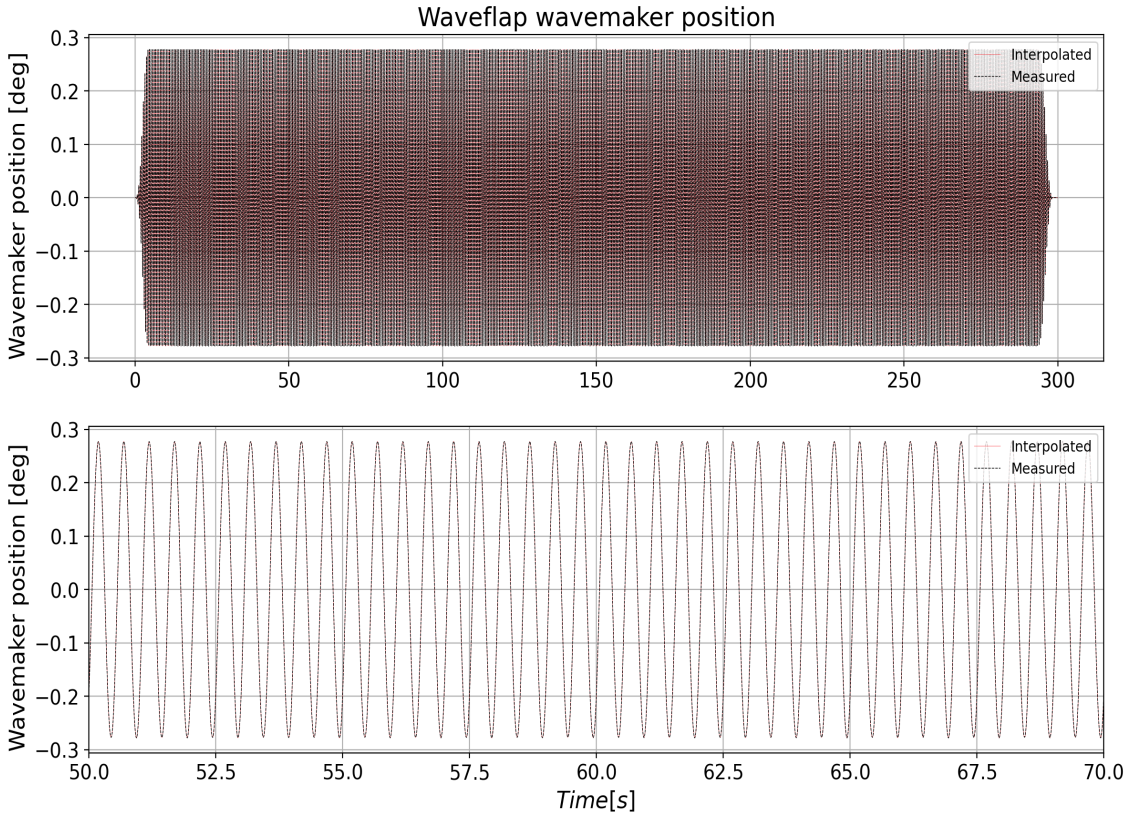


Figure 8.2: The top plot shows the comparison of the interpolated and measured wavemaker motion during one complete test and the bottom plot shows the zoomed-in part of the comparison shown in the top plot.

The plots shown in Fig. 8.2 can be generated by using the shared Python script **waveflap_wavemaker.py** shared on the GitHub repository. The interpolated signal agrees well with the measured signal, as the plots are visually indistinguishable.

8.3 Outreach activities

The outreach activities aim to inspire the non-specialist public by engaging them in the research work. The research work carried out in this European Industry Doctorate (EID) project, i.e. "Eagre/Aegir: High-Seas Wave-Impact Modelling", highlights the importance and role of science, technology, engineering and mathematics (STEM) in our lives. Therefore, the awareness of these topics in the general public is capable of instilling a positive change in society that will have fruitful results for mankind in the long run.

8.4 Online outreach activities

During the first year of the project, the rules related to in-person social activities were restrictive due to the COVID-19 pandemic. Therefore, virtual means for communication and outreach were employed. This section explains the two outreach activities that were carried out virtually.

8.4.1 Event 1: Differential Equations in Real Life

Wajiha Rehman took part as a speaker in an international conference held on 21-06-2021 in Pakistan, titled *"WUM - One-day international webinar on challenges, innovation and opportunities in mathematics"*, organised by Women University Multan, in which scientists with mathematical backgrounds delivered talks and shared their research with undergraduate-level female students of the university. The topic of Wajiha's talk was *"Differential Equations in Real Life"*, which explained the process of mathematical modelling of physical systems, i.e., water waves and fluid-structure interactions between water waves and a hyperelastic beam. The purpose of the talk was to inspire female students in Pakistan to choose mathematics as their future career. The talk can be found on the WUM YouTube channel (*time of presentation at circa 3:00:00 to 3:16:00hr*) at the hyperlink **The Women University Multan Official**.

8.4.2 Event 2: Meet a scientist

Wajiha Rehman joined the "Science Fuse Fellowship Programme" organised by **Science Fuse** to promote science among the local students in Pakistan. As a part of this fellowship, she took part in an interview, in a *"Meet a scientist"* program, in which Wajiha explained her research-project work to a non-mathematical audience at the high-school level. The interview was delivered in Urdu with the remit of inspiring the youth of Pakistan to pursue future careers in engineering and numerical modelling. The interview can be found at the Facebook hyperlink **Science Fuse** (dated 03-07-2021 –one needs to be logged on Facebook– `https://www.facebook.com/page/1602882396609358/search/?q=wajiha`).

8.5 Exhibition in MathsCity

Wajiha Rehman and Yang Lu visited **MathsCity**, located in Trinity shopping centre, Leeds, to carry out the outreach activities. The exhibition was done to introduce the concept of clean energy production by exploiting renewable natural resources, i.e. wind energy, to a group of

home-schooled children. The outreach activity consisted of three steps as follow:

- The first step was the introduction of general concepts. The activity started with a brief presentation on the concept of global warming, the challenges of harvesting wind energy in the oceans, the role of the PhD project in this scenario and the process of simulating water waves and wind turbines.

- The second step was about experimental demonstrations. After explaining the general overview of the project, experimental demonstrations were given by using a scaled wave-tank which had a paper windmill attached at one end. To apply water loads on the turbine's mast the waves were generated manually by using a plunger while a blow dryer was used to artificially generate the wind loads on the turbine's blades. The experimental set-up used for the demonstration is shown in Fig.8.3.

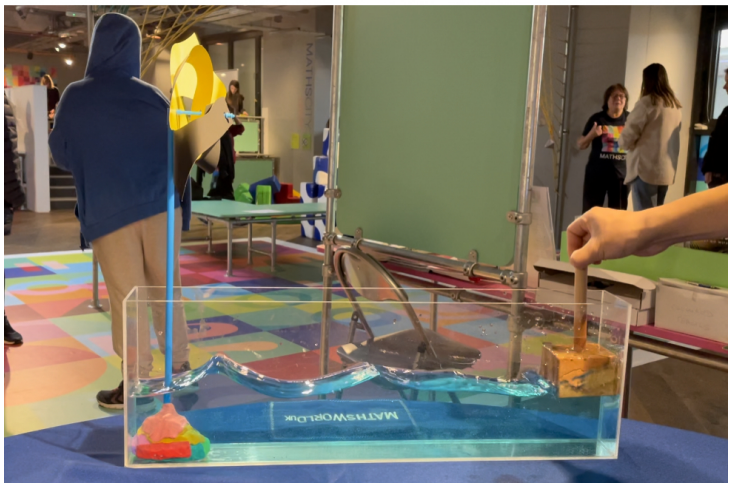


Figure 8.3: The experimental set-up used for the demonstrations is shown.

The purpose of performing these demonstrations was to give an idea of the magnitude of loads that are acting on an offshore wind turbine and let the audience feel the gravity of the situation when there is a sea storm.

- In the last step, the children were encouraged to make their own paper windmills by using the provided material. The children and their parents enjoyed this fun activity. A picture of the kids while making their own paper wind turbines is shown in Fig.8.4

Figure 8.4: Children building paper windmills from the provided material.

The CEO of **MathsCity**, Dr Katie Chicot, has given the following feedback on the Outreach event.

" *Around 40 home school children visited on the day and the staff reported that you had good interactions with the families. In terms of feedback on your activities, the first thing you should ask yourself is if I were to do this again, what would I do differently and what would I keep the same?*

I thought you were enthusiastic and accessible and you are authoritative. I think it is good that you worked together and were part of the presentation rather than making something stand-alone. One point that I noted was that the level of your slides was probably too high for the intended audience. I doubt any of them had seen a function so showing a few variables and functions perhaps didn't match the audience. I think you should build into your outreach plans ways of gauging how your audience is understanding your material and adapting your offering in the moment to that response. Your hands-on activities were at a junior level which was a good balance to the technicality of your slides. MathsCity attracts a range of ages and so its necessary to have a balance of activities that cover a range of maths levels.

Your session at MathsCity was well received and showed a new application of maths that the audience would not have otherwise seen. You definitely fulfilled the stated goal. I hope you enjoyed it and would deliver more outreach in the future. If you do then I think it would be good to build on what you've done already and add more activities and demonstrations."

Based on the feedback from the CEO of MathsCity, Wajiha and Ynag understood that the pre-

sentation content was technically advanced for the invited age group; however, the experimental demonstrations were at the level of the audience and helped in a better understanding of the process. Therefore, in the future, a greater emphasis will be given to the experimental demonstrations as they are more engaging. The scientific and intellectual impact of the activities can additionally be increased in the future by targeting high school students or undergraduates, with some basic knowledge of calculus. This would facilitate comprehension of the underlying mathematical issues to be conveyed, such as the societal dependence on modelling and computational simulation when solving complex real-world problems whose exact solutions elude analytical and theoretical approaches.

8.6 Details of the second outreach activity

On February 26^{th} 2022, Wajiha Rehman and Yang Lu jointly delivered a talk to year 9 students at a "Secondary Mathematics Masterclass" organised by **The Royal Institution** in the MALL of School of Mathematics, the University of Leeds. The topic of the talk was "*Maths for Green Power: Windmills in Sea Waves*". It was a two-hour-long talk which was divided into different sections.

- The first session was based on the explanation of the role of mathematics in real life and the process of carrying out research. The talk started with a discussion to access the current level of understanding of the students, followed by an explanation of the effects of global warming on climate change, and the possible solutions to prevent it. The concepts of green energy and technologies to exploit renewable energy resources were also explained in detail. It was concluded that the knowledge of mathematics is significant during the designing phase of technologies to generate power from renewable energy resources i.e. solar, wind and water.

- In the second session, the concept of "*Research Trinity*" was explained to the students by doing a thought experiment. The students were asked to write down their response to the following question, " *How can you find the temperature in the centre of the cake when the temperature at the boundaries is known?*". The responses were collected and categorised under one of the three pillars of research i.e. theoretical, experimental and numerical. At this stage, the majority of students said that they would use a thermometer to measure the temperature, i.e. experimental ways, and only a few suggested theoretical

and numerical ones. Based on the results, it was found that the majority of the students were unfamiliar with the numerical methods. Therefore, the process of mathematical and numerical modelling for solving the system of equations was explained by using standing waves as an example. The basic terminologies related to waves were taught, which was then followed by the derivation of the exact solution by using intuition instead of rigorous mathematics. Later the results of the numerical solutions were shown and an experimental demonstration was given by using the scaled wavetank.

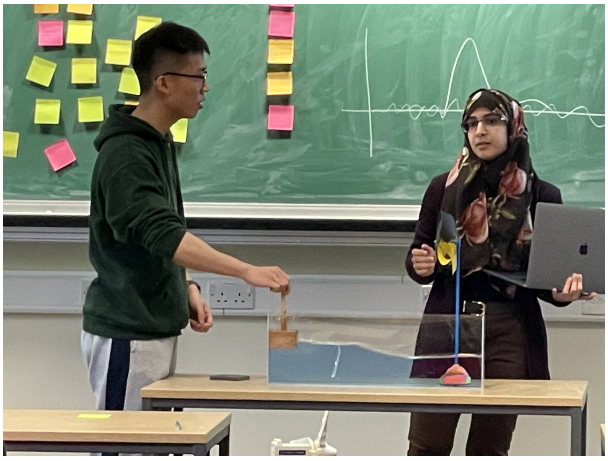


Figure 8.5: Wajiha and Yang while demonstrating the standing waves experimentally. The standing waves were produced in the scaled wavetank to demonstrate that the results obtained by the theoretical model (exact solution) and numerical solution were aligned with the experiments.

- During the third session, both researchers explained their research work focused on potential flow equations with a piston wave-maker, wave-turbine interactions model, and simulations from the mathematical and numerical modelling of water waves.

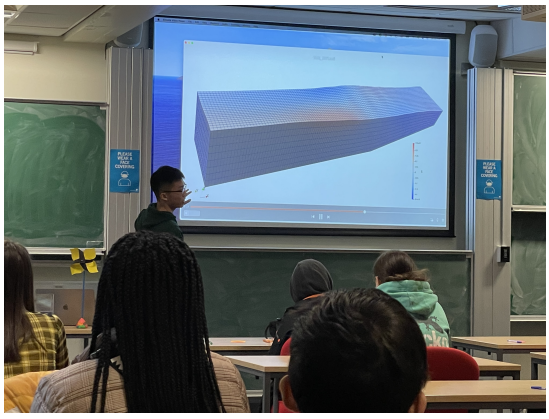


Figure 8.6: Yang while explaining the simulations' results to the students.

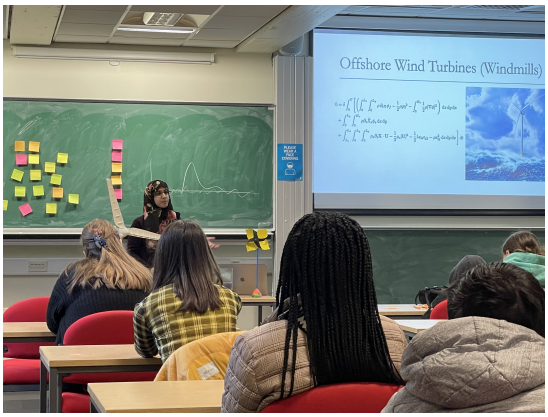


Figure 8.7: Wajiha while explaining the mathematical model used for simulating the wave impact on the turbine's mast.

- The fourth session was based on interactive activities. The students were told to share their views on the advantages and disadvantages of each method, i.e. theoretical, numerical and experimental, and suggest their preferred method. Each student was given a sticky note and was told to write down their answers anonymously. The responses of the student were collected and classified on the basis of their preferred method. This time, the majority of students suggested theoretical and numerical methods over experimental methods. Through this activity, researchers understood that their message related to the advantages of theoretical and numerical methods was well delivered.

Figure 8.8: Classification of the students' responses on the advantages and disadvantages of each method and their favourite method.

Figure 8.9: Captured during the discussion session.

8.7 An online presentation on fluid-structure interactions (FSI)

The final outreach activity of the project is about the explanation of the experimental modelling of the water-wave interactions with a flexible beam. This fluid-structure interaction study was performed at the Maritime Research Institute Netherlands (MARIN). The experimental study is explained in the form of a presentation which is shared publicly on LinkedIn. Therefore, this knowledge can be accessed and shared by a wide range of audiences across the world. The shared presentation starts with the introduction of general concepts related to fluid-structure interactions (FSI) and their significance in real life. After that, it explains how FSI problems are studied in the maritime industry and why benchmark experiments are important. Lastly, it provides insights on the actual experimental campaign that was carried out in MARIN's concept basin and the key results of the experiments.

Appendix A

Derivation of the exact solution of the shallow water equations with piston wave-maker

The equations of motion for nonlinear shallow water equations with piston wavemaker $R(t)$ at the left side boundary are as follows:

$$\delta\phi: \quad h_t + \partial_x(h\phi_x) = 0, \tag{A.1}$$

$$\delta h: \quad \partial_t\phi + \frac{1}{2}|\partial_x\phi|^2 + g(h - H_0) = 0, \quad \text{and} \tag{A.2}$$

$$\delta\phi|_{x=R(t)}: \quad (h\phi_x - R_t h)|_{x=R(t)} = 0, \tag{A.3}$$

which can be linearized around a state of rest, i.e. H_0, by introducing small perturbations of order $O(\epsilon)$ around H_0 as follows:

$$\phi = \phi_0 + \epsilon^1\phi_1 + O(\epsilon^2) + \ldots \quad \text{and,} \tag{A.4}$$

$$\eta = \eta_0 + \epsilon^1\eta_1 + O(\epsilon^2) + \ldots. \tag{A.5}$$

Neglecting the high order terms with $O(\epsilon^2)$ will linearize the equations, as follows:

$$\frac{1}{2}|\partial_x\phi|^2 = O(\phi^2) = O(\epsilon^2) \approx 0. \tag{A.6}$$

The linearization of $\partial_x\phi(x = R(t), t)$ is done by introducing the perturbation variable R in the Taylor expansion at $x = 0$

$$\partial_x\phi(x = R(t), t) = \partial_x\phi(x = 0, t) + \frac{\partial(\partial_x\phi)}{\partial x}|_{x=0}R(t) \approx \partial_x\phi(0, t) = R_t. \tag{A.7}$$

After linearization at H_0, the shallow water equations

$$\partial_t\eta = -H_0\nabla^2\phi, \tag{A.8a}$$

$$\partial_t\phi = -g\eta, \tag{A.8b}$$

$$\partial_x\phi = R_t \quad \text{at} \quad x = 0, \quad \text{and} \tag{A.8c}$$

$$\partial_x\phi = 0 \quad \text{at} \quad x = L, \tag{A.8d}$$

have the following format of solutions in a 1D domain of size L_x with a solid wall at the right side and piston wavemaker $R(t)$ on the left side of the domain. Taking the time derivative of (A.8b) and then putting it into (A.8a) gives an expression for ϕ which is as follows

$$\partial_{tt}\phi = c^2\partial_{xx}\phi, \tag{A.9}$$

where $c^2 = gH_0$ is the wave velocity. This equation is the wave equation which can be separated into a temporal $e^{i\sigma t}$ and spatial $\hat{\phi}(x)$ part by using the separation of variable, as follows:

$$\phi(x, t) = \hat{\phi}(x)\, e^{-i\sigma t}, \tag{A.10}$$

where $\sigma = \dfrac{2\pi}{T_p}$ is the forcing frequency and T_p is the time period of the forcing term. The resulting ordinary differential equation is given as

$$\begin{aligned}
\partial_{tt}(\hat{\phi}(x)\, e^{-i\sigma t}) &= c^2\partial_{xx}(\hat{\phi}(x)\, e^{-i\sigma t}) \\
-\sigma^2\hat{\phi}(x)\, e^{-i\sigma t} &= c^2\, e^{-i\sigma t}\partial_{xx}(\hat{\phi}(x)) \\
\partial_{xx}(\hat{\phi}(x)) &= -\frac{\sigma^2}{c^2}\hat{\phi}(x)
\end{aligned} \tag{A.11}$$

which is the eigenvalue equation for $\hat{\phi}(x)$ and it has a well-known plane wave solution which is given as

$$\begin{aligned}\hat{\phi}(x) &= Ae^{\pm ik_1 x} \\ \hat{\phi}(x) &= Ae^{+ik_1 x} + Be^{-ik_1 x}\end{aligned} \tag{A.12}$$

with wave number $k_p = \sigma/c$. Substituting (A.12) into (A.10) gives

$$\phi(x,t) = e^{-i\sigma t}\left(Ae^{+ik_p x} + Be^{-ik_p x}\right). \tag{A.13}$$

The boundary conditions at the solid wall ($x = L$) and wave-maker ($x = R(t)$) should be satisfied. The boundary condition at the solid wall ($x = L$) says that the velocity of waves in x direction, i.e. $\partial_x \phi$, must be equal to zero, which is stated as

$$\begin{aligned}\frac{\partial \phi}{\partial x}\Big|_{x=L} &= 0 \\ ike^{-i\sigma t}\left(Ae^{+ik_p L} - Be^{-ik_p L}\right) &= 0.\end{aligned} \tag{A.14}$$

Solving (A.14) yields the value of the coefficient B as follows

$$\begin{aligned}Ae^{+ik_p L} &= Be^{-ik_p L} \\ B &= Ae^{2ik_p L}.\end{aligned} \tag{A.15}$$

The value of B is then substituted in (A.12):

$$\begin{aligned}\hat{\phi}(x) &= \left(Ae^{ik_p x} + Be^{-ik_p x}\right) \\ &= Ae^{ik_1 L}\left(e^{ik_p(x-L)} + e^{-ik_p(x-L)}\right).\end{aligned} \tag{A.16}$$

Finally, the hyperbolic identity, i.e. $cosh(ix) = \frac{1}{2}(e^{ix} + e^{-ix}) = cos(x)$, is used to obtain the expression for $\hat{\phi}(x)$

$$\begin{aligned}\hat{\phi}(x) &= 2Ae^{ik_p L}\frac{e^{ik_p(x-L)} + e^{-ik_p(x-L)}}{2} \\ &= \tilde{A}\, cos(k_p(x-L)).\end{aligned} \tag{A.17}$$

Substituting (A.17) into (A.10) results into the expression for $\phi(x,t)$ as follows

$$\phi(x,t) = \tilde{A}e^{-i\sigma t}\, cos(k_p(x-L)). \tag{A.18}$$

Similarly, the boundary condition at the wave-maker ($x = R(t)$) says that the velocity of waves in x direction , i.e. $\partial_x\phi$, must be equal to the velocity of wave-maker which is:

$$R_t = U_0 i\sigma e^{-i\sigma t}, \tag{A.19}$$

and the motion of wavemaker $R(t)$ is given as

$$R(t) = -U_0 e^{-i\sigma t}. \tag{A.20}$$

Therefore, the boundary condition at the wave-maker ($x = R(t)$) is given as

$$\begin{aligned} \frac{\partial\phi}{\partial x}\Big|_{x=0} &= R_t \\ -\tilde{A}k_p e^{-i\sigma t}\, sin(k_p(0-L)) &= U_0 i\sigma e^{-i\sigma t} \\ \tilde{A}k_p\, sin(k_p L) &= U_0 i\sigma. \end{aligned} \tag{A.21}$$

As a result, the value of $\tilde{A}$ is obtained as

$$\tilde{A} = \frac{U_0 i\sigma}{k_p\, sin(k_p L)}. \tag{A.22}$$

Plugging the value of $\tilde{A}$ in (A.18) gives the final expression for $\phi(x,t)$ which is also the particular solution of the problem, denoted as $\phi_p(x,t)$, as follows

$$\phi_p(x,t) = \frac{U_0 i\sigma}{k_p\, sin(k_p L)} e^{-i\sigma t}\, cos(k_p(x-L)). \tag{A.23}$$

The expression for $\eta(x,t)$ is obtained by substituting the expression of $\partial_t\phi$ in (A.8b), as follows

$$\begin{aligned} \eta &= -\frac{1}{g}\frac{\partial\phi}{\partial t} \\ &= \frac{i\sigma}{g}\frac{U_0 i\sigma}{k_p\, sin(k_1 L)} e^{-i\sigma t}\, cos(k_p(x-L)). \end{aligned} \tag{A.24}$$

To avoid the phenomenon of resonance the value of forcing frequency σ must not be equal to the natural frequency of the system. To find the resonance frequency ω, consider the solution $\phi(x,t)$ as follows

$$\phi(x,t) = \hat{\phi}(x)\, e^{-i\omega t} \tag{A.25}$$

where, ω is the natural frequency of the system. Following the same procedure that is used to obtain the eigenvalue equation (A.11) for $\hat{\phi}(x)$ the eigenvalue equation for the spatial part $\hat{\phi_p}(x)$ of the particular $\phi_p(x,t)$ is

$$\hat{\phi}(x) = Ce^{k_2 x} + De^{-k_2 x} \tag{A.26}$$

with wave number $k_2 = \omega/c$. Putting (A.26) into (A.25) gives

$$\phi(x,t) = e^{-i\omega t}\Big(Ce^{ik_2 x} + De^{-ik_2 x}\Big). \tag{A.27}$$

In this case, the boundary condition at $x = L$ says that velocity in x direction i.e. $\partial_x \phi$ is equal to zero. This condition is stated as follows

$$\begin{aligned} \frac{\partial \phi}{\partial x}|_{x=L} &= 0 \\ ik_2\Big(Ce^{ik_2 L} - De^{-ik_2 L}\Big) &= 0 \end{aligned} \tag{A.28}$$

which gives the value of D as

$$D = Ce^{2ik_2 L}. \tag{A.29}$$

Substituting the value of D in (A.26) and then applying the hyperbolic identity i.e. $cosh(ix) = \frac{1}{2}(e^{ix} + e^{-ix}) = cos(x)$, gives

$$\hat{\phi}(x) = \tilde{C} cos(k_2(x - L)). \tag{A.30}$$

Putting (A.30) into (A.25) gives

$$\phi(x,t) = e^{-i\omega t}\tilde{C} cos(k_2(x - L)). \tag{A.31}$$

In the case when no forcing is acting on the domain the boundary condition at $x = 0$ is given as

$$\begin{aligned} \frac{\partial \phi}{\partial x}|_{x=0} &= 0 \\ e^{-i\omega t}\tilde{C} sin(k_2 L) &= 0. \end{aligned} \tag{A.32}$$

As a result of (A.32) the value of $\tilde{C}$ is zero and wave number k_2 is

$$k_2 = \frac{2\pi m_1}{L}, \tag{A.33}$$

and the natural frequency of the system is

$$\omega = ck_2 = c\frac{2\pi m_1}{L} \quad \text{and} \quad m_1 = 1, 2, 3, .. \tag{A.34}$$

After linearisation at H_0, the shallow water equations have the following format of free wave solutions

$$\eta(x,t) = \cos k_1 x \, (A \cos \omega t + B \sin \omega t), \quad \text{and} \tag{A.35a}$$

$$\phi(x,t) = \frac{g}{\omega} \cos k_1 x \, (-A \sin \omega t + B \cos \omega t), \tag{A.35b}$$

which satisfy (A.8b), as follows:

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= g \cos k_1 x \, (-A \cos \omega t - B \sin \omega t) \\ &= -g \cos k_1 x \, (A \cos \omega t + B \sin \omega t) \\ &= -g\eta. \end{aligned} \tag{A.36}$$

Next, considering that $\omega = \sqrt{gH}\sqrt{k_1^2}$ and

$$\frac{\partial \phi}{\partial x} = -k_1 \frac{g}{\omega} \sin k_1 x \, (-A \sin \omega t + B \cos \omega t), \tag{A.37a}$$

$$\frac{\partial^2 \phi}{\partial x^2} = -k_1^2 \frac{g}{\omega} \cos k_1 x \, (-A \sin \omega t + B \cos \omega t), \tag{A.37b}$$

we have

$$\begin{aligned} -H\nabla^2\phi &= -H\left(\frac{\partial^2\phi}{\partial x^2}\right) \\ &= \frac{gH\left(k_1^2\right)}{\omega}\cos k_1 x\left(-A\sin\omega t + B\cos\omega t\right) \\ &= \omega\cos k_1 x\left(-A\sin\omega t + B\cos\omega t\right). \end{aligned} \tag{A.38}$$

In addition,

$$\begin{aligned} \frac{\partial\eta}{\partial t} &= \omega\cos k_1 x\left(-A\sin\omega t + B\cos\omega t\right) \\ &= -H\nabla^2\phi, \end{aligned} \tag{A.39}$$

which means (A.8a) is also satisfied. Finally, the solid wall boundary conditions at the two ends of the domain , i.e. $x = 0$ and $x = L$, should be satisfied as follows

$$\left.\frac{\partial\phi}{\partial x}\right|_{x=0} = -k_1\frac{g}{\omega}\sin k_1 x\left(-A\sin\omega t + B\cos\omega t\right) = 0, \tag{A.40a}$$

$$\left.\frac{\partial\phi}{\partial x}\right|_{x=L_x} = -k_1\frac{g}{\omega}\sin k_1 L_x\left(-A\sin\omega t + B\cos\omega t\right) = 0. \tag{A.40b}$$

This implies $k_1 L_x = 2\pi m_1$. Therefore, the wave number should be $k_1 = 2\pi m_1/L_x$ with m_1 being a positive integer. Therefore, $\omega = \sqrt{gH}\dfrac{2\pi m_1}{L_x}$ The total solution for $\phi(x,t)$ is given as the sum of particular solution i.e. $\phi_p(x,t)$ and homogeneous solution i.e. $\phi_h(x,t)$

$$\phi(x,t) = e^{-i\sigma t}\tilde{A}cos(k_p(x-L)) + \frac{g}{\omega}\cos k_1 x\left(-A\sin\omega t + B\cos\omega t\right) \tag{A.41}$$

and the total solution for $\eta(x,t)$ is given as

$$\eta(x,t) = \frac{i\sigma}{g}\frac{U_0 i\sigma}{k_p\, sin(k_p L)}e^{-i\sigma t}\, cos(k_p(x-L)) + \cos k_1 x\left(A\cos\omega t + B\sin\omega t\right). \tag{A.42}$$

Appendix B

Spatial discretization of the VP for linear potential-flow equations

The corresponding variational principle (VP) for the linear potential-flow equations is given by:

$$0 = \delta \int_0^T \left(\int_{L_x} \phi \partial_t \eta - \frac{1}{2} g \eta^2 \mathrm{d}x - \int_{L_x} \int_0^{H_0} \frac{1}{2} \left| \nabla \phi \right|^2 \mathrm{d}z \mathrm{d}x \right) \mathrm{d}t. \tag{B.1}$$

Substituting the approximation of the unknown variables $\eta(x,t)$ and $\phi(x,t)$ by a finite linear combination of basis functions $\varphi_j(\mathbf{x})$, given as

$$\phi \approx \phi_h(\mathbf{x},t) = \phi_j(t), \varphi_j(\mathbf{x}), \quad \text{and} \quad \eta \approx \eta_h(\mathbf{x},t) = \eta_j(t) \varphi_j(\mathbf{x}). \tag{B.2}$$

into (B.1), and then evaluating the limits and boundary conditions yields the spatially discretized VP as follows:

$$0 = \delta \int_0^T \underbrace{\phi_k M_{kl} \frac{\mathrm{d}\eta_l}{\mathrm{d}t}}_{1} - \underbrace{\frac{1}{2} g \eta_k M_{kl} \eta_l}_{2} - \underbrace{\frac{1}{2} \phi_i S_{ij} \phi_j}_{3} \, \mathrm{d}t. \tag{B.3}$$

Taking the variations of the first term gives:

$$\delta \int_0^T \phi_k M_{kl} \frac{\mathrm{d}\eta_l}{\mathrm{d}t} \mathrm{d}t = \int_0^T \delta\phi_k M_{kl} \frac{\mathrm{d}\eta_l}{\mathrm{d}t} - \frac{\mathrm{d}\phi_k}{\mathrm{d}t} M_{kl} \delta\eta_l \, \mathrm{d}t. \tag{B.4}$$

The variations of the second term gives:

$$\delta \int_0^T \frac{1}{2} g \eta_k M_{kl} \eta_l \, \mathrm{d}t = \int_0^T g \eta_k M_{kl} \delta\eta_l \, \mathrm{d}t. \tag{B.5}$$

In (B.3), the stiffness matrix S_{ij} in the third term represents the whole domain which includes free surface nodes (denoted by subscripts k and l) as well as inner fluid domain (denoted by subscripts i' and j'), therefore it can be rewritten as

$$S_{ij} = \begin{bmatrix} S_{kl} & S_{kj'} \\ S_{i'l} & S_{i'j'} \end{bmatrix}. \tag{B.6}$$

Then we can split $S_{ij}\phi_i\phi_j$ in the third term of (B.3) as follows

$$\begin{aligned} \phi_i S_{ij} \phi_j &= \phi_k S_{kj} \phi_j + \phi_{i'} S_{i'j} \phi_j \\ &= \phi_k S_{kl} \phi_l + \phi_k S_{kj'} \phi_{j'} + \phi_{i'} S_{i'l} \phi_l + \phi_{i'} S_{i'j'} \phi_{j'}. \end{aligned} \tag{B.7}$$

After substituting (B.7) into the third term of (B.1) and taking the variations gives

$$\begin{aligned} &\delta \int_0^T \frac{1}{2} \phi_i S_{ij} \phi_j \, \mathrm{d}t \\ &= \int_0^T \delta\phi_k S_{kl} \phi_l + \frac{1}{2} \phi_k S_{kj'} \delta\phi_{j'} + \frac{1}{2} \delta\phi_k S_{kj'} \phi_{j'} + \frac{1}{2} \phi_{i'} S_{i'l} \delta\phi_l + \frac{1}{2} \delta\phi_{i'} S_{i'l} \phi_l + \phi_{i'} S_{i'j'} \delta\phi_{j'} \, \mathrm{d}t \\ &= \int_0^T \delta\phi_k S_{kl} \phi_l + \phi_k S_{kj'} \delta\phi_{j'} + \delta\phi_k S_{kj'} \phi_{j'} + \phi_{i'} S_{i'j'} \delta\phi_{j'} \, \mathrm{d}t. \end{aligned} \tag{B.8}$$

After substituting the variations of all the terms into(B.1), the following

$$\begin{aligned} 0 = \int_0^T & M_{kl} \delta\phi_k \frac{\mathrm{d}\eta_l}{\mathrm{d}t} - M_{kl} \frac{\mathrm{d}\phi_k}{\mathrm{d}t} \delta\eta_l - g M_{kl} \eta_k \delta\eta_l - \delta\phi_k S_{kl} \phi_l - \phi_k S_{kj'} \delta\phi_{j'} \\ & - \delta\phi_k S_{kj'} \phi_{j'} - \phi_{i'} S_{i'j'} \delta\phi_{j'} \, \mathrm{d}t \\ = \int_0^T & \left(-M_{kl} \frac{\mathrm{d}\phi_k}{\mathrm{d}t} - g \eta_k M_{kl} \right) \delta\eta_l + \left(M_{kl} \frac{\mathrm{d}\eta_l}{\mathrm{d}t} - S_{kl} \phi_l - S_{kj'} \phi_{j'} \right) \delta\phi_k \\ & - \left(\phi_k S_{kj'} + \phi_{i'} S_{i'j'} \right) \delta\phi_{j'} \, \mathrm{d}t. \end{aligned} \tag{B.9}$$

Equation (B.9) is equal to eq(4.32) in [72] excluding the terms related to coupling and beam. Finally, using the arbitrariness of variations $\delta\eta_l$, $\delta\phi_k$ and $\delta\phi_{j'}$, we can obtain a system of ordinary

and algebraic equations from (B.9), as follows

$$\delta\eta_l : \quad \frac{\mathrm{d}\phi_k}{\mathrm{d}t} M_{kl} + g\eta_k M_{kl} = 0, \tag{B.10}$$

$$\delta\phi_k : \quad M_{kl}\frac{\mathrm{d}\eta_l}{\mathrm{d}t} - S_{kj}\phi_j = 0, \tag{B.11}$$

$$\delta\phi_{j'} : \quad \phi_{i'} S_{i'j'} + \phi_k S_{kj'} = 0. \tag{B.12}$$

In addition, the discretized Laplace equation (B.12) shows that interior degrees of freedom can be eliminated:

$$\phi_{i'} = -S^{-1}_{i'j'}\left(S_{kj'}\phi_k\right) = -S^{-1}_{j'i'}\left(S_{kj'}\phi_k\right) = -S_{kj'}S^{-1}_{j'i'}\phi_k. \tag{B.13}$$

Substitute (B.7) into (B.3) and use (B.13), the variational principle becomes

$$\begin{aligned}
0 = \delta \int_0^T & \phi_k M_{kl}\frac{\mathrm{d}\eta_l}{\mathrm{d}t} - \frac{1}{2}g\eta_k M_{kl}\eta_l - \frac{1}{2}\Big(\phi_k S_{kl}\phi_l + \phi_k S_{kj'}\phi_{j'} + \phi_{i'}S_{i'l}\phi_l \\
& + \phi_{i'}S_{i'j'}\phi_{j'}\Big)\mathrm{d}t \\
= \delta \int_0^T & \phi_k M_{kl}\frac{\mathrm{d}\eta_l}{\mathrm{d}t} - \frac{1}{2}g\eta_k M_{kl}\eta_l - \frac{1}{2}\Big[\phi_k S_{kl}\phi_l + \phi_k S_{kj'}\phi_{j'} - S_{i'l}S^{-1}_{j'i'}(S_{kj'}\phi_k) \\
& - S_{i'j'}(S_{kj'}S^{-1}_{j'i'}\phi_k)\phi_{j'}\Big]\mathrm{d}t \\
= \delta \int_0^T & \phi_k M_{kl}\frac{\mathrm{d}\eta_l}{\mathrm{d}t} - \frac{1}{2}g\eta_k M_{kl}\eta_l - \frac{1}{2}\phi_k B_{kl}\phi_l \,\mathrm{d}t,
\end{aligned} \tag{B.14}$$

where $B_{kl} = S_{kl} - S_{kj'}S^{-1}_{j'i'}S_{i'l}$ is the Schur complement. Equation (B.14) is equal to eq(4.36) in Salwa's thesis [72] excluding the terms related to coupling and beam.

Appendix C

Availability of data

This section explains the online platform that is used to share the experimental data, the data arrangement into different folders, and the content of each folder. An open, public-access GitHub the repository has been created to share all experimental data[1].

C.0.1 Main folders

The repository has seven main folders corresponding to each experimental case.

1. *Exp1_carriage_rest_0.25m*

    - The data generated during experimental case 1 when the submerged depth of the beam's free end was 0.25m is shared in this folder. The folder has eight sub-folders which correspond to each wave condition. The wave parameters corresponding to each sub-folder are listed in Table. C.1.

Table C.1: Regular-wave parameters and characteristics when the carriage is at rest and 0.25m of the beam is submerged in water.

Sub-folder	H	T	λ	Steepness (H/λ)
	[m]	[s]	[m]	[-]
1_51	0.126	1	1.560	0.081
2_52	0.282	1.5	3.509	0.080
3_55	0.016	0.5	0.390	0.041
4_56	0.062	1	1.560	0.040
5_57	0.14	1.5	3.509	0.040
6_58	0.25	2	6.239	0.040
7_59	0.39	2.5	9.748	0.040
8_60	0.016	0.58	0.525	0.0305

[1]Data are also available upon request.

2. *Exp1_carriage_rest_0.5m*

   - The measurements corresponding to the wave parameters in experimental case 1 when the beam's submerged free-end length is 0.5m are shared in this folder. This folder also has eight sub-folders which correspond to each wave condition. The wave parameters corresponding to each sub-folder are listed in Table. C.2.

Table C.2: Regular-wave parameters and characteristics when the carriage is at rest and 0.5m of the beam is submerged in water.

Sub-folder	H	T	λ	Steepness (H/λ)
	[m]	[s]	[m]	[-]
0_50	0.032	0.5	0.39	0.082
1_51	0.126	1	1.56	0.081
2_52	0.282	1.5	3.51	0.080
3_55	0.016	0.5	0.39	0.041
4_56	0.062	1	1.56	0.040
5_57	0.14	1.5	3.51	0.040
6_58	0.25	2	6.24	0.040
7_60	0.016	0.58	0.52	0.030

3. *Exp2_carriage_moving_0.25m*

   - This folder shares the measurements obtained from the interactions of the regular waves with the flexible beam while the carriage was moving at a constant speed. The free end of the beam is submerged at 0.25m. This main folder has four sub-folders which correspond to each wave condition. The wave parameters corresponding to each sub-folder are listed in Table. C.3.

Table C.3: Regular-wave parameters and characteristics when the carriage is moving at a constant speed and 0.25m of the beam is submerged in water.

Sub-folder	H	T	λ	Steepness (H/λ)	Speed	Encounter frequency
	[m]	[s]	[m]	[-]	[m/s]	[rad/s]
1_51	0.126	1	1.560	0.081	0.297	7.480
2_55	0.016	0.5	0.390	0.041	0.149	14.967
3_56	0.062	1	1.560	0.040	0.297	7.480
4_57	0.14	1.5	3.509	0.040	0.446	4.987

4. *Exp2_carriage_moving_0.5m*

   - The results obtained in experimental case 2 when the submerged depth of the beam's free end is 0.5m are shared in this folder. The folder has four sub-folders which correspond to each wave condition and are listed in Table. C.4.

Table C.4: Regular-wave parameters and characteristics when the carriage is moving at a constant speed and 0.5m of the beam is submerged in water.

Sub-folder	H	T	λ	Steepness (H/λ)	Speed	Encounter frequency
	[m]	[s]	[m]	[-]	[m/s]	[rad/s]
1_51	0.126	1	1.56	0.081	-0.215	5.417
2_55	0.016	0.5	0.39	0.041	-0.1077	10.831
3_56	0.062	1	1.56	0.040	-0.2154	5.415
4_57	0.14	1.5	3.51	0.040	0.6864	5.418

5. *Exp3_irreg_waves_0.25m*

   - The results obtained in experimental case 3, when the submerged depth of the beam's free end is 0.25m, are shared in this folder. The folder has three sub-folders which correspond to each wave condition and are listed in Table. C.5.

Table C.5: Irregular-wave parameters and characteristics when the carriage is at rest and 0.25m of the beam is submerged in water.

Sub-folder	MARIN test no. 70065_02CB_2	Environment	Time	Irregular sea characteristics			
				JONSWAP type spectrum			
				Hs	Tp	Dir.	γ
			[s]	[m]	[s]	[deg]	[-]
North Sea state							
gain01	011_001_01	Gain 1.0	1781	0.34	2.25	180	2.9
gain_one_fourth	011_001_01	Gain 0.25	1781	0.085	2.25	180	2.9
gain_half	011_001_01	Gain 0.5	1781	0.17	2.25	180	2.9

6. *Exp3_irreg_waves_0.5m*

   - Similar to *Exp3_irreg_waves_0.25m*, this folder contains measurements for the water-beam interactions when the submerged depth of the beam's free end is 0.5m. The corresponding sub-folder and the wave parameters and characteristics are given in Table.C.6.

Table C.6: Irregular-wave parameters and characteristics when the carriage is at rest and 0.5m of the beam is submerged in water.

Folder name	MARIN test no. 70065_02CB_2	Environment	Time	Irregular sea characteristics			
				JONSWAP type spectrum			
				Hs	Tp	Dir.	γ
			[s]	[m]	[s]	[deg]	[-]
North Sea state							
gain_01	011_001_01	Gain 1.0	1781	0.34	2.25	180	2.9
gain_half	011_001_01	Gain 0.5	1781	0.17	2.25	180	2.9

7. *hammer_tests*

   - A hammer test is an experimental method for measuring a structure's natural response/frequency. This folder shares the *.h5m* files obtained from the dry and wet hammer tests. The file *dry_004_01.h5m* contains accelerations of the beam when hammer test is performed in air. A dry hammer test is performed to obtain the natural frequency of the beam in the air when it is unaffected by the water. The wet hammer tests are performed when the beam is submerged. The purpose of wet hammer tests is to observe the effect of water on the natural frequency of the beam, therefore, wet hammer tests are performed with two depths of the submerged beam's end, that is, 0.25m and 0.5m. The file *wet1_001_01.h5m* and *wet2_006_01.h5m* corresponds to the wet hammer tests with submerged end depth of 0.25m and 0.5m respectively.

C.0.2 Sub-folders

All the main folders have several sub-folders and each sub-folder consists of mainly two types of files, i.e. *.pan* and *.h5m*. The files with extension *.pan* states the general information about experimental tests and sensors in text format, which is depicted in Fig C.1.

Final preliminary analysis (Model) Project 80372.702

Test 80372_02CB_05_051_001_01
AEGRE
Proeven 0.25m
Wave 51
1

Facility CB
Scale 1.000

Ch.	Name	Dim.	Sensor	Loc.	Range Max	Range Min	Freq	Sample Amount	+/-	Mean	StdDev	Max	Min
1	C.IRIG	V	E-1-001-B-8005	LA24-8	100000.00	-100000.00	200 Hz	59995		-35.739	1.054	-33.072	-38.977
2	C.SPEED	m/s	S-1-281-F-9201	LA24-5	12.00	-12.00	200 Hz	59995		0.000	0.000	0.002	-0.002
3	C.FOTOPULS	V	E-1-010-V-9201	LA24-1	10.00	-10.00	200 Hz	59995		0.000	0.000	0.000	0.000
5	WAVE.FORE	m	W-1-874-V-0601	WA04-5	0.54	-0.46	200 Hz	59995		0.000	0.043	0.096	-0.069
6	WAVE.SB	m	W-1-874-V-0458	WA04-3	0.50	-0.50	200 Hz	59995		0.000	0.042	0.090	-0.064
7	AX.1	m/s^2	A-1-730-V-0012	LA25-1	20.05	-19.95	1.2 kHz	359973		0.005	0.909	4.445	-2.265
8	AY.1	m/s^2	A-1-730-V-0011	LA25-2	-19.81	20.19	1.2 kHz	359973		0.000	0.147	0.716	-0.814
9	AZ.1	m/s^2	A-1-730-V-0010	LA25-3	-29.88	10.12	1.2 kHz	359973		0.004	0.029	0.910	-0.795
10	AX.2	m/s^2	A-1-730-V-0006	LA25-4	19.97	-20.03	1.2 kHz	359973		0.004	0.676	3.312	-1.691
11	AY.2	m/s^2	A-1-730-V-0005	LA25-5	-20.22	19.78	1.2 kHz	359973		-0.001	0.111	0.535	-0.630
12	AZ.2	m/s^2	A-1-730-V-0004	LA25-6	-29.86	10.14	1.2 kHz	359973		0.004	0.041	0.933	-0.721
13	AX.3	m/s^2	A-1-730-V-0036	LA25-7	19.89	-20.11	1.2 kHz	359973		0.004	0.453	2.255	-1.188
14	AY.3	m/s^2	A-1-730-V-0035	LA25-8	-19.99	20.01	1.2 kHz	359973		0.001	0.080	0.387	-0.529
15	AZ.3	m/s^2	A-1-730-V-0034	LA18-1	-29.82	10.18	1.2 kHz	359973		0.001	0.039	0.864	-0.667
16	AX.4	m/s^2	A-1-730-V-0051	LA18-2	19.96	-20.04	1.2 kHz	359973		0.003	0.255	1.319	-0.729
17	AY.4	m/s^2	A-1-730-V-0050	LA18-3	-19.86	20.14	1.2 kHz	359973		0.000	0.048	0.332	-0.373
18	AZ.4	m/s^2	A-1-730-V-0049	LA18-4	-29.88	10.12	1.2 kHz	359973		0.001	0.035	0.787	-0.620
19	AX.5	m/s^2	A-1-730-V-0024	LA18-5	19.97	-20.03	1.2 kHz	359973		0.003	0.094	0.507	-0.456
20	AY.5	m/s^2	A-1-730-V-0023	LA18-6	-19.76	20.24	1.2 kHz	359973		0.001	0.024	0.213	-0.269
21	AZ.5	m/s^2	A-1-730-V-0022	LA18-7	-29.87	10.13	1.2 kHz	359973		0.000	0.027	0.692	-0.568
22	AX.6	m/s^2	A-1-730-V-0018	LA18-8	20.08	-19.92	1.2 kHz	359973		0.002	0.031	0.181	-0.217
23	AY.6	m/s^2	A-1-730-V-0017	LA10-1	-19.77	20.23	1.2 kHz	359973		0.000	0.010	0.148	-0.142
24	AZ.6	m/s^2	A-1-730-V-0016	LA10-2	-29.83	10.17	1.2 kHz	359973		0.000	0.021	0.569	-0.485

Figure C.1: Format of the *.pan* files.

There are three rows and the third row is divided into several columns. The second row states the information related to the experimental test, for example, the test number (80372_02CB_05_051_001_01), project name (AEGRE), submerged depth of the beam (Proeven 0.25m), gain (1), facility name (CB stands for concept basin), and scale (1.000). The first column of the third row shows the abbreviated sensor names. The abbreviation of the names which are used to plot the measurements are given in Table C.7.

Table C.7: Description of the sensor names mentioned in *.pan* file.

Name	Description
C.SPEED	Speed of the carriage
WAVE.FORE	Wave elevation measured by the probe which is located at the front of the beam (26.25 m away from the wavemaker)
WAVE.SB	Wave elevation measured by the probe which is in parallel to the beam (30 m away from the wavemaker)
AX.i	Accelerations of the beam in x direction recorded by the accelerometer, where i denotes the accelerometer number
AY.i	Accelerations of the beam in y direction recorded by the accelerometer, where i denotes the accelerometer number
AZ.i	Accelerations of the beam in z direction recorded by the accelerometer, where i denotes the accelerometer number
Flap 3 Pos	Position of the waveflap wavemaker

The number with the accelerations, e.g. AX.1, AY.2, and AZ.3, denotes the position of the accelerometer along the beam. The accelerometers are numbered from 1 to 6, where accelerometer number 1 is at the submerged free end of the beam while accelerometer number 6 is located at the fixed end of the beam. The rest of the accelerometers are numbered 2 to 5 from the free end to the fixed end. The second column shows the unit of the measurements, the fifth and sixth columns show the maximum and minimum range of the sensors respectively; the seventh column shows the frequency; and the last column shows the statistical parameters for the sensors' measurement. On the other hand, the files with extension *.h5m* contain the actual time-domain measurements obtained from the sensors. Each *.h5m* from the experimental case contains acceleration signal from all six accelerometers in the x-direction, wave elevation measured by the probe that is 26.25 m away from the wavemaker, wave elevation measured by the probe that is 30 m away from the wavemaker, carriage speed, and variation waveflap position throughout the run. These measurements can be read with the help of post-processing code. The post-processing codes based on MATLAB and Python scripts, with comments, are shared. The names of the MATLAB and Python scripts are **read_model_tst.m** and **read_model_tst.py** respectively. Each script needs the name of the *.h5m* file as user input. In addition to reading the *.h5m* file, the script plots the signals from the sensors. The MATLAB and Python scripts which are used to read and post-process the signals are shared next.

C.1 Post-processing codes

This section shares a MATLAB script, with comments, to read the *.h5m* files produced during the experimental study which is explained in Chapter 4 of this dissertation. The name of this script in GitHub repository is *read_model_tst.m*.

```
clear

%%# Put the file name in commas
h5mfile_in='80372_02CB_05_051_001_01.h5m';

%% Read data from accelerometers
t1=h5read(h5mfile_in,'/1200.31 Hz/Time'); %# time axis for accelerometers
AX1=h5read(h5mfile_in,'/1200.31 Hz/AX.1'); %# accelerometer at the free end
AX2=h5read(h5mfile_in,'/1200.31 Hz/AX.2');
AX3=h5read(h5mfile_in,'/1200.31 Hz/AX.3');
AX4=h5read(h5mfile_in,'/1200.31 Hz/AX.4');
AX5=h5read(h5mfile_in,'/1200.31 Hz/AX.5');
AX6=h5read(h5mfile_in,'/1200.31 Hz/AX.6'); %# accelerometer at the fixed end
%%# Wave Probes and carriage
t_wave=h5read(h5mfile_in,'/200.05 Hz/Time'); %# time axis for the waves
WAVE_FORE=h5read(h5mfile_in,'/200.05 Hz/WAVE.FORE'); % probe in front of the
    beam (26.25 m away from wavemaker)
WAVE_SB=h5read(h5mfile_in,'/200.05 Hz/WAVE.SB'); %# probe parallel to the beam
    (30 m away from wavemaker)

%%# Waveflap wavemaker position
t3 = h5read(h5mfile_in,'/100.00 Hz/Time'); %# time axis for the wavemaker
flap_pos=h5read(h5mfile_in,'/100.00 Hz/Flap 3 Pos'); % position of the waveflap
     wavemaker

%%# Waveflap Wavemaker position
plot(t3, flap_pos,'DisplayName', 'Wavemaker position')
ax = gca;
ax.FontSize = 15;
xlabel('Time [s]')
ylabel('Wavemaker position [deg]')
grid

%#  Top plot
```

```
figure
ax1 = nexttile;
plot(t3, flap_pos,'DisplayName', 'Wavemaker position')
title(ax1,'Wavemaker position [deg]')
ax1.FontSize = 20;
ylabel('Wavemaker position [deg]')
grid(ax1,'on')

%# Bottom plot
ax2 = nexttile;
plot(t_wave,WAVE_SB,'DisplayName', 'WAVE_SB')
title(ax2,'Wave elevation measured by the probe ')
ax2.FontSize = 20;
ylabel('Wave elevation [m]')
xlabel('Time [s]')
grid(ax2,'on')

%%# Top plot
figure
ax1 = nexttile;
plot(t_wave,WAVE_SB,'DisplayName', 'WAVE_SB')
title(ax1,'Plot 1')
ax1.FontSize = 20;
ylabel('Wave elevation')
grid(ax1,'on')

%# Bottom plot
ax2 = nexttile;
plot(t1, AX1,'DisplayName', 'AX1')
title(ax2,'Plot 2')
ax2.FontSize = 20;
xlabel('Time [s]')
ylabel('Acceleration')
grid(ax2,'on')
```

For demonstration, the provided MATLAB script is used to plot the comparison of the wavemaker position with the wave elevation measured by the wave probe that is 26.25 m away from the wavemaker.

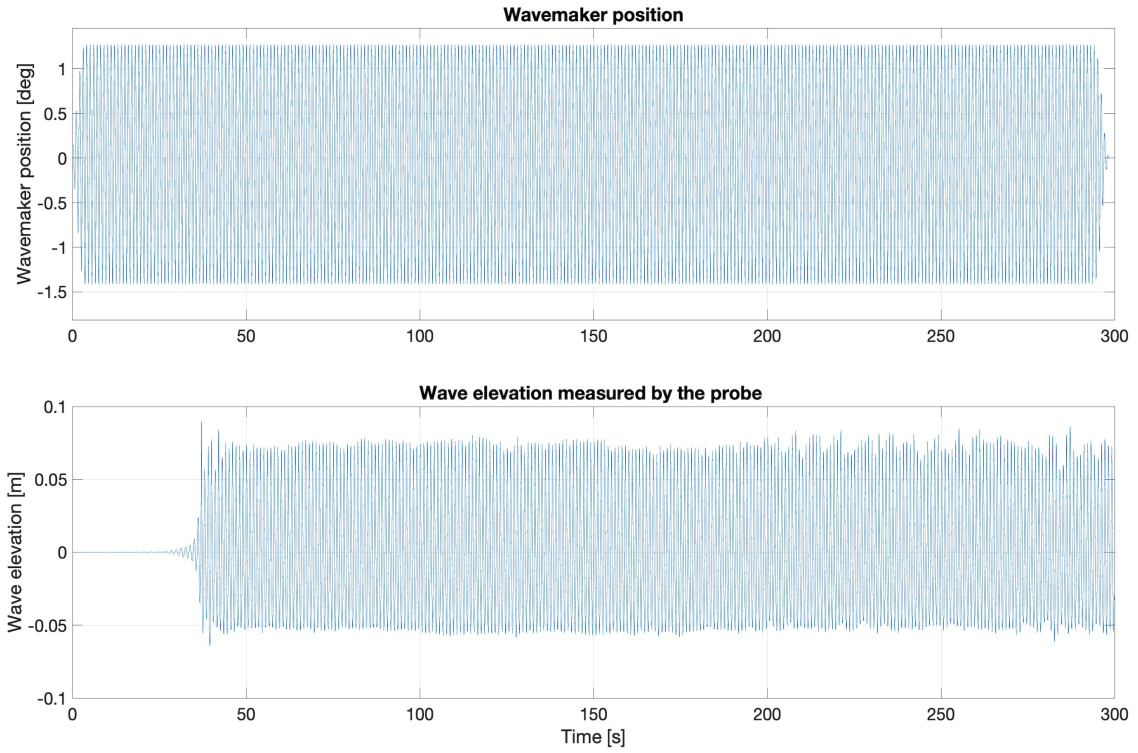


Figure C.2: The top plot shows the variation of the position of the waveflap wavemaker as time proceeds. The bottom plot shows the signals measured by the wave probe which is located in front of the beam.

A visual analysis of the top plot in Fig C.2 shows that the wavemaker starts moving gradually from 0 seconds and takes approximately 4 to 5 seconds to reach its maximum position because the wavemaker motion is ramped when it is about to commence and terminate. This ramped motion helps to keep the free surface smooth. It can be noticed from the bottom plot that the wave probe starts to measure the wave elevation at approximately 26 seconds which is because the probe is located 26.25 m away from the wavemaker. Similar to the wavemaker motion, the initial wave elevation signal detected by the probe is small in amplitude which gradually increases to maximum amplitude. In this case, the wave period is 1 second and the wavelength is 1.56 m. The test ran for 300 seconds which means the probe measures the reflected waves in the last 40 seconds of the test. Although, in this case, the effect of the reflected wave on the incident wave is not significant, it is preferable to use the wave data which is not affected by the reflected waves for code validation purposes.

A Python script has been shared on the GitHub site to read the *.h5m* files. The script with comments is shown below, the script needs the file name as user input.

```
1 # https://docs.h5py.org/en/stable/quick.html
2
```

```
import h5py
import numpy as np
import matplotlib.pyplot as plt

# Write file name within commas
filename = "exp1_c1.h5m"

#__________________  FIGURE PARAMETERS  _____________________#

tsize = 18 # font size of image title
tsize2 = 12
tic_size = 14
size = 16  # font size of image axes
tic_size = 14

#_________________ READ .h5m FILE  ___________________________#

with h5py.File(filename, "r") as f:
    # Print all root level object names (aka keys)
    # these can be group or dataset names
    print("Keys: %s" % f.keys())

    # Time array for the signals obtained from accelerometers
    t1 = f['/1200.31 Hz/Time'][()]

    # Time array for the signals obtained from  wave probe
    t2 = f['/200.05 Hz/Time'][()]

    # Time array for the signals obtained from  waveflap wavemaker position
    t3 = f['/100.00 Hz/Time'][()]

    # Accelerations in x-direction obtained from accelerometer at the submerged
     free end of the beam (AX1)
    AX1 = f['/1200.31 Hz/AX.1'][()]

    # Accelerations in x-direction obtained from accelerometer above the
    submerged free end of the beam (AX2)
    AX2 = f['/1200.31 Hz/AX.2'][()]

```

```
    # Accelerations in x-direction obtained from the third accelerometer from
    the submerged free end of the beam (AX3)
    AX3 = f['/1200.31 Hz/AX.3'][()]

    # Accelerations in x-direction obtained from the fourth accelerometer from
    the submerged free end of the beam (AX4)
    AX4 = f['/1200.31 Hz/AX.4'][()]

    # Accelerations in x-direction obtained from the fifth accelerometer from
    the submerged free end of the beam (AX5)
    AX5 = f['/1200.31 Hz/AX.5'][()]

    # Accelerations in x-direction obtained from the  accelerometer at the
    fixed end of the beam (AX6)
    AX6 = f['/1200.31 Hz/AX.6'][()]

    # Wave elevation obtained from the probe at the front of the beam
    Wave_fore = f['/200.05 Hz/WAVE.FORE'][()]

    # Wave elevation obtained from the probe parallel to the beam
    Wave_sb = f['/200.05 Hz/WAVE.SB'][()]

    # carriage speed
    C_speed = f['/200.05 Hz/C.SPEED'][()]

    # Waveflap wavemaker motion
    Wave_maker = f['/100.00 Hz/Flap 3 Pos'][()]

##____________________ Time step details ____________________##

time_2 = np.array(t2)
dt_wave_elevation = time_2[1] - time_2[0]
print('Time step for wave elevation =', dt_wave_elevation)

time_3 = np.array(t3)
dt_wavemaker = time_3[1] - time_3[0]
print('Time step for the wavemaker motion =', dt_wavemaker )

##_________________  PLOT SIGNALS ____________________________##
```

```
75
76 fig, (ax1, ax2) = plt.subplots(2)
77 ax1.set_title('waveflap wavemaker position',fontsize=tsize)
78 ax1.plot(t3, Wave_maker, 'r-', linewidth= 0.2 ,label = '$Wavemaker pos$ ')
79 ax1.set_ylabel('Wavemaker position [deg]  ',fontsize=size)
80 ax1.tick_params(axis='x', labelsize= tic_size)
81 ax1.tick_params(axis='y', labelsize= tic_size)
82 ax1.grid()
83
84 ax2.set_title('Wave elevation of the incoming wave',fontsize=tsize)
85 ax2.plot(t2, Wave_fore,  'b-', linewidth= 0.2 ,label = '$Wave_fore$ ')
86 ax2.set_xlabel('$Time [s]$ ',fontsize=size)
87 ax2.set_ylabel('Wave elevation [m] ',fontsize=size)
88 ax2.tick_params(axis='x', labelsize= tic_size)
89 ax2.tick_params(axis='y', labelsize= tic_size)
90 ax2.grid()
91
92
93 fig, (ax1, ax2) = plt.subplots(2)
94 ax1.set_title('Acceleration of the submerged free end of the beam',fontsize=
    tsize)
95 ax1.plot(t1, AX1, 'r-', linewidth= 0.2 , label = '$AX.1$ ')
96 ax1.set_ylabel('Accelerations in x-direction [m/s]  ',fontsize=size)
97 ax1.tick_params(axis='x', labelsize= tic_size)
98 ax1.tick_params(axis='y', labelsize= tic_size)
99 ax1.grid()
100
101 ax2.set_title('Wave elevation of the incoming wave',fontsize=tsize)
102 ax2.plot(t2, Wave_fore,  'b-', linewidth= 0.2 , label = '$Wave_fore$ ')
103 ax2.set_xlabel('$Time [s]$ ',fontsize=size)
104 ax2.set_ylabel('Wave elevation [m] ',fontsize=size)
105 ax2.tick_params(axis='x', labelsize= tic_size)
106 ax2.tick_params(axis='y', labelsize= tic_size)
107 ax2.grid()
108
109 plt.show()
110 print("---------  Program ends ----------")
```

As a demonstration of the Python script, one of the plots that are generated by the Python

script is shared in Fig. C.3, which compares the incident wave and the beam's response to the incident wave in the form of two sub-plots.

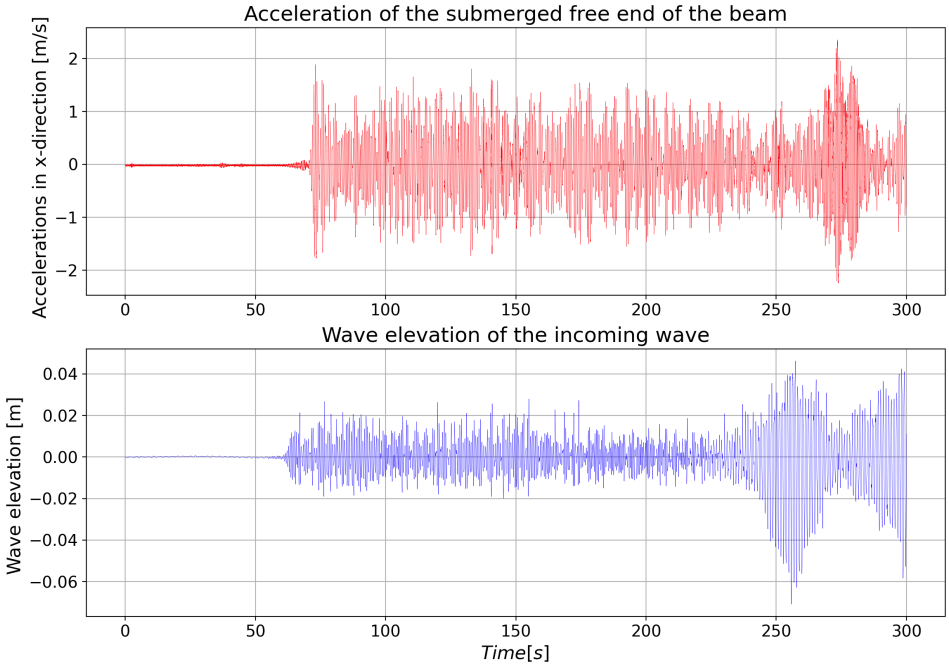


Figure C.3: The top plot shows the accelerations obtained from the accelerometer located at the submerged free-end of the beam when the incident wave, shown in the bottom plot, interacted with the beam.

It can be noticed that the signals in the last 50 seconds of the run are affected by the reflected waves. Therefore, the user should use the part of the signal that is not affected by the reflected waves for validation purposes.

References

[1] M.S. Alnaes. *UFL: a finite element form language*. In: Automated Solution of Differential Equations by the Finite Element Method by Logg, Mardal, and Wells (Eds), 2011.

[2] M.S. Alnaes et al. *Unified Form Language: A domain-specific language for weak formulations of partial differential equations*. Tech. rep. 2013. URL: `https://arxiv.org/pdf/1211.4047.pdf`.

[3] E. Bachynski, M. Thys, and V. Delhaye. "Dynamic response of a monopile wind turbine in waves: Experimental uncertainty analysis for validation of numerical tools". In: *Applied Ocean Research* 89 (2019), pp. 96–114.

[4] S. Balay et al. "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries". In: *Modern Software Tools in Scientific Computing*. Ed. by E. Arge, A. M. Bruaset, and H. P. Langtangen. Birkhäuser Press, 1997, pp. 163–202.

[5] S. Balay et al. *Petsc users manual revision 3.8*. Tech. rep. Office of Scientific and Technical Information (OSTI), 2017.

[6] F.K. Benra et al. "A comparison of one-way and two-way coupling methods for numerical analysis of fluid-structure interactions". In: *Journal of applied mathematics* 2011 (2011).

[7] S. Bhattacharya et al. "Soil-Structure Interactions (SSI) for offshore wind turbines". In: *IET Engineering and Technology Reference* 24.16 (2017).

[8] M. Bhinder et al. "Numerical and experimental study of a surging point absorber wave energy converter". In: *Proceedings of the 8th European Wave and Tidal Energy Conference, Uppsala, Sweden*. 2009, pp. 7–10.

[9] R. D. Blevins and R. Plunkett. “Formulas for natural frequency and mode shape”. In: *Journal of Applied Mechanics* 47.2 (1980), p. 461.

[10] O. Bokhove and A. Kalogirou. “Variational water wave modelling: from continuum to experiment”. In: *Bridges, T., Groves, M., and Nicholls, D., LMS Lecture Note Series, Cambridge University Press* 426 (2016), pp. 226–260.

[11] S.Y. Boo. “Linear and nonlinear irregular waves and forces in a numerical wave tank”. In: *Ocean Engineering* 29.5 (2002), pp. 475–493.

[12] J. Boussinesq. *Essai sur la théorie des eaux courantes*. Impr. nationale, 1877.

[13] F.J. Brink, F. Ferenc Izsák, and J.J.W. van der Vegt. “Hamiltonian Finite Element Discretization for Nonlinear Free Surface Water Waves”. In: *J. Sci. Comp.* 73 (2017), pp. 366–394.

[14] T. Bunnik, J. Helder, and E.J. de Ridder. “Deterministic simulation of breaking wave impact and flexible response of a fixed offshore wind turbine”. In: *International Conference on Offshore Mechanics and Arctic Engineering*. Vol. 56574. American Society of Mechanical Engineers. 2015, V009T09A049.

[15] T. Bunnik et al. *ReFRESCO theory manual*. Tech. rep. MARIN Academy, Wageningen, 2017.

[16] John R Chaplin and P Teigen. “Steady flow past a vertical surface-piercing circular cylinder”. In: *Journal of Fluids and Structures* 18.3-4 (2003), pp. 271–285.

[17] J. Choi and S.B. Yoon. “Numerical simulations using momentum source wave-maker applied to RANS equation model”. In: *Coastal Engineering* 56.10 (2009), pp. 1043–1060.

[18] Anil K Chopra. “Modal analysis of linear dynamic systems: physical interpretation”. In: *Journal of structural engineering* 122.5 (1996), pp. 517–527.

[19] G.F. Clauss, C.E. Schmittner, and R. Stuck. “Numerical wave tank: Simulation of extreme waves for the investigation of structural responses”. In: *International Conference on Offshore Mechanics and Arctic Engineering*. Vol. 41979. 2005, pp. 785–792.

[20] J.A. Cooke, C.A. McMahon, and M.R. North. “Sources of error in the design process”. In: *Recent Advances in Integrated Design and Manufacturing in Mechanical Engineering* (2003), pp. 421–430.

[21] G. Coulibaly et al. “Urban flood modeling using 2D shallow-water equations in Ouagadougou, Burkina Faso”. In: *Water* 12.8 (2020), p. 2120.

[22] A.J.C. Crespo et al. “Towards simulating floating offshore oscillating water column converters with smoothed particle hydrodynamics”. In: *Coastal Engineering* 126 (2017), pp. 11–26.

[23] J. Davidson, S. Giorgi, and J.V. Ringwood. “Linear parametric hydrodynamic models for ocean wave energy converters identified from numerical wave tank experiments”. In: *Ocean Engineering* 103 (2015), pp. 31–39.

[24] *Det Norske Veritas, Design of offshore wind turbine structures, Tech. Rep. DNV-OS-J101.* Tech. rep. Det Norske Veritas (DNV), 2017.

[25] Geological Digressions. *Tsunamis behave as shallow-water waves.* https://www.geological-digressions.com/tsunamis-behave-as-shallow-water-waves/. 2024.

[26] S. van Essen et al. “Linking experimental and numerical wave modelling”. In: *Journal of Marine Science and Engineering* 8.3 (2020), p. 198.

[27] D.J. Ewins. *Modal testing: theory, practice and application.* John Wiley & Sons, 2009.

[28] W. B. Feng et al. “Study on wave spectra in south coastal waters of Jiangsu”. In: *Applied Mechanics and Materials* 212 (2012), pp. 193–200.

[29] P. Ferrant. “Time domain computation of nonlinear diffraction loads upon three dimensional floating bodies”. In: *ISOPE International Ocean and Polar Engineering Conference.* ISOPE. 1995, ISOPE–I.

[30] Firedrake. *Defining variational problems.* `https://www.firedrakeproject.org/variational-problems.html`. Accessed: 2024-02-19. 2024.

[31] E Gagarina et al. “On variational and symplectic time integrators for Hamiltonian systems”. In: *Journal of computational physics* 306 (2016), pp. 370–389.

[32] E. Gagarina. “Variational approaches to water wave simulations”. PhD thesis. University of Twente, 2014.

[33] E. Gagarina et al. “On variational and symplectic time integrators for Hamiltonian systems”. In: *J. Comput. Phys.* 306 (2016), pp. 370–389.

[34] F. Gidel. “Variational water-wave models and pyramidal freak waves”. `https://etheses.whiterose.ac.uk/21730/`. PhD thesis. University of Leeds, 2018.

[35] F. Gidel, O. Bokhove, and M.A. Kelmanson. “Driven nonlinear potential flow with wave breaking at shallow-water beaches”. In: *Int. Conf. on Offshore Mechanics and Arctic Engineering*. Vol. 57632. American Society of Mechanical Engineers. 2017, V001T01A053.

[36] F. Gidel et al. “Variational and numerical modelling strategies for cost-effective simulations of driven free-surface wave”. 2022. URL: `https://eartharxiv.org/repository/view/3411/`.

[37] E. Hairer et al. “Geometric numerical integration”. In: *Oberwolfach Reports* 3.1 (2006), pp. 805–882.

[38] K. Hasselmann et al. “Measurements of wind-wave growth and swell decay during the Joint North Sea Wave Project (JONSWAP).” In: *Ergaenzungsheft zur Deutschen Hydrographischen Zeitschrift, Reihe A* (1973).

[39] J He and ZF Fu. “Mathematics for modal analysis”. In: *Modal Analysis* 2001 (2001), pp. 12–48.

[40] N.E. Huang et al. “A unified two-parameter wave spectral model for a general sea state”. In: *Journal of Fluid Mechanics* 112 (1981), pp. 203–224.

[41] N.G. Jacobsen, D.R. Fuhrman, and J. Fredsøe. “A wave generation toolbox for the open-source CFD library: OpenFoam®”. In: *International Journal for numerical methods in fluids* 70.9 (2012), pp. 1073–1088.

[42] Z. Jiang. "Installation of offshore wind turbines: A technical review". In: *Renewable and Sustainable Energy Reviews* 139 (2021), p. 110576.

[43] S.H. Jongsma. *On a method for performing fluid-structure interaction simulations with refresco (internal technical report 80165-12-RD).* Tech. rep. MARIN Academy, Wageningen, 2016.

[44] S.H. Jongsma, E.T.A. van der Weide, and J. Windt. *Implementation and verification of a partitioned strong coupling fluid-structure interaction approach in a finite volume method (internal technical report).* Tech. rep. MARIN Academy, Wageningen, 2016.

[45] J. Jonkman et al. *Definition of a 5-MW reference wind turbine for offshore system development.* Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2009.

[46] Garbis H. Keulegan and Lloyd H. Carpenter. "Forces on Cylinders and Plates in an Oscillating Fluid". In: *Journal of Research of the National Bureau of Standards* 60.5 (1858).

[47] J.W. Kim et al. "Technical and economic readiness review of CFD-based numerical wave basin for offshore floater design". In: *Offshore Technology Conference.* OTC. 2016, D011S014R002.

[48] Bonguk Koo et al. "Reynolds and Froude number effect on the flow past an interface-piercing circular cylinder". In: *International Journal of Naval Architecture and Ocean Engineering* 6.3 (2014), pp. 529–561.

[49] A. Kumar and T. Weir. "Wind power in Fiji: A preliminary analysis of the Butoni wind farm". In: *International solar energy society conference.* 2008.

[50] R. Kurnia, Badriana M.R., and E. van Groesen. "Hamiltonian Boussinesq Simulations for Waves Entering a Harbor with Access Channel". In: *J. Waterway, Port, Coastal, and Ocean Eng.* 144 (2017).

[51] C. Lanczos. *The variational principles of mechanics.* Courier Corporation, 2012.

[52] T.T. Le, D.H. Phung, and V.C. Tran. "Numerical simulation of tidal flow in Danang Bay Based on non-hydrostatic shallow water equations". In: *Pacific Journal of Mathematics for Industry* 8.1 (2016), p. 1.

[53] U.J. Lee, W.M. Jeong, and H.Y. Cho. "Estimation and analysis of JONSWAP spectrum parameter using observed data around Korean coast". In: *Journal of Marine Science and Engineering* 10.5 (2022), p. 578.

[54] P. Lin and P.L.F. Liu. "Internal wave-maker for Navier-Stokes equations models". In: *Journal of waterway, port, coastal, and ocean engineering* 125.4 (1999), pp. 207–215.

[55] X. Lu et al. "A CFD study of focused extreme wave impact on decks of offshore structures". In: *International Conference on Offshore Mechanics and Arctic Engineering*. Vol. 45400. American Society of Mechanical Engineers. 2014, V002T08A047.

[56] J. C. Luke. "A variational principle for a fluid with a free surface". In: *J. Fluid Mechanics* 27.2 (1967), pp. 395–397.

[57] MARIN. *ReFRESCO*. `https://www.marin.nl/en/facilities-and-tools/software/refresco`. Accessed: 2023-08-01. 2023.

[58] MARIN. *SEACAL Theory Manual*. Tech. rep. CRS SEACAL Working Group, 2023.

[59] *MARIN Concept Basin*. `file:///Users/mmwr/Downloads/Concept_Basin.pdf`. Accessed: 19-11-2022.

[60] Leonard Meirovitch. *Fundamentals of vibrations*. Waveland Press, 2010.

[61] J. Orszaghova, A.GL. Borthwick, and P. H. Taylor. "From the paddle to the beach–A Boussinesq shallow water numerical wave tank based on Madsen and Sørensen's equations". In: *Journal of Computational Physics* 231.2 (2012), pp. 328–344.

[62] C.R. Ortloff and M.J. Krafft. "Numerical Test Tank: Simulation of Ocean Engineering Problems by Computational Fluid Dynamics". In: *Offshore Technology Conference*. OTC. 1997, OTC–8269.

[63] J. Park, D. Cho, and T. Jang. “A numerical experiment on a new piston-type wavemaker: Shallow water approximation”. In: *International Journal of Naval Architecture and Ocean Engineering* 15 (2023), p. 100535.

[64] W. J. Pierson Jr and L. Moskowitz. “A proposed spectral form for fully developed wind seas based on the similarity theory of SA Kitaigorodskii”. In: *Journal of geophysical research* 69.24 (1964), pp. 5181–5190.

[65] K. Ram, M.R. Ahmed, and Y.H. Lee. “Experimental Study of Wave Forces on an Offshore Wind Turbine Tower Model”. In: *2017 4th Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)*. IEEE. 2017, pp. 265–270.

[66] F. Rathgeber et al. “Firedrake: automating the finite element method by composing abstractions”. In: *ACM Transactions on Mathematical Software (TOMS)* 43.3 (2016), pp. 1–27.

[67] W. Rehman, O. Bokhove, and M.A. Kelmanson. “A Systematic Approach for Developing a Numerical Wavetank to Simulate Driven Shallow- and Deep-Water Waves”. In: *Proc. ASME 2023 42nd Int. Conf. on Ocean, Offshore and Arctic Eng.* ASME. 2023, p. 10.

[68] W. Rehman, S. Paboeuf, and J.P. Tomy. “A Comparison of Different Fluid-Structure Interaction Analysis Techniques for the Marine Propeller”. In: *ASME Power Conference*. Vol. 85109. American Society of Mechanical Engineers. 2021, V001T12A008.

[69] W. Rehman et al. “Experimental modeling of water-wave interactions With a flexible beam”. In: *Int. Conf. on Offshore Mechanics and Arctic Eng.* Vol. 86892. American Society of Mechanical Engineers. 2023, V007T08A022.

[70] W. Rehman et al. “Experimental Modelling of Water-Wave Interactions with a Flexible Beam”. In: *EarthArXiv eprints* (2024), X5998B.

[71] T. Salwa, O. Bokhove, and M.A. Kelmanson. “Variational modelling of wave-structure interactions with an offshore wind-turbine mast”. In: *J. Eng. Math.* 107.1 (2017), pp. 61–85.

[72] T.J. Salwa. "On variational modelling of wave slamming by water waves". http://etheses.whiterose.ac.uk/23778/. PhD thesis. University of Leeds, 2018.

[73] P. Sassi et al. "Simulation of vorticity wind turbines". In: *Heliyon* 6.10 (2020), e05155.

[74] P. Schmitt et al. "The opportunities and limitations of using CFD in the development of wave energy converters". In: *Marine & Offshore Renewable Energy* (2012), pp. 89–97.

[75] D.M. Storer. "Dynamic analysis of non-linear structures using higher order frequency response functions". PhD thesis. The University of Manchester (United Kingdom), 1991.

[76] M.A. Storti et al. "Fluid-structure interaction with a staged algorithm. Aerospace applications." In: *CIMEC Document Repository* (2006).

[77] L. Suja-Thauvin et al. "Experimental results of a multimode monopile offshore wind turbine support structure subjected to steep and breaking irregular waves". In: *Ocean Engineering* 146 (2017), pp. 339–351.

[78] K. Tanizawa. "The state of the art on numerical wave tank". In: *Proc. of 4th Osaka Colloquium on Seakeeping Performance of Ships, 2000*. 2000, pp. 95–114.

[79] *The University of Edinburgh*. https://www.eng.ed.ac.uk/about/news/20170329/marine-energy-testing-tank-sets-sights-new-horizons. Accessed: 2022-07-20.

[80] A.C. Varsoliwala and T.R. Singh. "Mathematical modeling of tsunami wave propagation at mid ocean and its amplification and run-up on shore". In: *Journal of Ocean Engineering and Science* 6.4 (2021), pp. 367–375.

[81] W. Wang et al. "A fully nonlinear potential flow wave modelling procedure for simulations of offshore sea states with various wave breaking scenarios". In: *Applied Ocean Res.* 117 (2021), p. 102898.

[82] F. M. White. *Fluid Mechanics*. McGraw Hill Higher Education, 2006.

[83] D.C. Wilcox et al. *Turbulence modeling for CFD*. Vol. 2. DCW industries La Canada, CA, 1998.

[84] C. Windt, J. Davidson, and J.V. Ringwood. "High-fidelity numerical modelling of ocean wave energy systems: A review of computational fluid dynamics-based numerical wave tanks". In: *Renewable and Sustainable Energy Reviews* 93 (2018), pp. 610–630.

[85] C. Windt et al. "On the assessment of numerical wave makers in CFD simulations". In: *Journal of Marine Science and Engineering* 7.2 (2019), p. 47.

[86] J. Yan et al. "Computational free-surface fluid–structure interaction with application to floating offshore wind turbines". In: *Computers & Fluids* 141 (2016), pp. 155–174.

[87] C. Yang and R.C. Ertekin. "Numerical simulation of nonlinear wave diffraction by a vertical cylinder". In: *Journal of Offshore Mechanics and Arctic Engineering* 114.1 (Feb. 1992), pp. 36–44.